Πανεπιστημιο Πατρων

Διπλωματικη Εργασια

Χρονικά Γραφήματα

Συγγραφέας:
Ελένη Ακρίδα

Επιβλέπων:
Παύλος Σπυράκης, Καθηγητης

Υποβάλλεται προς εκπλήρωση των απαιτήσεων για το Μεταπτυχιακό Δίπλωμα Ειδίκευσης

στο

Τμήμα Μαθηματικών

4 Ιουνίου 2013

Η τριμελης εξεταστικη Επιτροπη

Παύλος Σπυράκης	*Χαράλαμπος Ζαγούρας*	*Φίλιππος Αλεβίζος*
Καθηγητης (Επιβλεπων)	Καθηγητης	Αναπλ. Καθηγητης

Δήλωση Συγγραφικής Πατρότητας

Εγώ, η Ελένη Ακρίδα, δηλώνω πως, βάσει όσων γνωρίζω, το περιεχόμενο της παρούσας διπλωματικής εργασίας με τίτλο «Χρονικά Γραφήματα» και όσα παρουσιάζονται σε αυτήν είναι προϊόν δικής μου δουλειάς και υπάρχουν αναφορές σε όλες τις πηγές που χρησιμοποίησα.

Δηλώνω υπεύθυνα ότι η παρούσα εργασία για τη λήψη του μεταπτυχιακού τίτλου σπουδών του Διατμηματικού Μεταπτυχιακού Προγράμματος 'Μαθηματικά των Υπολογιστών και των Αποφάσεων' δεν έχει υποβληθεί ούτε έχει εγκριθεί στο πλαίσιο κάποιου άλλου μεταπτυχιακού ή προπτυχιακού τίτλου σπουδών, στην Ελλάδα ή στο εξωτερικό.

Signed: _____

Date: _____

"I don't know what I may seem to the world, but as to myself, I seem to have been only like a boy playing on the sea-shore and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me."

Isaac Newton

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

Περίληψη

Σχολή Θετικών Επιστημών
Τμήμα Μαθηματικών

Μεταπτυχιακό Δίπλωμα Ειδίκευσης

Χρονικά Γραφήματα

από την Ελένη Ακρίδα

Στη διπλωματική εργασία προς παρουσίαση, πραγματευόμαστε ένα νέο είδος γραφημάτων, τα χρονικά γραφήματα, και διάφορες παραλλαγές τους.

Ένα χρονικό γράφημα είναι μια διατεταγμενη τριάδα $G = \{V, E, L\}$, όπου:

- V μη κενό πεπερασμένο σύνολο (καλείται σύνολο κορυφών)
- E σύνολο m στοιχείων, καθένα από τα οποία είναι δισύνολο στοιχείων του V (καλείται σύνολο ακμών), και
- $L = \{L_e, \forall e \in E\} = \{L_{e_1}, L_{e_2}, \ldots, L_{e_m}\}$, όπου $L_{e_i},\ i = 1, \ldots, m$, σύνολο θετικών ακεραίων τιμών που αντιστοιχίζονται στην ακμή $e_i \in E$ (καλείται ανάθεση χρονικών ετικετών ή απλώς ανάθεση)

Οι τιμές που αντιστοιχίζονται σε κάθε ακμή του γραφήματος καλούνται χρονικές ετικέτες της ακμής και δηλώνουν τις χρονικές στιγμές, κατά τις οποίες έχουμε τη δυνατότητα να τη διασχίσουμε (από το ένα της άκρο προς το άλλο).

Για να αντιληφθεί κανείς το ενδιαφέρον των χρονικών γραφημάτων, μπορεί να σκεφτεί τη δυνατότητα εφαρμογής τους στην καθημερινότητα. Για παράδειγμα, οι χρονικές ετικέτες που ανατίθενται σε μία ακμή ενός *κατευθυνόμενου*[1] χρονικού γραφήματος μπορούν να παραλληλιστούν με τις ώρες, στις οποίες γίνονται αναχωρήσεις αεροπλάνων από μία πόλη προς μια άλλη. Έτσι, η μελέτη των χρονικών γραφημάτων θα μπορούσε να συμβάλει στην οργάνωση των πτήσεων ενός αεροδρομίου.

Ένα χρονικό μονοπάτι (ή "ταξίδι") σε ένα χρονικό γράφημα είναι ένα μονοπάτι, στις ακμές του οποίου μπορούμε να βρούμε αυστηρά αύξουσα σειρά χρονικών ετικετών.

Στην εργασία, μεταξύ άλλων, γίνεται μελέτη της συνδετικότητας στα χρονικά γραφήματα, καθώς και κατασκευή και μελέτη αλγορίθμων εύρεσης χρονικών μονοπατιών ("ταξιδίων") που φθάνουν το δυνατόν συντομότερα στον προορισμό τους (τελική κορυφή μονοπατιού). Επιπλέον, μελετώνται στατιστικά τα Χρονικά Γραφήματα, με επικέντρωση στο αναμενόμενο πλήθος χρονικών μονοπατιών σε ένα γράφημα, καθώς και στη Χρονική Διάμετρο ενός γραφήματος, όπως αυτή ορίζεται στην εργασία.

[1] Όπως και στα γραφήματα, έτσι και στα χρονικά γραφήματα, η κατεύθυνση στις ακμές ενός γραφήματος σημαίνει πως τα στοιχεία του συνόλου E είναι διατεταγμένα ζεύγη στοιχείων του συνόλου V

UNIVERSITY OF PATRAS

Abstract

School of Science
Department of Mathematics

Master of Science

Temporal Graphs

by Eleni Akrida

In the thesis, we are dealing with a new type of graphs,called Temporal Graphs, and several variants.

A temporal graph is an ordered triplet $G = \{V, E, L\}$, where:

- V stands for a nonempty finite set (called set of vertices)
- E stands for a set of m elements, each of which are 2-element subsets of V (called set of edges), and
- $L = \{L_e, \forall e \in E\} = \{L_{e_1}, L_{e_2}, \ldots, L_{e_m}\}$, where $L_{e_i},\ i = 1, \ldots, m$, is a set of positive integers mapped to edge $e_i \in E$ (called assignment of time labels or simply assignment)

The values assigned to each edge of the graph are called time labels of the edge and indicate the times at which we can cross it (from one end to the other).

In order to understand the interest of temporal graphs, one may think their applicability to everyday life. For example, the time labels assigned to an edge of a *directed*[2]

[2]The direction in the edges of a graph means that the elements of the set E are ordered pairs of elements of V

temporal graph can be paralleled to the flight departure times from one city to another. Therefore, the study of temporal graphs could contribute to the organization of flights at an airport.

A temporal path (or "journey") in a temporal graph is a path, on the edges of which we can find strictly ascending time labels.

In the thesis, among others, we study the connectivity of temporal graphs and we construct and study several algorithms that find temporal paths which arrive the soonest possible at their destination (final vertice of the path).

Furthermore, we examine temporal graphs statistically, focusing on the expected number of temporal paths in a graph as well as in the Temporal Diameter of a graph, also defined in the thesis.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε από την Ελένη Ακρίδα, μεταπτυχιακή φοιτήτρια του ΔΠΜΣ 'Μαθηματικά των Υπολογιστών και των Αποφάσεων' του Πανεπιστημίου Πατρών. Η μελέτη και εκπόνηση της εργασίας παραγματοποιήθηκε κατά το ακαδημαϊκό έτος 2012-2013 υπό την επίβλεψη του κ. Παύλου Σπυράκη, καθηγητή του τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Πατρών.

Στον κ. Παύλο Σπυράκη οφείλω τις θερμές μου ευχαριστίες για την καθοδήγηση και την πολύτιμη υποστήριξή του καθ' όλη τη διάρκεια διεκπεραίωσης της παρούσας εργασίας.

Ιδιαίτερες ευχαριστίες θα ήθελα να απευθύνω επίσης στον κ. Χαράλαμπο Ζαγούρα, καθηγητή του Τμήματος Μαθηματικών του Πανεπιστημίου Πατρών, για τη συμβολή του στην αγάπη που ανέπτυξα στη Θεωρία Γραφημάτων, μετέπειτα αντικείμενο της μελέτης μου.

Ευχαριστώ επίσης τους καθηγητές του τμήματος Μαθηματικών του Πανεπιστημίου Πατρών, κ. Δημήτρη Καββαδία, κ. Παναγιώτη Αλεβίζο και κ. Φίλιππο Αλεβίζο, για τις συμβουλές και το χρόνο που μου αφιέρωσαν κατά τη συγγραφή και διόρθωση της εργασίας.

Τέλος, ευχαριστώ από καρδιάς τους γονείς μου, Χρήστο και Ειρήνη, και τα αδέλφια μου, Εβίνα και Γιώργο, για τη συνεχή συμπαράσταση, την αγάπη και την κατανόηση που έδειξαν όλο αυτόν τον καιρό.

Ελένη Ακρίδα

Περιεχόμενα

Κατάλογος σχημάτων

Σύμβολα

Σύμβολο		***Σελίδα***
n	πλήθος κορυφών γραφήματος ή χρονικού γραφήματος	2
m	πλήθος ακμών γραφήματος ή χρονικού γραφήματος	2
(u, v, l)	χρονική ακμή με άκρα u, v και ετικέτα διαθεσιμότητας l	2
$d(\ )$	στιγμή αναχώρησης ταξιδίου	3
$a(\ )$	στιγμή άφιξης ταξιδίου	3
$age(\ )$	ηλικία χρονικού γραφήματος	20
$U - RTD$	ομοιόμορφο τυχαίο κατευθυνόμενο χρονικό γράφημα	61
$U - RTG$	ομοιόμορφο τυχαίο μη κατευθυνόμενο χρονικό γράφημα	61
d	χρονική διάμετρος	61
$\tau(\ )$	αριθμός χρονικότητας χρονικού γραφήματος	13
$\delta(\ ,\ )$	χρονική απόσταση ζεύγους κορυφών	61

Αφιερωμένη στους γονείς μου,
που με έκαναν να αγαπήσω τα Μαθηματικά,
μου έμαθαν την επιμονή, την υπομονή και τη
σημασία του καθορισμού στόχων στη ζωή.

Κεφάλαιο 1

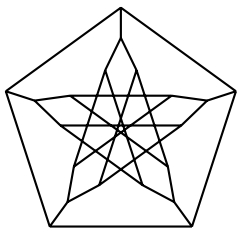
Εισαγωγή στα Χρονικά Γραφήματα

1.1 Θεωρία Γραφημάτων

Τα Διακριτά Μαθηματικά είναι κλάδος των μαθηματικών που μελετά μαθηματικές δομές, οι οποίες είναι εκ φύσεως διακριτές ή αριθμήσιμες και στις οποίες δεν υπάρχει η έννοια της συνέχειας. Η θεωρία γραφημάτων είναι ο κλάδος εκείνος των Διακριτών Μαθηματικών που μελετά τα γραφήματα.

1.1.1 Μη κατευθυνόμενα Γραφήματα

Γράφημα: *Διατεταγμένο ζεύγος G={V,E}, όπου V ένα μη κενό πεπερασμένο σύνολο που καλούμε σύνολο κορυφών και E σύνολο που καλείται σύνολο ακμών και του οποίου τα στοιχεία είναι δισύνολα των στοιχείων του V.*
(Συνηθίζεται όταν αναφερόμαστε στα σύνολα V και E ενός γραφήματος G, να τα συμβολίζουμε V(G) και E(G) αντίστοιχα.)



Σχημα 1.1: Ένα γράφημα.

Στο εξής, θα συμβολίζουμε n το πλήθος των κορυφών ενός γραφήματος και m το πλήθος των ακμών του, εκτός αν αναφέρεται διαφορετικά.

1.1.1.1 Μη κατευθυνόμενα Χρονικά Γραφήματα

Ο ορισμός του γραφήματος είναι η βάση μας για να ορίσουμε τα Χρονικά Γραφήματα (Temporal Graphs).

Χρονικό Γράφημα (Temporal Graph): *Διατεταγμένη τριάδα G={V,E,L}, όπου V ένα μη κενό πεπερασμένο σύνολο που καλούμε σύνολο κορυφών, E σύνολο που καλείται σύνολο ακμών, του οποίου τα στοιχεία είναι δισύνολα των στοιχείων του V και L={*$L_e, e \in E$*}= {*$L_{e_1}, L_{e_2}, \ldots, L_{e_m}$*} σύνολο συνόλων* $L_{e_i},\ i = 1, \ldots, m$, *όπου* $L_{e_i},\ i = 1, \ldots, m$ *σύνολο θετικών ακεραίων τιμών που αντιστοιχίζονται στην ακμή* $e_i \in E$. *Οι τιμές που αντιστοιχίζονται σε κάθε ακμή του γραφήματος καλούνται χρονικές ετικέτες (time labels) ή απλώς ετικέτες (labels) της ακμής και δηλώνουν τις χρονικές στιγμές, στις οποίες η ακμή είναι διαθέσιμη (ή ισοδύναμα τις χρονικές στιγμές, στις οποίες μπορούμε να μεταβούμε από το ένα άκρο της στο άλλο). Το σύνολο των ετικετών που αντιστοιχίζονται στην τυχαία ακμή του χρονικού γραφήματος καλείται σύνολο ετικετών της ακμής.*

Έτσι, μπορούμε πλέον να μιλάμε για χρονικές ακμές (time edges) που λογίζονται ως τριάδες (u,v,l), όπου u,v άκρα ακμής του χρονικού γραφήματος και $l \in L_{\{u,v\}}$ μια χρονική ετικέτα της ακμής αυτής. Δηλαδή, αν μια ακμή e={u,v} έχει περισσότερες από μία χρονικές ετικέτες, λ.χ. έχει σύνολο χρονικών ετικετών $L_e = \{l_1, l_2, l_3\}$, τότε στην ακμή αυτή αντιστοιχούν τρεις χρονικές ακμές, οι (u, v, l_1), (u, v, l_2) και (u, v, l_3).

Ένας άλλος συμβολισμός ενός χρονικού γραφήματος προκύπτει αν θεωρήσουμε γνωστό το γράφημα $G = \{V, E\}$, στις ακμές του οποίου ανατίθενται χρονικές ετικέτες, καθώς και την ανάθεση $L=\{L_e, e \in E(G)\}$ των ετικετών στις ακμές του G. Τότε, το χρονικό γράφημα που προκύπτει συμβολίζεται με $G' = (G, L)$ ή ακόμη με $G' = G(L)$.

Ένα χρονικό γράφημα G με $max_{e \in E(G)}\{|L_e|\} = k \in \mathbb{N}$ καλείται k-επισημασμένο (k-labeled). Στην ειδική περίπτωση, όπου k=1 λέμε πως το G είναι μονοεπισημασμένο (single-labeled). Όταν αναφερόμαστε σε χρονικά γραφήματα που έχουν περισσότερες από μία ετικέτες, χωρίς να θέλουμε να συγκεκριμενοποιήσουμε το πλήθος τους, τα καλούμε εν γένει πολυεπισημασμένα (multi-labeled).

Κωδικοποίηση χρονικών γραφημάτων Θα μελετήσουμε διαφορετικούς τρόπους κωδικοποίησης χρονικών γραφημάτων στον υπολογιστή [1]. Ας θεωρήσουμε χρονικό

γράφημα με σύνολο κορυφών $V = \{u_1, u_2, \ldots, u_n\}$. Δύο βασικοί τρόποι κωδικοποίησης ξεχωρίζουν:

- Κωδικοποίηση διανύσματος: ο τρόπος αυτός περιλαμβάνει ένα 3-διάστατο διάνυσμα $A = (a_{ijt})$, $n^2 \times l$ στοιχείων, όπου $a_{ijt} = 1$ αν η τιμή t είναι χρονική ετικέτα της ακμής (u_i, u_j) και $a_{ijt} = 0$ διαφορετικά.
- Κωδικοποίηση λίστας: διακρίνουμε τρεις τύπους κωδικοποίησης λίστας· *κορυφές-χρόνοι-κορυφές (ΚΧΚ), κορυφές-κορυφές-χρόνοι (ΚΚΧ) και χρόνοι-κορυφές-κορυφές(ΧΚΚ).* Στην κωδικοποίηση ΚΧΚ έχουμε ένα διάνυσμα Adj από n λίστες, μία για κάθε κορυφή του V. Το $Adj[u_i]$ περιέχει μια λίστα για κάθε χρονική στιγμή t από την οποία μπορώ να ξεκινήσω ταξίδι από την u_i (και να διασχίσω συντρέχουσα ακμή[1]), και αυτή η λίστα περιέχει τους γείτονες της u_i τη χρονική στιγμή t[2]. Στους άλλους δύο τύπους, η κωδικοποίηση ακολουθεί την ίδια ιδέα.

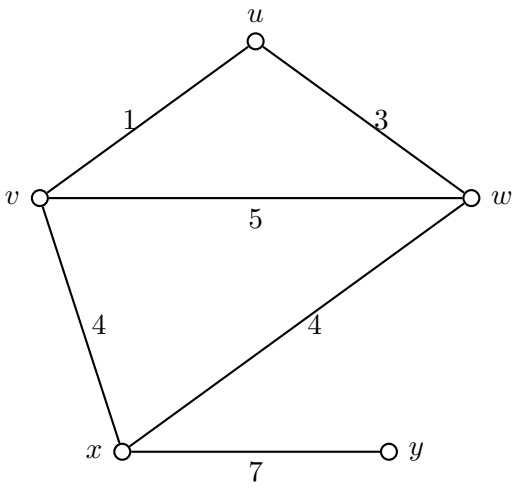
Ταξίδια και άλλες νέες έννοιες Στην πράξη, η ανάθεση χρονικών τιμών ως ετικετών μιας ακμής σημαίνει τη διαθεσιμότητα της ακμής αυτής στις αντίστοιχες χρονικές στιγμές. Πριν, όμως, εξηγήσουμε αναλυτικότερα την έννοια της διαθεσιμότητας τυχαίας ακμής χρονικού γραφήματος, πρέπει να ορίσουμε το ταξίδι (journey) σε ένα χρονικό γράφημα.

*Ένα **ταξίδι (journey)** j από μια κορυφή u σε μια κορυφή v ((u,v)-ταξίδι ή (u,v)-journey) είναι μια ακολουθία χρονικών ακμών $(u, u_1, l_1), (u_1, u_2, l_2), \ldots, (u_{k-1}, v, l_k)$*τέτοια, ώστε $l_i < l_{i+1}$ για κάθε $1 \leq i \leq k-1$. Με άλλα λόγια είναι ένα (u,v)-μονοπάτι, στις ακμές του οποίου συναντάμε αυστηρά αύξουσα σειρά χρονικών ετικετών. Καλούμε το l_1 ***στιγμή ή χρόνο αναχώρησης*** *(departure time)* και το l_k ***στιγμή ή χρόνο άφιξης*** *(arrival time)* και συμβολίζουμε d(j) και a(j) αντίστοιχα.

Έτσι, στο Σχήμα 1.2, υπάρχει (w,y)-ταξίδι j_1, μέσω της x, αλλά δεν υπάρχει (y,v)-ταξίδι.
Πράγματι, η ακμή {w,x} είναι ***διαθέσιμη*** τη χρονική στιγμή 4. Αυτό σημαίνει ότι μπορώ στιγμιαία να τη διασχίσω τη στιγμή $t_1 = 4$ και να μεταβώ από την κορυφή w στην κορυφή x ή το αντίστροφο. Συνεπώς, στο ταξίδι j_1 διασχίζουμε την {w,x} τη στιγμή $t_1 = 4$ και περιμένουμε να έρθει η στιγμή $t_2 = 7$ για να διασχίσουμε στη συνέχεια την ακμή {x,y}. Επομένως, φθάνουμε στην κορυφή y τη χρονική στιγμή $t_2 = 7$.

[1] στην πραγματικότητα, περιέχει μια λίστα για κάθε διαφορετική ετικέτα t που συναντώ στις διάφορες συντρέχουσες στη u_i ακμές

[2] τα άλλα ακρά των ακμών που συντρέχουν στη u_i τη χρονική στιγμή t



Σχημα 1.2: Παράδειγμα μη κατευθυνόμενου χρονικού γραφήματος, στις ακμές του οποίου δίδονται τιμές από το σύνολο $L = \{1, 2, \ldots, 7\}$.

Παρατηρήστε πως τα χρονικά γραφήματα, όπως ορίστηκαν πιο πάνω, είναι μη κατευθυνόμενα γραφήματα και αν τα αντιμετωπίζαμε απλώς ως σημασμένα (labeled) γραφήματα δε θα χρειαζόταν να λάβουμε υπ' όψιν μας οποιαδήποτε κατεύθυνση. Παρ' όλα αυτά, στα διάφορα ταξίδια που συναντάμε σε οποιοδήποτε χρονικό γράφημα υπονοείται κατεύθυνση. Άλλωστε, δείτε στο Σχήμα 1.2 πώς ενώ υπάρχει (w,y)-ταξίδι, δεν υπάρχει (y,w)-ταξίδι!

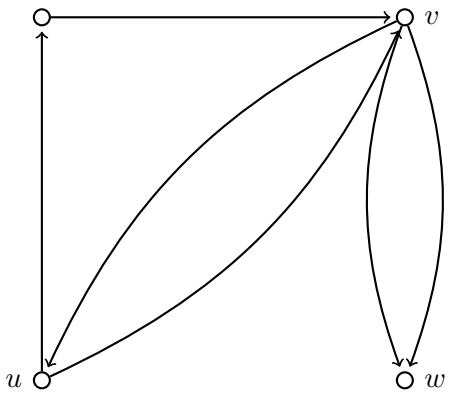
1.1.2 Κατευθυνόμενα Γραφήματα

Καλούμε ***Κατευθυνόμενο Γράφημα*** *ένα δισύνολο G=(V,E), όπου V ένα μη κενό πεπερασμένο σύνολο που καλούμε σύνολο κορυφών και E σύνολο που καλείται σύνολο ακμών και του οποίου τα στοιχεία είναι διατεταγμένα ζεύγη των στοιχείων του V.*
(Όπως και στην περίπτωση των μη κατευθυνόμενων γραφημάτων, έτσι και εδώ συνηθίζεται όταν αναφερόμαστε στα σύνολα V και E ενός γραφήματος G, να τα συμβολίζουμε V(G) και E(G) αντίστοιχα.)

Είναι προφανές ότι στην περίπτωση των κατευθυνόμενων γραφημάτων, αν u,v είναι κορυφές, τότε η ακμή {u,v} είναι διαφορετική από την ακμή {v,u} (βλ. Σχήμα 1.3).

Αξίζει, επίσης, να σημειωθεί πως συχνά συμβαίνει να έχουμε παράλληλες ακμές[3] στα κατευθυνόμενα γραφήματα, τα οποία τότε καλούνται "*πολυγραφήματα*". Στο

[3]Παράλληλες ακμές: ακμές κατευθυνόμενου γραφήματος, οι οποίες έχουν κοινή αρχή και κοινό πέρας.



Σχημα 1.3: Ένα κατευθυνόμενο γράφημα.

παράδειγμα του Σχήματος 1.3, έχουμε δύο παράλληλες ακμές από τη v στη w. Το διατεταγμένο ζεύγος {v,w} θα γραφεί στο σύνολο E των ακμών του γραφήματος δύο φορές: $E = \Big\{ \ldots, \{v, w\}, \ldots, \{v, w\}, \ldots \Big\}$

Πρέπει, λοιπόν, να αναφέρουμε πως καταχρηστικά χρησιμοποιούμε τον όρο 'σύνολο' για το E, αφού ο ορισμός του συνόλου θέλει τα στοιχεία του να είναι διακεκριμένα[4].

1.1.2.1 Κατευθυνόμενα Χρονικά Γραφήματα

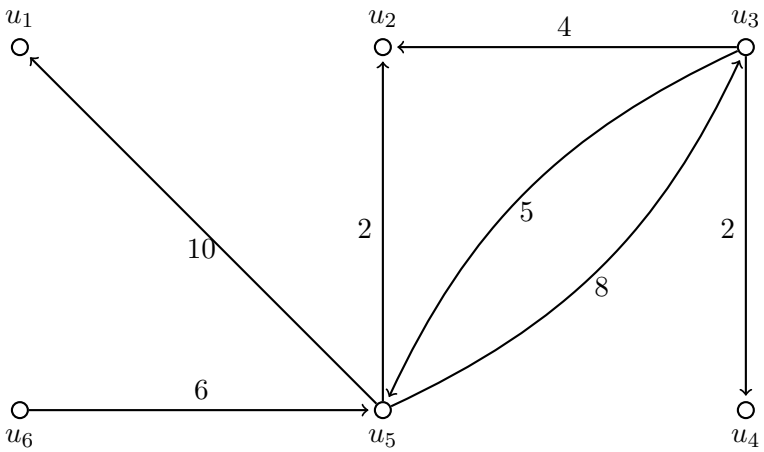
Όπως και στα μη κατευθυνόμενα γραφήματα, έτσι και στα κατευθυνόμενα αναθέτουμε σε κάθε ακμή μια χρονική ετικέτα από ένα σύνολο l διακριτών χρονικών στιγμών:

Κατευθυνόμενο Χρονικό Γράφημα (Directed Temporal Graph): *Διατεταγμένη τριάδα G=(V,E,L), όπου V ένα μη κενό πεπερασμένο σύνολο που καλούμε σύνολο κορυφών, E σύνολο που καλείται σύνολο ακμών, του οποίου τα στοιχεία είναι διατεταγμένα ζεύγη στοιχείων του V και L=$\{L_e, e \in E\}$= $\{L_{e_1}, L_{e_2}, \ldots, L_{e_m}\}$ σύνολο συνόλων $L_{e_i},\ i = 1, \ldots, m$, όπου $L_{e_i},\ i = 1, \ldots, m$ σύνολο θετικών ακεραίων τιμών που αντιστοιχίζονται στην ακμή $e_i \in E$.*
Οι τιμές που αντιστοιχίζονται σε κάθε ακμή του γραφήματος καλούνται χρονικές ετικέτες (time labels) ή απλώς ετικέτες (labels) της ακμής.

Όπως και στην περίπτωση των μη κατευθυνόμενων γραφημάτων, έτσι και εδώ, αν έχουμε το γράφημα $G = \{V, E\}$, στις ακμές του οποίου ανατίθενται χρονικές ετικέτες, καθώς και την ανάθεση L=$\{L_e, e \in E(G)\}$ των ετικετών στις ακμές του G, τότε

[4]In mathematics, a set is a collection of well defined and distinct objects, considered as an object in its own right.

το κατευθυνόμενο χρονικό γράφημα που προκύπτει μπορεί να συμβολιστεί επίσης με $G' = (G, L)$ ή ακόμη με $G' = G(L)$.



Σχημα 1.4: Ένα κατευθυνόμενο χρονικό γράφημα.

Οι έννοιες της *διαθεσιμότητας μιας ακμής*, του *ταξιδίου*, της *στιγμής αναχώρησης* και της *στιγμής άφιξης* (βλ. 1.1.1.1, σελ. 3) επεκτείνονται και στα κατευθυνόμενα χρονικά γραφήματα, λαμβάνοντας πάντα υπ' όψιν τις κατευθύνσεις των ακμών όπως αυτές ορίζονται στο σύνολο ακμών του εκάστοτε κατευθυνόμενου χρονικού γραφήματος.

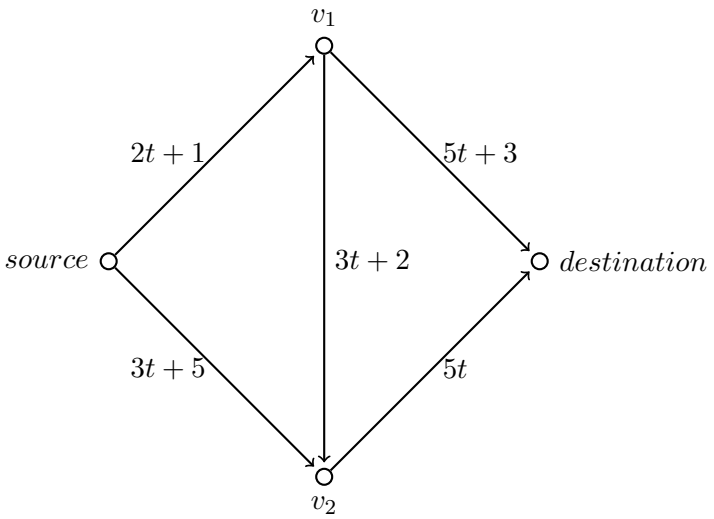
Έτσι, στο Σχήμα 1.4 δεν υπάρχει (u_2, u_6)-ταξίδι, καθώς δεν υπάρχει κατευθυνόμενο μονοπάτι από τη u_2 προς την u_6.
Μπορεί, βέβαια, να υπάρχει κατευθυνόμενο μονοπάτι από μια κορυφή προς μια άλλη, αλλά η σειρά των χρονικών ετικετών στην ακολουθία των ακμών του μονοπατιού να μην είναι αυστηρώς αύξουσα. Ένα τέτοιο παράδειγμα βλέπουμε στο Σχήμα 1.4, όπου από τη u_6 υπάρχει κατευθυνόμενο μονοπάτι προς τη u_2, αλλά δεν υπάρχει το αντίστοιχο ταξίδι. Αντίθετα, από τη u_3 προς τη u_1 υπάρχει ταξίδι μέσω της u_5, με στιγμή άφιξης $t_1 = 10$.

1.1.3 Περιοδικά χρονικά γραφήματα

Τα περιοδικά χρονικά γραφήματα είναι μια ειδική κατηγορία χρονικών γραφημάτων, στα οποία το σύνολο χρονικών ετικετών της τυχαίας ακμής e του γραφήματος δίνεται από μια πρωτοβάθμια ως προς t (χρόνος) παράσταση $a_e \cdot t + b_e$, όπου $a_e, b_e \in \mathbb{N}$.

Περιοδικό Χρονικό Γράφημα (Periodic Temporal Graph): *Χρονικό γράφημα (κατευθυνόμενο ή μη)* $G=\{V,E,L\}$, *όπου* $L=\{L_e, e \in E\}= \{L_{e_1}, L_{e_2}, \ldots, L_{e_m}\}$, *με* $L_{e_i} = \{a_{e_i}t + b_{e_i},\ t \in \mathbb{N}\},\ i = 1, \ldots, m,\ a_e, b_e \in \mathbb{N}$.

Έτσι, αν μια ακμή παίρνει ετικέτες από την παράσταση $2t + 1$, είναι διαθέσιμη τις χρονικές στιγμές $1, 3, 5, 7, 9, \ldots$, δηλαδή είναι *περιοδικά* διαθέσιμη από τη χρονική στιγμή 1 και μετά, με περίοδο ίση με 2. Παρατηρήστε το Σχήμα 1.5 και ας συζητήσουμε τη διαθεσιμότητα κάθε ακμής του. Για να το κάνουμε λίγο πιο ενδιαφέρον, ας υποθέσουμε ότι ο χρόνος ξεκινά να μετρά τη στιγμή $t_0 = 11$.



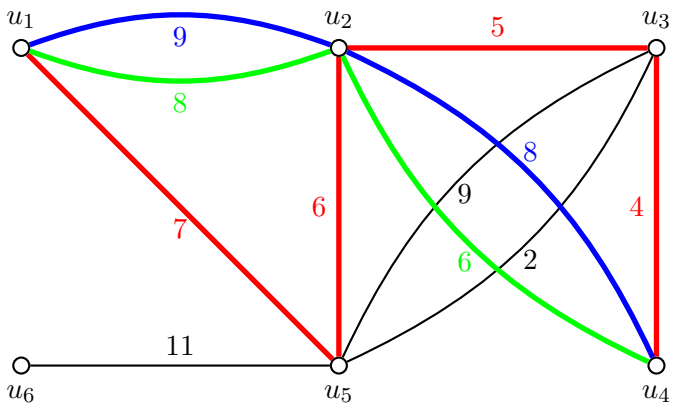
Σχημα 1.5: Παράδειγμα διαθεσιμότητας ακμών σε περιοδικά χρονικά γραφήματα.

- Η ακμή $\{source, v_1\}$ είναι διαθέσιμη τις χρονικές στιγμές $(2 \times 11 + 1 =)23,\ (2 \times 12 + 1 =)25,\ (2 \times 13 + 1 =)27$, κ.ο.κ.
- Η ακμή $\{source, v_2\}$ είναι διαθέσιμη τις χρονικές στιγμές $(3 \times 11 + 5 =)38,\ (3 \times 12 + 5 =)41,\ (3 \times 13 + 5 =)44$, κ.ο.κ.
- Η ακμή $\{v_1, v_2\}$ είναι διαθέσιμη τις χρονικές στιγμές $(3 \times 11 + 2 =)35,\ (3 \times 12 + 2 =)38,\ (3 \times 13 + 2 =)41$, κ.ο.κ.
- Η ακμή $\{v_2, destination\}$ είναι διαθέσιμη τις χρονικές στιγμές $(5 \times 11 =)55,\ (5 \times 12 =)60,\ (5 \times 13 =)65$, κ.ο.κ.
- Η ακμή $\{v_1, destination\}$ είναι διαθέσιμη τις χρονικές στιγμές $(5 \times 11 + 3 =)58,\ (5 \times 12 + 3 =)63,\ (5 \times 13 + 3 =)68$, κ.ο.κ.

Επομένως, μπορώ να περάσω από την *source* στη v_1 τη χρονική στιγμή 23, αλλά δεν θα μπορέσω να συνεχίσω προς την *destination* παρά μόνο όταν έρθει η χρονική στιγμή 58!

1.2 Μια εκτενέστερη ματιά στα ταξίδια

Πρώτιστα Ταξίδια Σε ένα χρονικό γράφημα, ένα (u,v)-ταξίδι j από χρονική στιγμή t καλείται ***πρώτιστο*** *(foremost)* αν η στιγμή αναχώρησής του είναι τουλάχιστον t και η στιγμή άφιξής του είναι η ελάχιστη δυνατή, υπό την ανάθεση L που έχει γίνει στο γράφημα.
Με μαθηματικούς όρους, αν J το σύνολο όλων των (u,v)-ταξιδίων j που έχουν χρόνο αναχώρησης $d(j) \geq t$, ένα ταξίδι $j \in J$ είναι πρώτιστο αν ο χρόνος άφιξής του είναι $a(j) = min_{j' \in J}\{a(j')\}$. Στο εξής θα εννοείται πως ο χρόνος t είναι η στιγμή μηδέν (t=0), εκτός αν αναφέρεται διαφορετικά.



Σχημα 1.6: Μελέτη ταξιδίων σε παράδειγμα μη κατευθυνόμενου χρονικού γραφήματος.

Στο παράδειγμα του Σχήματος 1.6, το πρώτιστο (u_4, u_1)-ταξίδι είναι αυτό που ακολουθεί την κόκκινη γραμμή από τη u_4 προς τη u_1 και η στιγμή άφιξής του είναι $t_1 = 7$.

Τάχιστα Ταξίδια Ένα (u,v)-ταξίδι j καλείται ***τάχιστο*** *(fastest)* (u,v)-ταξίδι αν η διάρκειά του είναι η ελάχιστη δυνατή. Με τον όρο ***διάρκεια*** *(duration)* ταξιδίου εννοούμε την ποσότητα $a(j) - d(j) + 1$.

Έτσι, στο Σχήμα 1.6 μπορεί το πρώτιστο (u_4, u_1)-ταξίδι να είναι αυτό που φαίνεται με κόκκινη γραμμή, όμως το τάχιστο (u_4, u_1)-ταξίδι είναι αυτό που ακολουθεί τη μπλε γραμμή από τη u_4 προς τη u_1, με διάρκεια ταξιδίου ίση με δύο.

Συντομότερα Ταξίδια Ένα (u,v)-ταξίδι j καλείται ***συντομότερο*** *(shortest)* (u,v)-ταξίδι αν έχει το ελάχιστο μήκος από όλα τα (u,v)-ταξίδια. Καλούμε ***μήκος*** του ταξιδίου το πλήθος των ακμών που διασχίζει.

Έτσι, στο Σχήμα 1.6 συναντάμε περισσότερα από ένα συντομότερα (u_4, u_1)-ταξίδια. Ένα από αυτά είναι αυτό που ακολουθεί την πράσινη γραμμή από τη u_4 προς τη u_1 με μήκος 2.
Σημειώστε πως η διάρκεια του τελευταίου είναι ίση με τρία (8-6+1=3), επομένως δεν είναι επιπλέον το τάχιστο (u_4, u_1)-ταξίδι. Αντίθετα, το (u_4, u_1)-ταξίδι που ακολουθεί την μπλε γραμμή είναι συγχρόνως τάχιστο και συντομότερο.

Σημείωση Είναι προφανές πως μπορεί να έχουμε περισσότερα του ενός πρώτιστα, τάχιστα ή συντομότερα ταξίδια από μια κορυφή u σε μια κορυφή v σε ένα χρονικό γράφημα. Όμως, αυτά θα έχουν μεταξύ τους την ίδια στιγμή άφιξης, την ίδια διάρκεια ή το ίδιο μήκος αντίστοιχα.

Κεφάλαιο 2

Συνδετικότητα στα χρονικά γραφήματα

2.1 Χρονικά-συνδετικά χρονικά γραφήματα

Σκοπός αυτής της ενότητας είναι η μελέτη της συνδετικότητας στα χρονικά γραφήματα. Ας θεωρήσουμε (ισχυρά) συνδετικό (κατευθυνόμενο) γράφημα G(V,E) και σύνολο χρονικών ετικετών $L_0 = \{1, 2, \ldots, l\}$. Γεννάται το ερώτημα:
"Μπορούμε να βρούμε αντιστοίχιση χρονικών ετικετών από το σύνολο L_0 στις ακμές του G έτσι, ώστε το χρονικό γράφημα που προκύπτει να είναι (ισχυρά) χρονικά-συνδετικό (κατευθυνόμενο) γράφημα;"
Με τον όρο ***ισχυρά χρονικά-συνδετικό*** *(strongly temporally-connected)* κατευθυνόμενο χρονικό γράφημα ή ***χρονικά-συνδετικό*** *(temporally-connected)* χρονικό γράφημα εννοούμε χρονικό κατευθυνόμενο ή μη αντίστοιχα γράφημα, για κάθε ζεύγος κορυφών u,v του οποίου υπάρχει ταξίδι από τη u προς τη v και από τη v προς τη u.

Πριν συνεχίσουμε, είναι καλό να κάνουμε μερικές παρατηρήσεις:

- Αν το απλό γράφημα G(V,E) είναι μη συνδετικό (ή μη ισχυρά συνδετικό, στην περίπτωση που είναι κατευθυνόμενο) τότε το ερώτημά μας απαντάται αρνητικά. Δηλαδή, αν το G(V,E) είναι λ.χ. κατευθυνόμενο και όχι ισχυρά συνδετικό γράφημα, τότε οποιαδήποτε ανάθεση ετικετών στις ακμές του κι αν θεωρήσουμε, το χρονικό γράφημα που προκύπτει επίσης δεν είναι ισχυρά χρονικά-συνδετικό.
- Πρέπει το μέγιστο στοιχείο l του συνόλου L_0 να είναι τουλάχιστον ίσο με το μήκος του μεγαλύτερου μονοπατιού του απλού γραφήματος G(V,E).

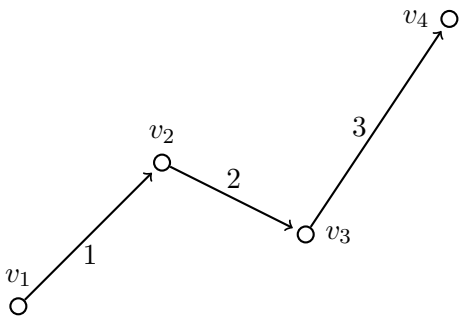
- Προφανώς, αν σε κάθε ακμή δώσουμε όλες τις δυνατές χρονικές στιγμές ως ετικέτες, το ερώτημά μας απαντάται θετικά. Εδώ, όμως, γεννάται ένα νέο ερώτημα. Πόσο "καλή" είναι αυτή η ανάθεση;

Σίγουρα, η ανάθεση όλων των δυνατών ετικετών σε κάθε ακμή είναι μια εύκολη λύση στο πρόβλημά μας. Όμως, αν θέλουμε να *ελαχιστοποιήσουμε τη μέγιστη ετικέτα που χρησιμοποιούμε ή το μέγιστο πλήθος των ετικετών που χρησιμοποιούνται σε κάθε ακμή* ή να συνδυάσουμε αυτά τα δύο, τότε δεν είναι πάντα η καλύτερη.

Στις ακόλουθες παραγράφους θα ορίσουμε και θα μελετήσουμε κάποιες μετρικές που χρησιμοποιούμε στα χρονικά γραφήματα και μας παρέχουν πληροφορία για το πόσο καλή είναι μια ανάθεση ετικετών σε αυτά. Βασικό μας μέλημα θα είναι οι διάφορες αναθέσεις χρονικών ετικετών στις ακμές γραφήματος G να *διατηρούν όλα τα απλά μονοπάτια του G*, δηλαδή αν υπάρχει απλό μονοπάτι από μια κορυφή $u \in V(G)$ σε μια άλλη $v \in V(G)$, να υπάρχει στο αντίστοιχο χρονικό γράφημα το (u, v)-ταξίδι που διασχίζει ακριβώς τις ακμές αυτού του (u, v)-μονοπατιού.

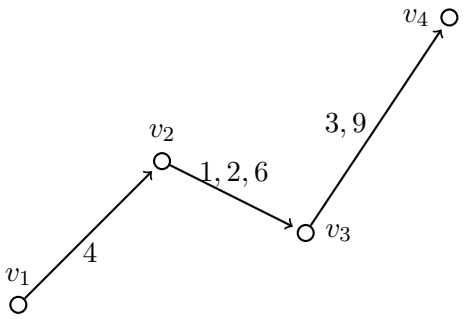
2.1.1 Αριθμός χρονικότητας

Ένα μέτρο που μας παρέχει πληροφορία για το πόσο καλή είναι μια ανάθεση χρονικών ετικετών στις ακμές ενός χρονικού γραφήματος φαίνεται πως είναι το πλήθος των ετικετών που δίνουμε σε κάθε ακμή. Αν, δηλαδή, σε κάθε ακμή ενός χρονικού γραφήματος ανατίθεται μία ακριβώς χρονική ετικέτα με τρόπο που να διατηρούνται όλα τα απλά μονοπάτια του αντίστοιχου απλού γραφήματος, τότε προφανώς έχουμε κάνει μια πολύ καλή ανάθεση. Δείτε το παράδειγμα του Σχήματος 2.1.



Σχημα 2.1: Παράδειγμα καλής ανάθεσης χρονικών ετικετών σε κατευθυνόμενο χρονικό γράφημα.

Για να καταλάβουμε καλύτερα τα παραπάνω, ας δούμε το ίδιο κατευθυνόμενο μονοπάτι, αλλά με μια διαφορετική ανάθεση χρονικών ετικετών.



Σχημα 2.2: Παράδειγμα μιας όχι τόσο καλής ανάθεσης χρονικών ετικετών σε κατευθυνόμενο χρονικό γράφημα.

Σύμφωνα με την ανάθεση ετικετών του Σχήματος 2.2, η ακμή $\{v_1, v_2\}$ είναι διαθέσιμη τη χρονική στιγμή 4, η ακμή $\{v_2, v_3\}$ είναι διαθέσιμη τις χρονικές στιγμές 1,2 και 6 και τέλος η ακμή $\{v_3, v_4\}$ είναι διαθέσιμη τις χρονικές στιγμές 3 και 9. Προφανώς, όλα τα απλά μονοπάτια του αντίστοιχου απλού γραφήματος διατηρούνται, αλλά γιατί να χρησιμοποιούμε περισσότερες χρονικές ετικέτες στις ακμές μας, ενώ μπορούμε να έχουμε το ίδιο αποτέλεσμα με λιγότερες;

Είναι προφανές πως μπορεί περισσότερες από μία αναθέσεις χρονικών ετικετών στις ακμές ενός γραφήματος να διατηρούν όλα τα απλά μονοπάτια του, δίνοντας μας επίσης τον ελάχιστο δυνατό μέγιστο πληθάριθμο συνόλου ετικετών ακμής. Το τελευταίο, όμως, είναι μοναδικός αριθμός για κάθε γράφημα και καλείται αριθμός χρονικότητας.

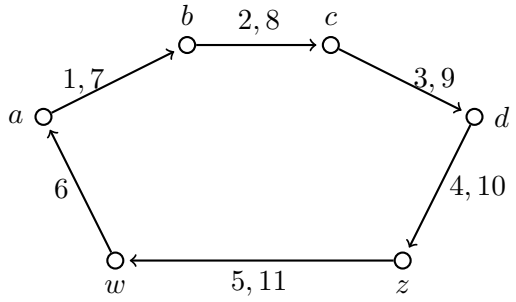
Αριθμός χρονικότητας (Temporality number ή Temporality): *Ο αριθμός χρονικότητας ενός γραφήματος G συμβολίζεται με $\tau(G)$ και δίνεται από τον τύπο*

$$\tau(G) = min\{max\{|L(e)|,\ e \in E(G)\},\ L: \text{ all-paths preserving assignment}\}$$

Μπορούμε, δηλαδή, να τον υπολογίσουμε αν για κάθε ανάθεση χρονικών ετικετών στο G, που διατηρεί όλα τα απλά μονοπάτια του, υπολογίσουμε το μέγιστο πληθάριθμο συνόλου ετικετών ακμής και από αυτούς κρατήσουμε τον ελάχιστο.

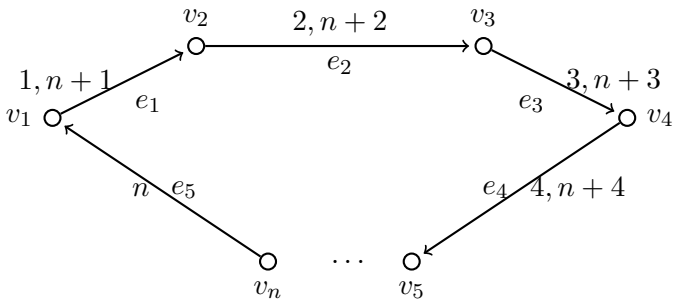
Στο σχήμα 2.3 μπορούμε να δούμε ότι ο αριθμός χρονικότητας του γραφήματος είναι 2 και μάλιστα δίνεται μια ανάθεση χρονικών ετικετών στις ακμές του γραφήματος που διατηρεί όλα τα απλά μονοπάτια του και χρησιμοποιεί το πολύ δύο ετικέτες σε κάθε ακμή.

Για να κατανοήσουμε καλύτερα το λόγο για τον οποίο τα γράφημα του Σχήματος 2.3 έχει αριθμό χρονικότητας ίσο με 2, όπως και κάθε κατευθυνόμενος κύκλος,



Σχημα 2.3: Αριθμός χρονικότητας κατευθυνόμενου κύκλου C, τ(C)=2.

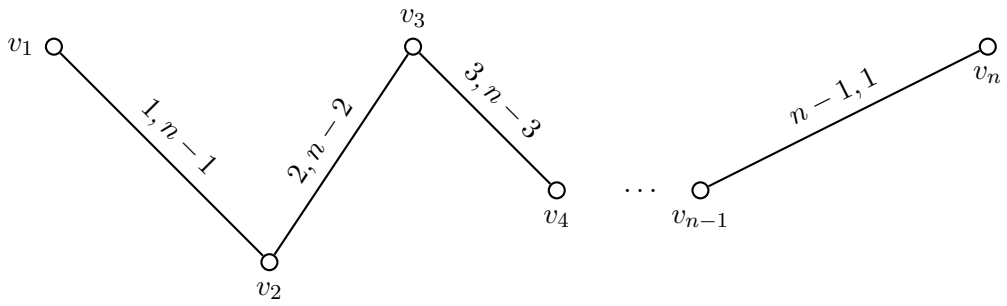
ας σκεφτούμε έναν τυχαίο κατευθυνόμενο κύκλο, C_n, n κορυφών. Ας ξεκινήσουμε από τυχαία κορυφή του και ας προσπαθήσουμε να αναθέσουμε χρονικές ετικέτες σε όλες τις ακμές, με τρόπο που να προκύπτουν ταξίδια από κάθε κορυφή προς κάθε άλλη. Για να γίνει αυτό, πρέπει σε κάθε ακμή που διαγράφουμε να αυξάνει η αντίστοιχη ετικέτα. Σε κάποιο σημείο της διαδικασίας επιστρέφουμε στην αρχική μας ακμή, στην οποία πρέπει να αναθέσουμε ακόμη μία ετικέτα, για να μπορούμε να συνεχίσουμε τα ταξίδια που ξεκίνησαν από προηγούμενη ακμή. Δεν είναι δύσκολο να κατανοήσει κανείς ότι με χρήση δύο ετικετών σε κάθε ακμή ενός οποιουδήποτε κατευθυνόμενου κύκλου (λ.χ. δίνουμε στην ακμή e_i τις ετικέτες i, $n+i$), μπορούμε να πάρουμε όλα τα ταξίδια τα αντίστοιχα των απλών μονοπατιών του (βλ. Σχήμα 2.4).



Σχημα 2.4: Ανάθεση χρονικών ετικετών σε κατευθυνόμενο κύκλο έτσι, ώστε να διατηρούνται όλα τα απλά μονοπάτια και να έχω το έλαχιστο μέγιστο πλήθος ετικετών.

Επομένως, καταλήγουμε στο ότι ο αριθμός χρονικότητας τυχαίου κατευθυνόμενου κύκλου G είναι τ(G)=2 και ο αριθμός χρονικότητας κατευθυνόμενου γραφήματος H που περιέχει κατευθυνόμενο κύκλο είναι τ(H)≥2. Μπορούμε, άραγε, να βρούμε

ένα άνω όριο για τον αριθμό χρονικότητας οποιουδήποτε γραφήματος; Ας θεωρήσουμε το μήκος, έστω p, του μεγαλύτερου μονοπατιού (longest path) του τυχαίου γραφήματος G. Τότε, με μια "επιπόλαιη" ανάθεση των χρονικών ετικετών $1, 2, \ldots, p$ σε κάθε ακμή του γραφήματος, καταφέρνουμε να διατηρήσουμε όλα τα απλά του μονοπάτια στο αντίστοιχο χρονικό γράφημα. Πράγματι, για κάθε μονοπάτι $< e_1, e_2, \ldots, e_k >,\ e_1, e_2, \ldots, e_k \in E(G)$, του γραφήματος μπορούμε να πάρουμε αντίστοιχο ταξίδι χρησιμοποιώντας την αύξουσα ακολουθία των χρονικών ετικετών $1, 2, \ldots, k$ αντίστοιχα, καθώς ισχύει $k \leq p$. Συνεπώς, *ένα άνω όριο για τον αριθμό χρονικότητας τυχαίου γραφήματος είναι το μήκος του μεγαλύτερου μονοπατιού του*. Βέβαια, πολύ συχνά συναντάται ο αριθμός χρονικότητας να είναι μικρότερος από αυτό το άνω όριο. Ένα παράδειγμα φαίνεται στο Σχήμα 2.5, όπου το γράφημα είναι το ίδιο ένα απλό μονοπάτι, μήκους n-1, όπου n το πλήθος των κορυφών του, και ο αριθμός χρονικότητάς του είναι ίσος με 2.



Σχημα 2.5: Καλή ανάθεση χρονικών ετικετών σε γράφημα που είναι το ίδιο ένα απλό μονοπάτι n κορυφών.

Σημειώνεται πως αν το γράφημα του Σχήματος 2.5 ήταν ένα κατευθυνόμενο μονοπάτι, ο αριθμός χρονικότητάς του θα ήταν ίσος με 1.

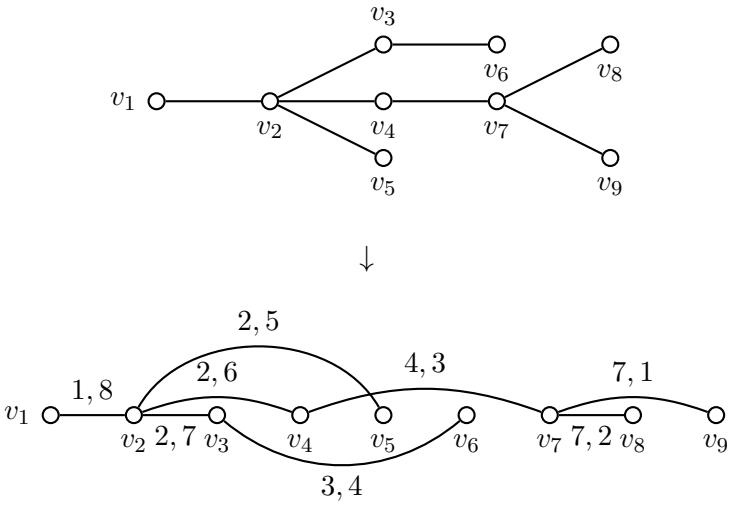
Μερικά θεωρήματα για τον αριθμό χρονικότητας

Θεώρημα. Αν το γράφημα G είναι δένδρο n κορυφών, τότε τ(G)=2.

Απόδειξη. Ας θεωρήσουμε τυχαία κορυφή του G ως ρίζα του. Τότε, μπορούμε να ταξινομήσουμε όλες του τις κορυφές ανάλογα με την απόστασή τους από τη ρίζα αυτή σε μία σειρά $v_1, v_2, \ldots, v_n$. Προφανώς, για κάθε ακμή $\{v_i, v_j\}$ σε αυτή την ταξινόμηση ισχύει $i < j$. Ας δώσουμε σε κάθε ακμή ετικέτες i και $n - (j - 1)$. Έτσι,

η τυχαία κορυφή $v_l \in V(G)$ έχει συντρέχουσες ακμές με ετικέτες:

$$\begin{cases} l' \text{ και } n-(l-1) & \text{όπου } l' < l \\ & \text{ή} \\ l \text{ και } n-(m-1) & \text{όπου } m > l. \end{cases}$$



Σχημα 2.6: Καλή ανάθεση χρονικών ετικετών σε δένδρο n κορυφών.

Οι ακμές της πρώτης κατηγορίας, αν υπάρχουν για τη v_l, είναι αυτές που έχουν κατεύθυνση (στην ταξινόμηση που ορίσαμε) από τη ρίζα προς τη v_l, ενώ αυτές της δεύτερης κατηγορίας, αν υπάρχουν για τη v_l, έχουν κατεύθυνση από τη v_l προς κάποια απόληξη. Τώρα, αρκεί να σκεφτούμε πως κάθε μονοπάτι πάνω στο δένδρο μπορεί να έχει κατεύθυνση από τη ρίζα προς τις απολήξεις του δένδρου ή αντίστροφα. Στην πρώτη περίπτωση, το μονοπάτι διασχίζει μια ακολουθία από (διακριτές) κορυφές πάνω στη σειρά που τις έχουμε ταξινομήσει και έτσι φθάνει σε μια κορυφή v_l μέσω μιας ακμής με ετικέτα $l' < l$ και φεύγει από τη v_l μέσω ακμής με ετικέτα l. Αυτό αποδεικνύει ότι για οποιοδήποτε μονοπάτι με κατεύθυνση από τη ρίζα προς κάποια απόληξη υπάρχει ταξίδι στο αντίστοιχο χρονικό γράφημα που κατασκευάσαμε. Στην περίπτωση που το μονοπάτι έχει κατεύθυνση από κάποια απόληξη προς τη ρίζα, διασχίζει μια ακολουθία από (διακριτές) κορυφές πάνω στην αντίστροφη σειρά από αυτήν που τις έχουμε ταξινομήσει και έτσι φθάνει σε μια κορυφή v_l μέσω μιας ακμής με ετικέτα $n-(m-1) < n-(l-1)$[1] και φεύγει από τη v_l μέσω ακμής με ετικέτα $n-(l-1)$. Αυτό αποδεικνύει ότι για οποιοδήποτε μονοπάτι με κατεύθυνση από κάποια απόληξη προς τη ρίζα υπάρχει ταξίδι στο αντίστοιχο χρονικό γράφημα που κατασκευάσαμε.

[1]Είναι $l < m \Leftrightarrow n-(l-1) > n-(m-1)$.

Καταφέραμε να διατηρήσουμε όλα τα μονοπάτια του τυχαίου δένδρου G με ανάθεση ακριβώς δύο χρονικών ετικετών σε κάθε ακμή του. Άρα, τ(G)=2.

Θεώρημα. Αν το G είναι κατευθυνόμενο ακυκλικό γράφημα n κορυφών, τότε τ(G)=1.

Απόδειξη. Ας θεωρήσουμε μια ταξινόμηση, $v_1, v_2, \ldots, v_n$, των κορυφών του G τέτοια, ώστε αν το G περιέχει μια ακμή $\{v_i, v_j\}$, τότε η κορυφή v_i να βρίσκεται στην ταξινόμηση πριν από την κορυφή v_j. Προφανώς, κάθε ακμή του G σε αυτή την ταξινόμηση έχει τη μορφή $\{v_i, v_j\}$, όπου $i < j$. Δίνουμε σε κάθε ακμή $\{v_i, v_j\}$ την ετικέτα i. Έτσι, κάθε εισερχόμενη ακμή της τυχαίας $v_l \in V(G)$ έχει ετικέτα $l' < l$ και κάθε εξερχόμενη ακμή έχει ετικέτα l. Κάθε απλό μονοπάτι του G διασχίζει μια ακολουθία από (διακριτές) κορυφές πάνω στη σειρά που τις έχουμε ταξινομήσει και έτσι φθάνει σε μια κορυφή v_l μέσω μιας ακμής με ετικέτα $l' < l$ και φεύγει από τη v_l μέσω ακμής με ετικέτα l. Αυτό αποδεικνύει ότι για οποιοδήποτε μονοπάτι του G, υπάρχει ταξίδι στο αντίστοιχο χρονικό γράφημα που κατασκευάσαμε.

Καταφέραμε να διατηρήσουμε όλα τα μονοπάτια του τυχαίου κατευθυνόμενου ακυκλικού γραφήματος G με ανάθεση ακριβώς μιας χρονικής ετικέτας σε κάθε ακμή του. Άρα, τ(G)=1.

Παρατηρήστε ότι αν επιπλέον έπρεπε με τις αναθέσεις χρονικών ετικετών σε τυχαίο κατευθυνόμενο γράφημα να διατηρούμε επίσης τους κύκλους μήκους 2, τότε το τελευταίο θεώρημα θα διατυπωνόταν:

"Το G είναι κατευθυνόμενο ακυκλικό γράφημα n κορυφών ανν τ(G)=1."

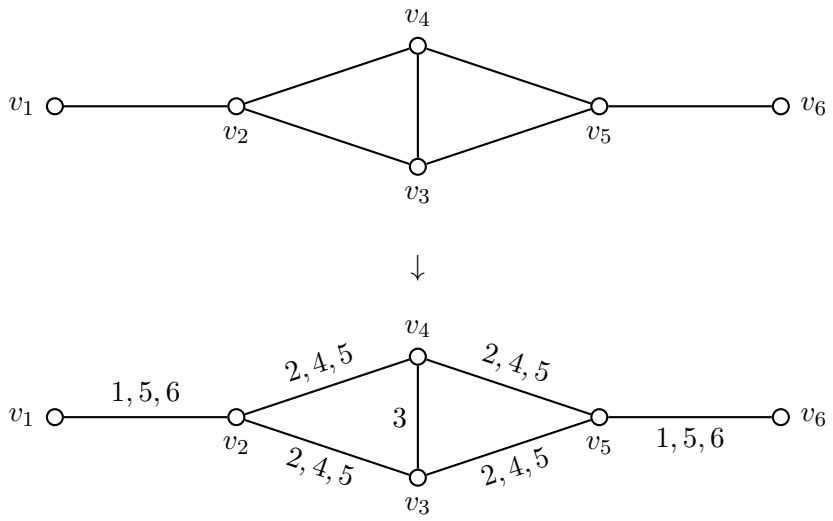
Αυτό συμβαίνει γιατί όλοι οι κατευθυνόμενοι κύκλοι μήκους τουλάχιστον 3 απαιτούν ανάθεση τουλάχιστον δύο χρονικών ετικετών σε τουλάχιστον μία ακμή τους, ώστε να διατηρούνται όλα τα απλά τους μονοπάτια. Αυτό δε συμβαίνει με τους κύκλους μήκους 2 (αφού δεν περιλαμβάνουν κάποιο απλό μονοπάτι πέρα από τις δύο ακμές τους). Αν, λοιπόν, θέλαμε να τους διατηρήσουμε, θα έπρεπε να χρησιμοποιήσουμε μια δεύτερη ετικέτα. Έτσι, η ύπαρξη κύκλου μήκους δύο σε κατευθυνόμενο γράφημα θα έκανε αναγκαία τη χρήση δύο τουλάχιστον ετικετών στο γράφημα, υποχρεώνοντας τον αριθμό χρονικότητάς του να είναι τουλάχιστον 2.

Μερικές παρατηρήσεις για τον αριθμό χρονικότητας

Μέχρι τώρα είδαμε γραφήματα, κατευθυνόμενα και μη, με αριθμούς χρονικότητας το πολύ ίσους με 2. Είναι εύλογο, λοιπόν, κανείς να αναρρωτιέται αν όλα τα γραφήματα

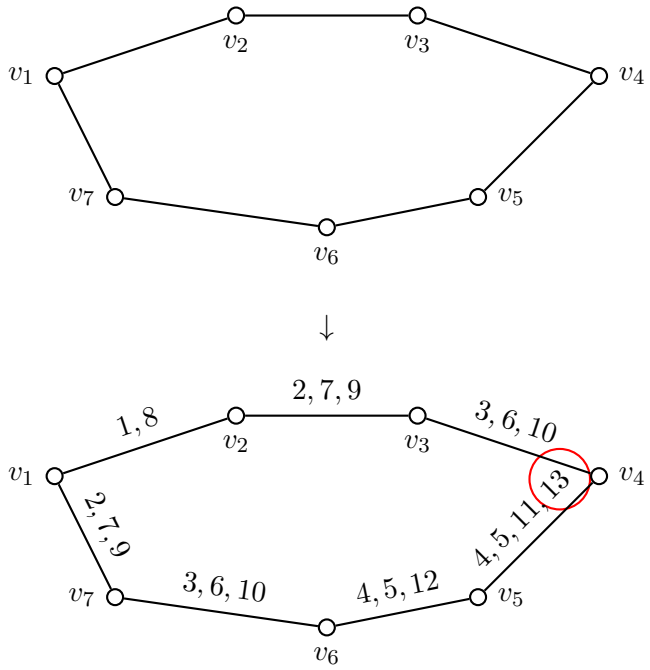
έχουν αριθμό χρονικότητας το πολύ 2. Η απάντηση σε ένα τέτοιο ερώτημα θα ήταν αρνητική. Ακολούθως, δίδονται κάποια σχετικά παραδείγματα.

Στο Σχήμα 2.7, παρατηρούμε ένα μη κατευθυνόμενο γράφημα έξι κορυφών, στις ακμές του οποίου έχουν ανατεθεί ετικέτες έτσι, ώστε να διατηρούνται όλα τα απλά του μονοπάτια. Είναι εύκολο κανείς να καταλάβει πως είναι αναγκαία η ανάθεση τουλάχιστον τριών ετικετών στις ακμές του. Συνεπώς, ο αριθμός χρονικότητας του γραφήματος του Σχήματος 2.7 είναι 3.



Σχημα 2.7: Αριθμός χρονικότητας ίσος με 3 σε μη κατευθυνόμενο χρονικό γράφημα

Ένα άλλο παράδειγμα γραφήματος με αριθμό χρονικότητας μεγαλύτερο από 2 φαίνεται στο Σχήμα 2.8. Το γράφημα είναι ο κύκλος μήκους 7, σε τουλάχιστον μία ακμή του οποίου είναι αναγκαία η ανάθεση τεσσάρων ετικετών. Όπως φαίνεται στο σχήμα, ο αριθμός χρονικότητας του κύκλου μήκους 7 είναι $\tau(C_7)$=4.



Σχημα 2.8: Αριθμός χρονικότητας κύκλου μήκους 7, $\tau(C_7)$=4.

Στον Πίνακα 2.1 αναγράφονται οι αριθμοί χρονικότητας μερικών γνωστών γραφημάτων:

Πινακας 2.1: Αριθμοί χρονικότητας γνωστών γραφημάτων

Graph	Temporality Number
Ακυκλικό κατευθυνόμενο (DAG)	1
Ακυκλικό μη κατευθυνόμενο (AG)	2
Star graph (ειδική κατηγορία AG)	2
Κατευθυνόμενος κύκλος μήκους 2	1
Κατευθυνόμενος κύκλος μήκους ≥ 3	2
Μη κατευθυνόμενος κύκλος μήκους 2	1
Μη κατευθυνόμενος κύκλος μήκους 3	2
Μη κατευθυνόμενος κύκλος μήκους 4, 5 ή 6	3
Μη κατευθυνόμενος κύκλος μήκους ≥ 7	4

2.1.2 Ηλικία

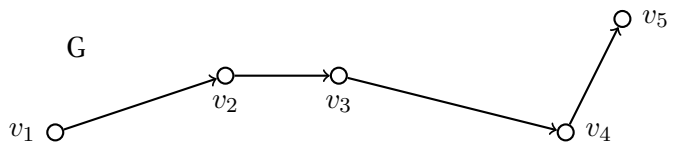
Ας εισάγουμε τώρα μια νέα έννοια που θα μπορούσε επίσης να θεωρηθεί μέτρο που παρέχει πληροφορία για το πόσο καλή είναι μια ανάθεση L.

Ηλικία χρονικού γραφήματος (Age of a temporal graph): *Είναι η μέγιστη ετικέτα που δίνει η ανάθεση που έχει γίνει στο χρονικό γράφημα. Έτσι, αν G είναι ένα γράφημα και* $L = \{L_e, e \in E(G)\}$ *μια ανάθεση χρονικών ετικετών στις ακμές του G, τότε*

$$age(G, L) = max\{maxL_e,\ e \in E(G)\}$$

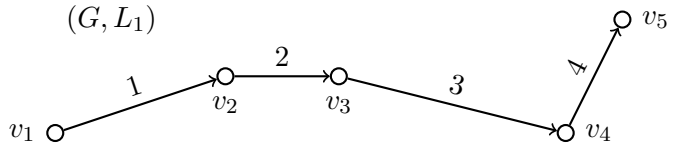
Ας παρατηρήσουμε εδώ μια σημαντική διαφορά της ηλικίας ενός χρονικού γραφήματος από τον αριθμό χρονικότητας. Δεν μπορούμε να μιλήσουμε για την ηλικία ενός γραφήματος, παρά μόνο αν σε αυτό έχει γίνει ανάθεση χρονικών ετικετών. Εξ' ορισμού, η ηλικία ενός γραφήματος αλλάζει ανάλογα με την ανάθεση που έχει γίνει στις ακμές του. Αντίθετα, ο αριθμός χρονικότητας ενός γραφήματος, επειδή υπολογίζεται μελετώντας όλες τις δυνατές αναθέσεις χρονικών ετικετών στις ακμές του, οι οποίες διατηρούν όλα τα απλά μονοπάτια του, είναι μοναδικός για κάθε απλό γράφημα.

Για παράδειγμα, μπορούμε να πούμε ότι ο αριθμός χρονικότητας του γραφήματος G του Σχήματος 2.9 είναι τ(G)=1, αλλά δεν μπορούμε να μιλήσουμε για την ηλικία του, χωρίς πριν να μας έχει δοθεί μια ανάθεση χρονικών ετικετών στις ακμές του.



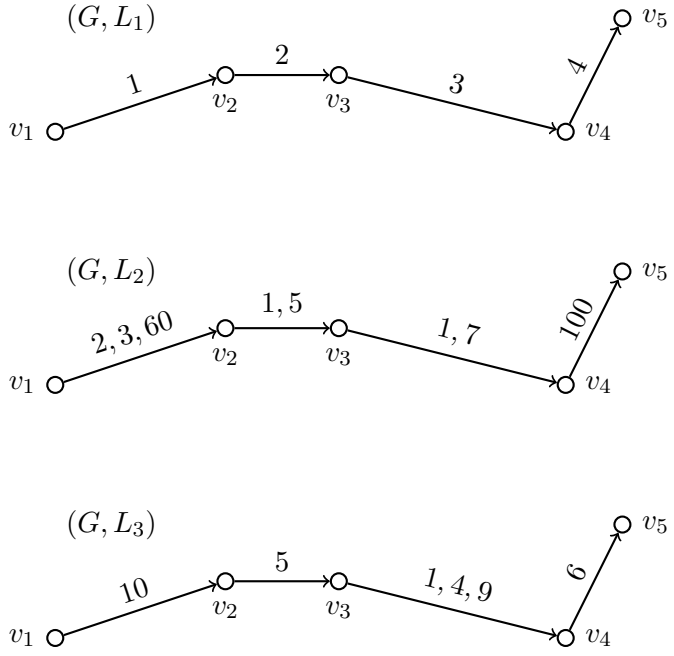
Σχημα 2.9: Ο αριθμός χρονικότητας ενός γραφήματος είναι συγκεκριμένος για κάθε απλό γράφημα

Στο Σχήμα 2.10 φαίνεται επίσης μια ανάθεση ετικετών στις ακμές του G τέτοια, ώστε να διατηρούνται όλα τα απλά μονοπάτια του G και σε κάθε ακμή να έχει ανατεθεί μία μόνο ετικέτα.



Σχημα 2.10: Παράδειγμα ανάθεσης μοναδικής ετικέτας σε κάθε ακμή έτσι, ώστε να διατηρούντα όλα τα απλά μονοπάτια του G

Ας αναθέσουμε, λοιπόν, στις ακμές του γραφήματος G του σχήματος 2.9 χρονικές ετικέτες με διαφόρους τρόπους, όπως φαίνονται στο σχήμα 2.11 παρακάτω.

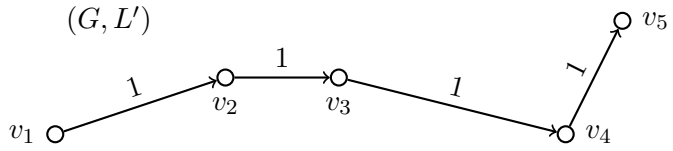


Σχημα 2.11: Η ηλικία ενός χρονικού γραφήματος εξαρτάται από τις ετικέτες που έχουν ανατεθεί στις ακμές του.

Η ηλικία του χρονικού γραφήματος (G, L_1) είναι ίση με $max\{1, 2, 3, 4\} = 4$. Η ηλικία του χρονικού γραφήματος (G, L_2) είναι ίση με $max\{max\{2, 3, 60\}, max\{1, 5\}, max\{1, 7\}, max\{100\}\} = max\{60, 5, 7, 100\} = 100$. Τέλος, η ηλικία του χρονικού γραφήματος (G, L_3) είναι ίση με $max\{10, 5, 9, 6\} = 10$. Ας παρατηρήσουμε εδώ ότι στο χρονικό γράφημα (G, L_3) δε διατηρούνται όλα τα απλά μονοπάτια του G. Παρ'όλα αυτά, η ηλικία υπολογίζεται όπως αναφέρθηκε και πιο πάνω ως η μέγιστη ετικέτα που έχει ανατεθεί στο γράφημα.

Μερικές παρατηρήσεις για την Ηλικία ενός χρονικού γραφήματος

Ας θεωρήσουμε το γράφημα G του σχήματος 2.9 και ας αναθέσουμε σε κάθε ακμή του τη χρονική ετικέτα "ένα" (1). Έστω L' η τελευταία ανάθεση (βλ. Σχήμα 2.12). Θα σκεφτόταν κανείς "Ωραία! Βρήκαμε ένα χρονικό γράφημα που προκύπτει από το G -κατασκευάσαμε, δηλαδή, μια ανάθεση ετικετών στις ακμές του G- με πολύ μικρή ηλικία". Δυστυχώς, όμως, τα πράγματα μάλλον δεν είναι τόσο απλά. Γιατί μπορεί να έχουμε επιτύχει μικρή ηλικία στο χρονικό μας γράφημα, αλλά δεν μπορούμε να "ταξιδέψουμε" παρά μόνο από μία κορυφή του G στην κορυφή εκείνη προς την οποία εξέρχεται ακμή στο G. Μπορώ, δηλαδή, να διασχίσω κάθε ακμή τη χρονική στιγμή "ένα", αλλά δεν μπορώ να συνεχίσω διασχίζοντας κάποια άλλη ακμή μετέπειτα.



Σχημα 2.12: Παρατηρήσεις για την ηλικία ενός χρονικού γραφήματος.

Συχνά, λοιπόν, μας ενδιαφέρει να βρούμε μια ανάθεση χρονικών ετικετών σε ένα γράφημα που να δίνει *μικρή ηλικία*, πληρώντας κάποια επιπλέον συνθήκη.

Για παράδειγμα, μπορεί να θέλουμε να βρούμε μια ανάθεση ετικετών στις ακμές ενός γραφήματος, η οποία διατηρεί όλα τα απλά μονοπάτια του γραφήματος και είναι αυτή που εχει τη μικρότερη ηλικία από όλες τις αναθέσεις που διατηρούν όλα τα απλά μονοπάτια του γραφήματος. Στο γράφημα του σχήματος 2.11, η ανάθεση αυτή θα ήταν η L_1.

Συνηθίζεται η επιπλέον συνθήκη που ζητείται να πληρεί η ανάθεση ετικετών στις ακμές του γραφήματος να σχετίζεται με τη συνδετικότητα μέσα στο χρονικό γράφημα και κατ' επέκταση με τη δυνατότητα που μας δίνεται να ταξιδεύουμε μεταξύ των διαφόρων κορυφών του.

Ας παρομοιάσουμε ένα κατευθυνόμενο χρονικό γράφημα με ένα δίκτυο πόλεων που συνδέονται με μονόδρομους και ας είναι οι χρονικές ετικέτες σε κάθε ακμή {u,v} οι ώρες στις οποίες επιτρέπεται ένα λεωφορείο να ξεκινήσει από την πόλη u για να κατευθυνθεί προς την πόλη v (το παράδειγμα αυτό εύκολα επεκτείνεται και σε μη κατευθυνόμενα γραφήματα). Μια ανάθεση που διατηρεί όλα τα απλά μονοπάτια του απλού γραφήματος θα σήμαινε ότι τα δρομολόγια των λεωφορίων στο δίκτυο πόλεών μας είναι τέτοια, που να μπορούμε να μετακινηθούμε με λεωφορείο από οποιαδήποτε πόλη προς οποιαδήποτε άλλη, εφόσον υπάρχει δρόμος που τις συνδέει.

Συχνά, όμως, δεν είναι οι ενδιάμεσοι σταθμοί των λεφορείων που μας ενδιαφέρουν, αλλά μόνο να φθάσουμε από την αφετηρία στον προορισμό μας, άσχετα από τις ενδιάμεσες στάσεις-σταθμούς. Δηλαδή, δε μας απασχολεί αν θα διατηρούνται όλα τα μονοπάτια του απλού γραφήματος, αλλά εάν θα διατηρείται η λεγόμενη ***προσβασιμότητα*** *(reachability).*

Λέμε ότι μια ανάθεση χρονικών ετικετών στις ακμές ενός γραφήματος G διατηρεί την προσβασιμότητα (reachability) του G, εάν:

$$\forall u, v \in V(G), \exists (u,v) - path\ in\ G \Leftrightarrow \exists (u,v) - journey\ in\ G$$

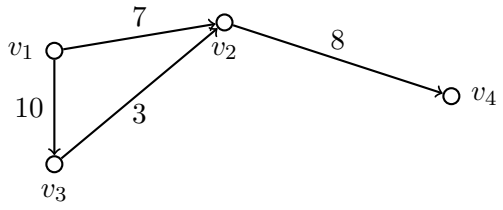
Η συνεπαγωγή $\exists (u,v) - journey\ in\ G \Rightarrow \exists (u,v) - path\ in\ G$ προφανώς ισχύει για κάθε ζεύγος κορυφών $u, v \in V(G)$. Άρα, ο παραπάνω ορισμός της διατήρησης της προσβασιμότητας εναλλακτικά γίνεται:

"*Λέμε ότι μια ανάθεση χρονικών ετικετών στις ακμές ενός γραφήματος G διατηρεί την προσβασιμότητα (reachability) του G, εάν:*

$$\forall u, v \in V(G), \exists (u,v) - path\ in\ G \Rightarrow \exists (u,v) - journey\ in\ G"$$

Αυτό σημαίνει ότι μια ανάθεση χρονικών ετικετών, L, στις ακμές ενός γραφήματος G διατηρεί την προσβασιμότητα του G, εάν για κάθε ζεύγος κορυφών που στο G συνδέονται με μονοπάτι(α), η L διατηρεί τουλάχιστον ένα από αυτά.

Προφανώς, μια ανάθεση που διατηρεί όλα τα μονοπάτια ενός γραφήματος διατηρεί επίσης την προσβασιμότητα του γραφήματος. Το αντίθετο δεν ισχύει, κάτι που επιβεβαιώνει το Σχήμα 2.13.



Σχημα 2.13: Διατήρηση προσβασιμότητας γραφήματος, χωρίς απαραίτητα να διατηρούνται όλα τα μονοπάτια του.

Στο απλό γράφημα του Σχήματος 2.13 συναντάμε δύο (v_1, v_2)-μονοπάτια, ένα (v_1, v_3)-μονοπάτι, δύο (v_1, v_4)-μονοπάτια, ένα (v_2, v_4)-μονοπάτι, ένα (v_3, v_2)-μονοπάτι και ένα (v_3, v_4)-μονοπάτι. Η ανάθεση που έχουμε κάνει στις ακμές του γραφήματος διατηρεί όλα τα μονοπάτια εκτός από τα (v_1, v_3, v_2) και -κατ' επέκταση- (v_1, v_3, v_2, v_4).

Παρ' όλα αυτά, η προσβασιμότητα του γραφήματος διατηρείται, καθώς από κάθε $v_i,\ i = 1, \ldots, 4$ υπάρχει *τουλάχιστον ένα* ταξίδι προς τις κορυφές προς τις οποίες επίσης υπάρχει μονοπάτι στο αρχικό γράφημα.

Στην ακόλουθη ενότητα, θα μελετήσουμε μερικούς αλγορίθμους που απαντούν και επιλύουν ζητήματα που συχνά προκύπτουν στα χρονικά γραφήματα και έχουν να κάνουν με τη συνδετικότητα και τη διατήρηση ιδιοτήτων των απλών γραφημάτων από τα οποία αυτά προκύπτουν.

Κεφάλαιο 3

Αλγόριθμοι στα χρονικά γραφήματα

3.1 Εύρεση πρωτίστου ταξιδίου σε μονοεπισημασμένα μη κατευθυνόμενα χρονικά γραφήματα

Δοθέντος χρονικού γραφήματος $Graph = \{V, E, L\}$, συχνά αναζητάμε ένα πρώτιστο ταξίδι από μια κορυφή (πηγή ή source) σε μία άλλη (προορισμός ή destination) ή ένα πρώτιστο ταξίδι από κάθε κορυφή προς κάθε άλλη. Οι ακόλουθοι αλγόριθμοι μάς επιτρέπουν να βρούμε τέτοια πρώτιστα ταξίδια.

3.1.1 Αλγόριθμος FJSL

Ο αλγόριθμος FJSL (**F**oremost **J**ourney in a **S**ingle-**L**abeled temporal graph) έχει εφαρμογή σε μονοεπισημασμένα μη κατευθυνόμενα γραφήματα και βρίσκει ένα πρώτιστο ταξίδι από μια δεδομένη κορυφή (source) προς μια άλλη δεδομένη κορυφή (destination). Πιο συγκεκριμένα, αν υπάρχει τέτοιο ταξίδι επιστρέφει την ακολουθία των κορυφών που περιλαμβάνει το ταξίδι, ενώ αν τέτοιο ταξίδι δεν υπάρχει, ο αλγόριθμος επιστρέφει την κενή ακολουθία (empty sequence).

Algorithm 1 The foremost journey finding algorithm FJSL

```
1: procedure FOREMOST (SOURCE,DESTINATION)-JOURNEY(Graph, source, destination,
   start_time)
2:     for all vertices v ∈ V(Graph) do                    ▷ Initializations
3:         time_to[v] := infinity;                         ▷ Unknown time function
4:                                                         from source to v
```

Algorithm 1 The foremost journey finding algorithm FJSL (continued)

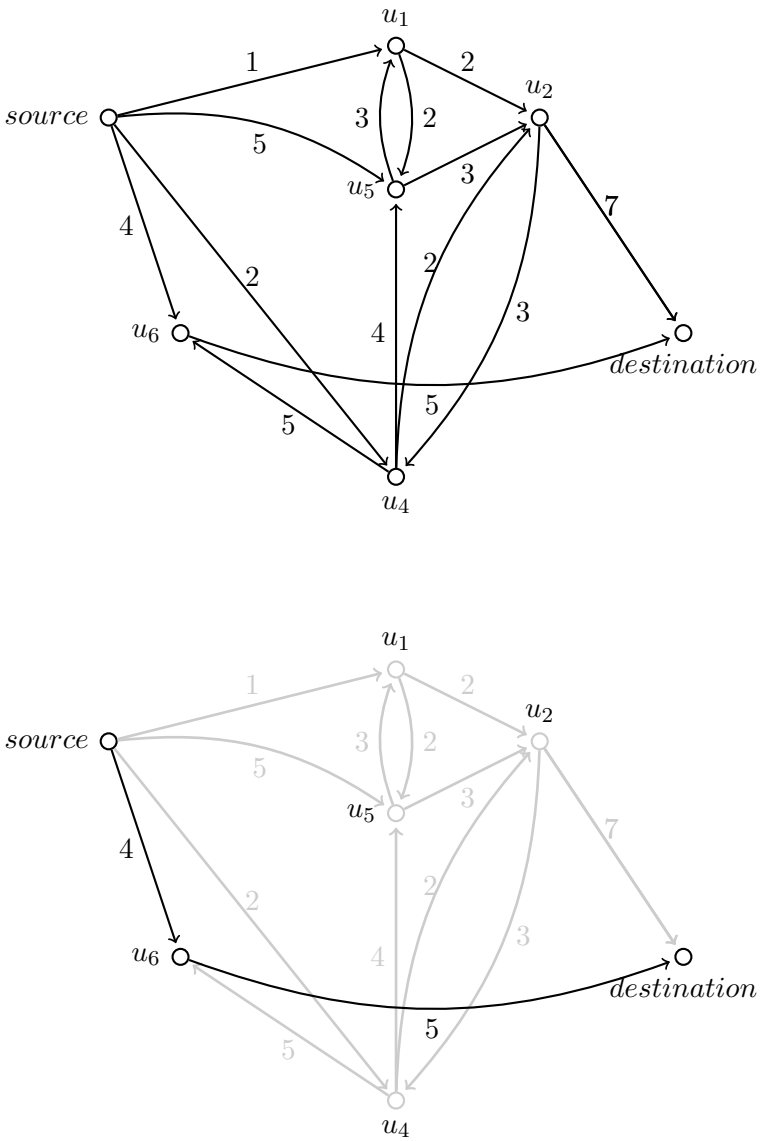
```
5:          previous[v] := undefined;                    ▷ Previous node in optimal
6:                                                         journey from source
7:      end for
8:      time_to[source] := start_time;                   ▷ Time to go from source
9:                                                         to source is the time
10:                                                        we start counting
11:     Q := the set of all nodes in Graph;              ▷ All nodes in Graph are
12:                                                        unoptimized - thus are
13:                                                        in Q
14:     while Q is not empty do                          ▷ The main loop
15:         u := vertex in Q with smallest time in time_to[];  ▷ Start node in first case
16:         remove u from Q;
17:         if u == destination then
18:             t := u;
19:             S := empty sequence;                     ▷ Save the foremost
20:                                                        journey from source
21:                                                        to destination
22:             while previous[t] is defined do
23:                 insert t at the beginning of S;
24:                 t := previous[t];
25:             end while
26:         end if
27:         if time_to[u] == infinity then
28:             Q := empty set;
29:         end if
30:         for all neighbors v of u do
31:             if time_between[u,v] > time_to[u] then
32:                 if time_between[u,v] < time_to[v] then   ▷ Save the current time
33:                                                          for the current node
34:                     time_to[v] := time_between[u,v];
35:                     previous[v] := u;
36:                     decrease-key v in Q;             ▷ Reorder v in the queue Q
37:                 end if
38:             end if
39:         end for
40:     end while
41:     if S is not empty then
42:         write "The foremost journey from", source, "to", destination, "is:", source,
    S;
43:     end if
44:     return time_to[destination];
45: end procedure
```

Εφαρμογή Σημειώνεται πως ο Αλγόριθμος 1 εφαρμόζεται σε κατευθυνόμενα χρονικά γραφήματα χωρίς παραλλαγές στα βήματα, αν θεωρήσουμε ως γείτονες

(neighbors) μιας κορυφής u τις κορυφές

$$v \in V(G) : \exists \{u, v\} \in E(G)$$

Ένα παράδειγμα εκτέλεσης του Αλγορίθμου 1 και μάλιστα σε κατευθυνόμενο χρονικό γράφημα φαίνεται στο Σχήμα 3.1 (αποτέλεσμα εκτέλεσης). Ως start_time λογίζεται η χρονική στιγμή μηδέν (0).



Σχημα 3.1: Παράδειγμα εκτέλεσης αλγορίθμου εύρεσης πρωτίστου ταξιδίου με δεδομένη αρχική και τελική κορυφή.

Απόδειξη ισχύος αλγορίθμου FJSL Θα παρουσιάσουμε μια απόδειξη της ισχύος του παραπάνω αλγορίθμου με επαγωγή στο $|S|$. Για συντομία, θα συμβολίσουμε:

- s τη source
- t τη destination και
- $d(x)$ το χρόνο $time\_to[x]$ που χρειαζόμαστε για να μεταβούμε από την s στην τυχαία κορυφή x του γραφήματος κατά μήκος του πρωτίστου (s, x)-ταξιδίου.

Απόδειξη Όπως αναφέραμε ήδη, η απόδειξη θα γίνει με επαγωγή στο $|S|$. Ας αποδείξουμε ότι ο αλγόριθμος έχει ισχύ (εκτελείται σωστά) στην περίπτωση του $|S| = 1$.

Base Case Στην περίπτωση αυτή, στην πρώτη επανάληψη του βασικού βρόχου του αλγορίθμου το $d(t)$ παίρνει την τελική του τιμή που είναι ο χρόνος $time\_between[s, t]$. Πράγματι, αφού το πρώτιστο ταξίδι περιλαμβάνει $|S| = 1$ ακμή του γραφήματος, για κάθε άλλη (συντεχουσα στην περίπτωση του μη κατευθυνόμενο, εισερχόμενη στην περίπτωση του κατευθυνόμενου γραφήματος) ακμή της t, έστω $\{u, t\}$, ισχύει $time\_to[u] > time\_between[u, t]$ ή $time\_to[t] \leq time\_between[u, t]$. Άρα, σε καμία επανάληψη του βασικού βρόχου δεν επηρεάζεται η τιμή που πήρε το $time\_to[t]$ στην 1^η επανάληψη.

Inductive Hypothesis Υποθέτουμε πως ο αλγόριθμος έχει ισχύ στην περίπτωση του $|S| = k \geq 1$. Δηλαδή, ο αλγόριθμος βρίσκει σωστά πρώτιστα ταξίδια από την s στην t μήκους (πλήθους ακμών) το πολύ k.

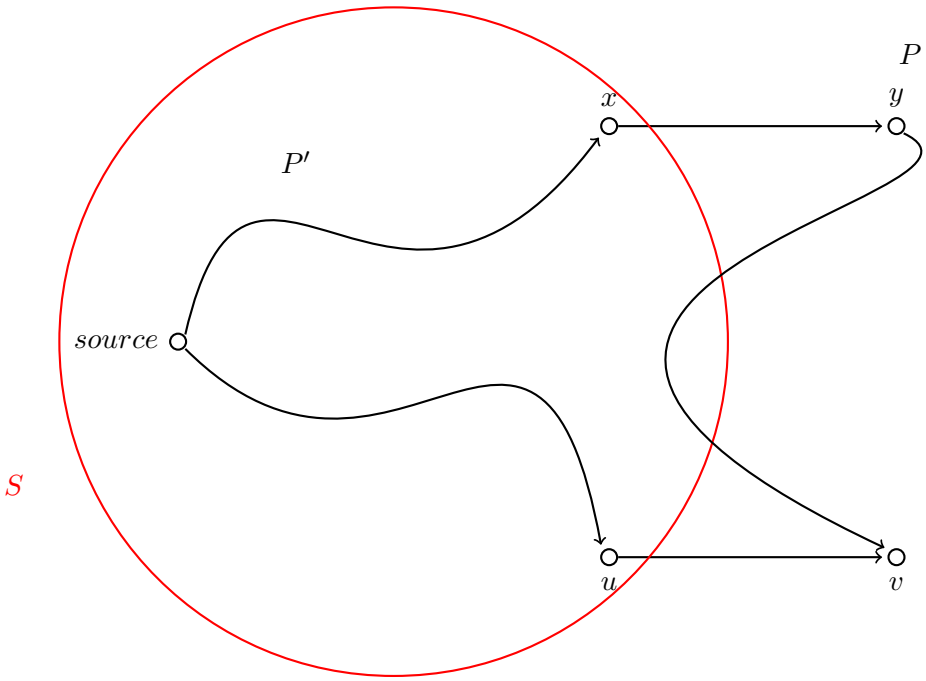
Θα δείξουμε ότι βρίσκει σωστά πρώτιστα ταξίδια μήκους $k + 1$.

Έστω v η επόμενη κορυφή που προστίθεται στο S και έστω $\{u, v\}$ η ακμή που επιλέχθηκε. Το πρώτιστο (s, u)-ταξίδι μαζί με τη $\{u, v\}$ είναι ένα ταξίδι με χρόνο άφιξης $\pi(v)$. Ας θεωρήσουμε τυχαίο (s, v)-ταξίδι, P. Θα δείξουμε ότι δεν έχει χρόνο άφιξης μεγαλύτερο από $\pi(v)$. Έστω $\{x, y\}$ η πρώτη ακμή του P που απομακρύνεται από το S (δηλαδή, η y είναι η πρώτη κορυφή του P που δεν ανήκει στο S) και έστω P' το τμήμα του P από την αρχή του έως την κορυφή x.

Το P έχει ήδη πολύ μεγάλο χρόνο άφιξης από τη στιγμή που αφήνει το S. Πράγματι:

$$l(P) \overbrace{\geq}^{non\ negative\ time\ labels} l(P') + l(x, y) + l(P_{\{y,v\}}) \overbrace{\geq}^{inductive\ hypothesis}$$

$$\underbrace{\geq}_{inductive\ hypothesis} d(x) + l(x, y) \underbrace{\geq}_{definition\ of\ \pi(y)} \pi(y) \geq \pi(v)$$



Σχημα 3.2: Σχήμα Απόδειξης Ισχύος του αλγορίθμου εύρεσης πρωτίστου ταξιδίου σε χρονικά γραφήματα.

Χρόνος εκτέλεσης αλγορίθμου FJSL Ένα άνω όριο του χρόνου εκτέλεσης του αλγορίθμου FJSL σε χρονικό γράφημα με σύνολο κορυφών V και σύνολο ακμών E μπορεί να εκφραστεί ως συνάρτηση των |V| και |E|.

Στη γενική περίπτωση, ο χρόνος εκτέλεσης είναι $O(|E|\cdot dk_Q + |V|\cdot em_Q)$, όπου dk_Q και em_Q οι χρόνοι που απαιτούνται για να εκτελεστεί η εντολή "decrease-key" και για να επιλεγεί η κορυφή του συνόλου Q με την ελάχιστη τιμή στο $time\_to[]$, αντίστοιχα.

Στην απλούστερη εκτέλεση του FJSL αλγορίθμου, οι κορυφές του συνόλου Q αποθηκεύονται σε μια απλά συνδεδεμένη λίστα ή διάνυσμα και η εξαγωγή της κορυφής με την ελάχιστη τιμή στο $time\_to[]$ είναι μια απλή γραμμική αναζήτηση σε όλες τις κορυφές του Q. Σε αυτή την περίπτωση, ο χρόνος εκτέλεσης του αλγορίθμου είναι $O(|E| + |V|^2) = O(|V|^2)$.

3.1.2 Αλγόριθμος FJSL_EXTENDED

Είναι πολύ εύκολο να επεκτείνουμε τον αλγόριθμο FJSL έτσι, ώστε να βρίσκει το πρώτιστο (u, v)-ταξίδι, για κάθε ζεύγος κορυφών $u, v \in V(G)$.

Ο αλγόριθμος FJSL_EXTENDED έχει εφαρμογή σε μονοεπισημασμένα μη κατευθυνόμενα χρονικά γραφήματα, με δυνατότητα επέκτασης σε κατευθυνόμενα χρονικά γραφήματα, αν θεωρήσουμε ως γείτονες (neighbors) μιας κορυφής u τις κορυφές

$$v \in V(G) : \exists\{u, v\} \in E(G)$$

Λαμβάνει ως είσοδο το γράφημα και την αρχική χρονική στιγμή και επιστρέφει τετραγωνικό πίνακα S τάξης n. Το στοιχείο $s_{i,j}$ του S είναι η κενή ακολουθία (empty sequence) όταν δεν υπάρχει (v_i, v_j)-ταξίδι στο γράφημα. Διαφορετικά, είναι η ακολουθία των κορυφών, εκτός της v_i, που περιλαμβάνονται στο πρώτιστο (v_i, v_j)-ταξίδι στο γράφημα, με αντίστροφη σειρά.

Algorithm 2 The foremost journey finding algorithm FJSL_EXTENDED

```
1:  procedure FOREMOST JOURNEY(Graph, start_time)
2:      for all vertices v ∈ V(Graph) do
3:          for all vertices w ∈ V(Graph) do                ▷ Initializations
4:              S[v,w] := empty sequence;
5:          end for
6:      end for
7:      for all vertices w ∈ V(Graph) do
8:          source := w;                                    ▷ Current source
9:          for all vertices z ∈ V(Graph) − {w} do
10:             destination := z;                           ▷ Current destination
11:             for all vertices v ∈ V(Graph) do            ▷ Initializations
12:                 time_to[v] := infinity;                 ▷ Unknown time function
13:                                                           from source to v
14:                 previous[v] := undefined;               ▷ Previous node in optimal
15:                                                           journey from source
16:             end for
17:             time_to[source] := start_time;              ▷ Time to go from source
18:                                                           to source is the time
19:                                                           we start counting
20:             Q := the set of all nodes in Graph;         ▷ All nodes in Graph are
21:                                                           unoptimized - thus are
22:                                                           in Q
23:             while Q is not empty do                     ▷ The main loop
24:                 u := vertex in Q with smallest time in time_to[];   ▷ Start node
25:                                                                      in first case
26:                 remove u from Q;
27:                 if u == destination then
28:                     t := u;
29:                     while previous[t] is defined do
30:                         insert t at the beginning of S[source,destination];
31:                         t := previous[t];
32:                     end while
```

Algorithm 2 The foremost journey finding algorithm FJSL_EXTENDED (continued)

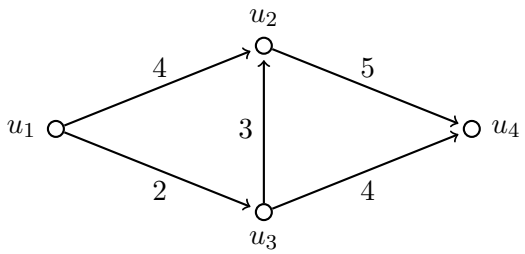
```
33:                 end if
34:                 if time_to[u] == infinity then
35:                     Q := empty set;
36:                 end if
37:                 for all neighbors v of u do
38:                     if time_between[u,v] > time_to[u] then
39:                         if time_between[u,v] < time_to[v] then      ▷ Save the current
40:                                                                                 time
41:                                                                  for the current node
42:                             time_to[v] := time_between[u,v];
43:                             previous[v] := u;
44:                             decrease-key v in Q;                ▷ Reorder v in the queue Q
45:                         end if
46:                     end if
47:                 end for
48:             end while
49:             if S[source,destination] is not empty then
50:                 write "The foremost journey from", source, "to", destination, "is:",
                    source, S[source,destination];
51:             end if
52:             return time_to[destination];
53:         end for
54:     end for
55:     return S;
56: end procedure
```

Όπως είναι φανερό, βασιστήκαμε στον αλγόριθμο FJSL και αφού αρχικοποιήσαμε τον πίνακα S θέσαμε *κάθε κορυφή* ως source *και* ως destination και υπολογίσαμε για κάθε τέτοιο διατεταγμένο ζεύγος (source, destination) το αντίστοιχο πρώτιστο ταξίδι, εάν αυτό υπάρχει. Συνεπώς, ο χρόνος εκτέλεσης του αλγορίθμου FJSL_EXTENDED είναι $O(|V|^4)$.

Εφαρμογή Ένα παράδειγμα εκτέλεσης του Αλγορίθμου 2 στο κατευθυνόμενο χρονικό γράφημα του Σχήματος 3.3 φαίνεται ακολούθως. Ως start_time λογίζεται η χρονική στιγμή μηδέν (0).

Ο αλγόριθμος επιστρέφει τελικά τον ακόλουθο πίνακα:

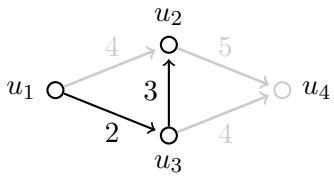
$$S = \begin{bmatrix} \{\} & \{u_3, u_2\} & \{u_3\} & \{u_3, u_4\} \\ \{\} & \{\} & \{\} & \{u_4\} \\ \{\} & \{u_2\} & \{\} & \{u_4\} \\ \{\} & \{\} & \{\} & \{\} \end{bmatrix}.$$



Σχημα 3.3: Παράδειγμα εκτέλεσης αλγορίθμου εύρεσης πρωτίστου ταξιδίου για όλα τα ζέυγη αρχικής και τελικής κορυφή.

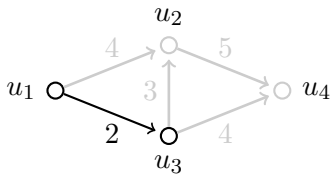
Συνεπώς, όπως μπορούμε εύκολα να δούμε στον παραπάνω πίνακα και να επιβεβαιώσουμε, παρατηρώντας το Σχήμα 3.3:

- Το πρώτιστο (u_1, u_2)-ταξίδι αποτελείται από τις κορυφές u_1, u_3, u_2 με τη σειρά που αναγράφονται (βλ. Σχήμα 3.4)



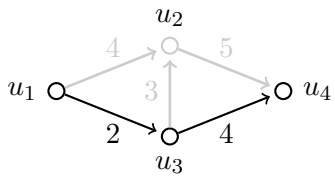
Σχημα 3.4: Αποτέλεσμα FJSL_EXTENDED για το ζεύγος (u_1, u_2).

- Το πρώτιστο (u_1, u_3)-ταξίδι αποτελείται από τις κορυφές u_1, u_3 με τη σειρά που αναγράφονται (βλ. Σχήμα 3.5)



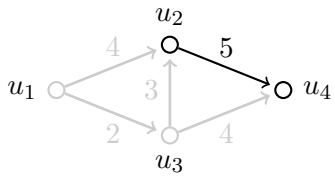
Σχημα 3.5: Αποτέλεσμα FJSL_EXTENDED για το ζεύγος (u_1, u_3).

- Το πρώτιστο (u_1, u_4)-ταξίδι αποτελείται από τις κορυφές u_1, u_3, u_4 με τη σειρά που αναγράφονται (βλ. Σχήμα 3.6)



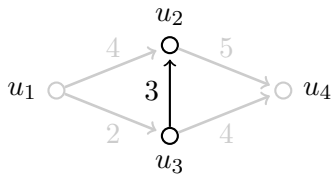
Σχημα 3.6: Αποτέλεσμα FJSL_EXTENDED για το ζεύγος (u_1, u_4).

- Το πρώτιστο (u_2, u_4)-ταξίδι αποτελείται από τις κορυφές u_2, u_4 με τη σειρά που αναγράφονται (βλ. Σχήμα 3.7)



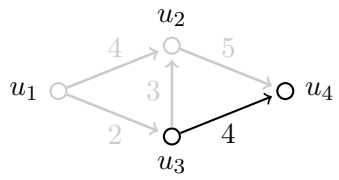
Σχημα 3.7: Αποτέλεσμα FJSL_EXTENDED για το ζεύγος (u_2, u_4).

- Το πρώτιστο (u_3, u_2)-ταξίδι αποτελείται από τις κορυφές u_3, u_2 με τη σειρά που αναγράφονται (βλ. Σχήμα 3.8)



Σχημα 3.8: Αποτέλεσμα FJSL_EXTENDED για το ζεύγος (u_2, u_4).

- Το πρώτιστο (u_3, u_4)-ταξίδι αποτελείται από τις κορυφές u_3, u_4 με τη σειρά που αναγράφονται (βλ. Σχήμα 3.9)



Σχημα 3.9: Αποτέλεσμα FJSL_EXTENDED για το ζεύγος (u_2, u_4).

- Όλα τα υπόλοιπα διατεταγμένα ζεύγη κορυφών δεν έχουν αντίστοιχα πρώτιστα ταξίδια

3.2 Εύρεση πρωτίστου ταξιδίου σε πολυεπισημασμένα χρονικά γραφήματα

Βασιζόμενοι στους αλγορίθμους FJSL και FJSL_EXTENDED, θα κατασκευάσουμε τους αλγορίθμους FJML και FJML_EXTENDED αντιστοίχως, οι οποίοι είναι επεκτάσεις των δύο πρώτων αλγορίθμων σε πολυεπισημασμένα χρονικά γραφήματα, κατευθυνόμενα ή μη. Όπως και πριν, για τα κατευθυνόμενα γραφήματα αρκεί να θεωρήσουμε ως γείτονες (neighbors) μιας κορυφής u τις κορυφές

$$v \in V(G) : \exists\{u, v\} \in E(G)$$

Η βασική ιδέα είναι πως πλέον σε κάθε ακμή αντιστοιχίζεται ένα σύνολο ετικετών L. Έτσι, στον κυρίως βρόχο εξετάζουμε τα στοιχεία του συνόλου L κάθε τρέχουσας ακμής, από τη μικρότερη προς τη μεγαλύτερη ετικέτα, μέχρις ότου βρούμε τη μικρότερη ετικέτα που μπορεί να χρησιμοποιηθεί για να κατασκευάσουμε το ζητούμενο πρώτιστο ταξίδι (αν αυτή υπάρχει).

3.2.1 Αλγόριθμος FJML

Ο αλγόριθμος FJML (**F**oremost **J**ourney in a **M**ulti-**L**abeled temporal graph) βρίσκει ένα πρώτιστο ταξίδι από μια δεδομένη κορυφή (source) προς μια άλλη δεδομένη κορυφή (destination) σε ένα πολυεπισημασμένο χρονικό γράφημα. Πιο συγκεκριμένα, αν υπάρχει τέτοιο ταξίδι επιστρέφει την ακολουθία των κορυφών, εκτός της αρχικής, που περιλαμβάνει το ταξίδι, με αντίστροφη σειρά, ενώ αν τέτοιο ταξίδι δεν υπάρχει, ο αλγόριθμος επιστρέφει την κενή ακολουθία (empty sequence).

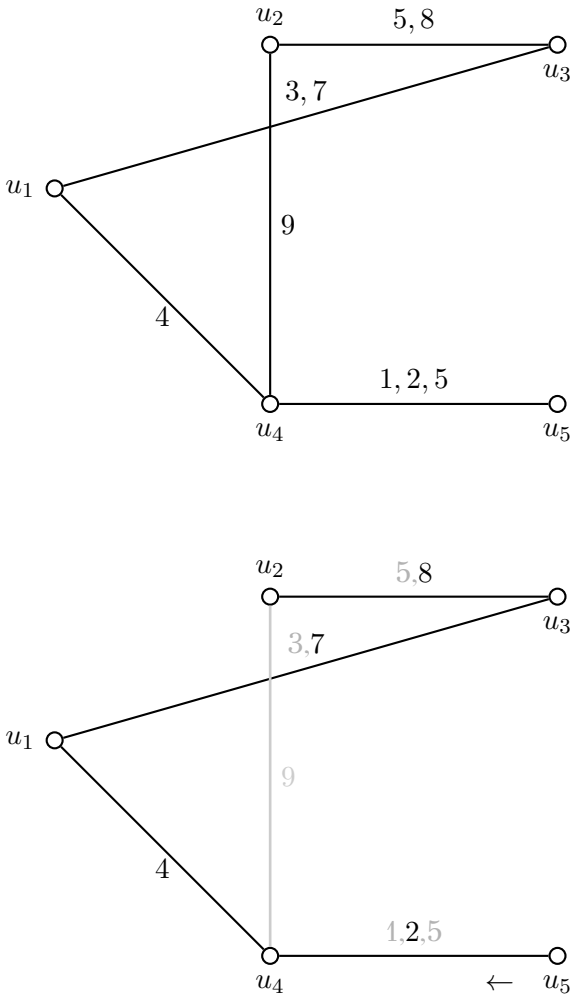
Algorithm 3 The foremost journey finding algorithm FJML

```
1: procedure FOREMOST (SOURCE,DESTINATION)-JOURNEY ML(Graph, source, destination,
   start_time)
2:     for all vertices v ∈ V(Graph) do                          ▷ Initializations
3:         time_to[v] := infinity;                               ▷ Unknown time function
4:                                                                  from source to v
5:         previous[v] := undefined;                             ▷ Previous node in optimal
6:                                                                  journey from source
7:     end for
8:     time_to[source] := start_time;                            ▷ Time to go from source
9:                                                                  to source is the time
10:                                                                 we start counting
11:    Q := the set of all nodes in Graph;                       ▷ All nodes in Graph are
12:                                                                unoptimized - thus are
13:                                                                in Q
14:    while Q is not empty do                                   ▷ The main loop
15:        u := vertex in Q with smallest time in time_to[];     ▷ Start node in first case
16:        remove u from Q;
17:        if u == destination then
18:            t := u;
19:            S := empty sequence;                              ▷ Save the foremost
20:                                                                journey from source
21:                                                                to destination
22:            while previous[t] is defined do
23:                insert t at the beginning of S;
24:                t := previous[t];
25:            end while
26:        end if
27:        if time_to[u] == infinity then
28:            Q := empty set;
29:        end if
30:        for all neighbors v of u do
31:            L := set of all labels of edge {u,v};
32:            while L is not empty do
33:                l := smallest label in L;
34:                remove l from L;
35:                if l > time_to[u] then
36:                    if l < time_to[v] then                    ▷ Save the current time
37:                                                                for the current node
38:                        time_to[v] := l;
39:                        previous[v] := u;
40:                        decrease-key v in Q;                  ▷ Reorder v in the queue Q
41:                        L := empty set;
42:                    end if
43:                end if
44:            end while
45:        end for
46:    end while
```

Algorithm 3 The foremost journey finding algorithm FJML (continued)

```
47:     if S is not empty then
48:         write "The foremost journey from", source, "to", destination, "is:", source,
            S;
49:     end if
50:     return time_to[destination];
51: end procedure
```

Εφαρμογή Ας θεωρήσουμε το γράφημα G του Σχήματος 3.10 και ας εκτελέσουμε τον αλγόριθμο FJML με source την κορυφή v_5, destination την κορυφή v_2 και start_time τη χρονική στιγμή $t_0 = 2$.



Σχημα 3.10: Παράδειγμα εκτέλεσης αλγορίθμου εύρεσης πρωτίστου ταξιδίου με δεδομένη αρχική και τελική κορυφή σε πολυεπισημασμένο μη κατευθυνόμενο χρονικό γράφημα.

Παρατηρούμε πως ο αλγόριθμος επιστρέφει το ταξίδι που δημιουργείται στις ακμές του μονοπατιού $(v_5, v_4, v_1, v_3, v_2)$. Προφανώς, αν η αρχική χρονική στιγμή ήταν $t_0 \geq 3$, τότε ο αλγόριθμος θα επέστρεφε μια κενή ακολουθία $\{\}$. Αν η αρχική χρονική στιγμή ήταν $t_0 = 0$, ο αλγόριθμος θα επέστρεφε το ταξίδι που δημιουργείται στις ακμές του ίδιου μονοπατιου με χρήση, όμως, των χρονικών ακμών $(v_5, v_4, 1)$, $(v_4, v_1, 4)$, $(v_1, v_3, 7)$, $(v_3, v_2, 8)$ με αυτή τη σειρά.

3.2.2 Αλγόριθμος FJML_EXTENDED

Ο αλγόριθμος FJML_EXTENDED έχει εφαρμογή σε πολυεπισημασμένα χρονικά γραφήματα, λαμβάνει ως είσοδο το γράφημα και την αρχική χρονική στιγμή και επιστρέφει τετραγωνικό πίνακα S τάξης n. Το στοιχείο $s_{i,j}$ του S είναι η κενή ακολουθία (empty sequence) όταν δεν υπάρχει (v_i, v_j)-ταξίδι στο γράφημα. Διαφορετικά, είναι η ακολουθία των κορυφών, εκτός της v_i, που περιλαμβάνονται στο πρώτιστο (v_i, v_j)-ταξίδι στο γράφημα, με αντίστροφη σειρά.

Algorithm 4 The foremost journey finding algorithm FJML_EXTENDED

```
1:  procedure FOREMOST JOURNEY ML(Graph, start_time)
2:      for all vertices v ∈ V(Graph) do
3:          for all vertices w ∈ V(Graph) do                ▷ Initializations
4:              S[v,w] := empty sequence;
5:          end for
6:      end for
7:      for all vertices w ∈ V(Graph) do
8:          source := w;                                    ▷ Current source
9:          for all vertices z ∈ V(Graph) − {w} do
10:             destination := z;                           ▷ Current destination
11:             for all vertices v ∈ V(Graph) do            ▷ Initializations
12:                 time_to[v] := infinity;                 ▷ Unknown time function
13:                                                           from source to v
14:                 previous[v] := undefined;               ▷ Previous node in optimal
15:                                                           journey from source
16:             end for
17:             time_to[source] := start_time;              ▷ Time to go from source
18:                                                           to source is the time
19:                                                           we start counting
20:             Q := the set of all nodes in Graph;         ▷ All nodes in Graph are
21:                                                           unoptimized - thus are
22:                                                           in Q
23:             while Q is not empty do                     ▷ The main loop
24:                 u := vertex in Q with smallest time in time_to[];  ▷ Start node
25:                                                           in first case
```

Algorithm 4 The foremost journey finding algorithm FJML_EXTENDED (continued)

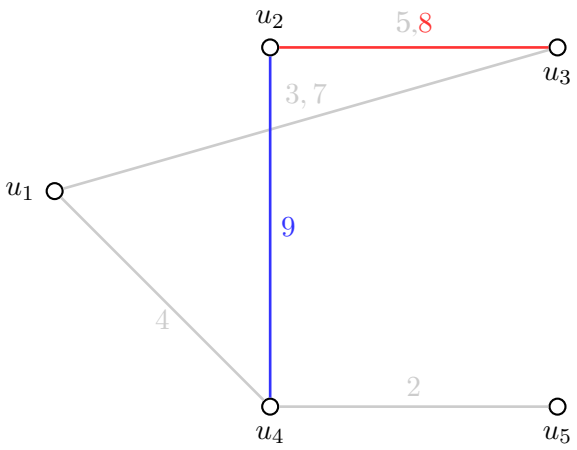
```
26:                 remove u from Q;
27:                 if u == destination then
28:                     t := u;
29:                     while previous[t] is defined do
30:                         insert t at the beginning of S[source,destination];
31:                         t := previous[t];
32:                     end while
33:                 end if
34:                 if time_to[u] == infinity then
35:                     Q := empty set;
36:                 end if
37:                 for all neighbors v of u do
38:                     L := set of all labels of edge {u,v};
39:                     while L is not empty do
40:                         l := smallest label in L;
41:                         remove l from L;
42:                         if l > time_to[u] then
43:                             if l < time_to[v] then          ▷ Save the current time
44:                                                               for the current node
45:                                 time_to[v] := l;
46:                                 previous[v] := u;
47:                                 decrease-key v in Q;        ▷ Reorder v in the queue Q
48:                                 L := empty set;
49:                             end if
50:                         end if
51:                     end while
52:                 end for
53:             end while
54:             if S[source,destination] is not empty then
55:                 write "The foremost journey from", source, "to", destination, "is:",
                    source, S[source,destination];
56:             end if
57:             return time_to[destination];
58:         end for
59:     end for
60:     return S;
61: end procedure
```

Εφαρμογή 1 Ας θεωρήσουμε το γράφημα G του Σχήματος 3.10 και ας εκτελέσουμε τον αλγόριθμο FJML_EXTENDED με start_time τη χρονική στιγμή $t_0 = 8$.

Ο αλγόριθμος προφανώς θα επιστρέψει κενές ακολουθίες για όλα τα διατεταγμένα ζεύγη κορυφών, εκτός από τα:

- (v_2, v_3): χρόνος άφιξης ίσος με 8

- (v_2, v_4): χρόνος άφιξης ίσος με 9
- (v_3, v_2): χρόνος άφιξης ίσος με 8
- (v_3, v_4): χρόνος άφιξης ίσος με 9
- (v_4, v_2): χρόνος άφιξης ίσος με 9

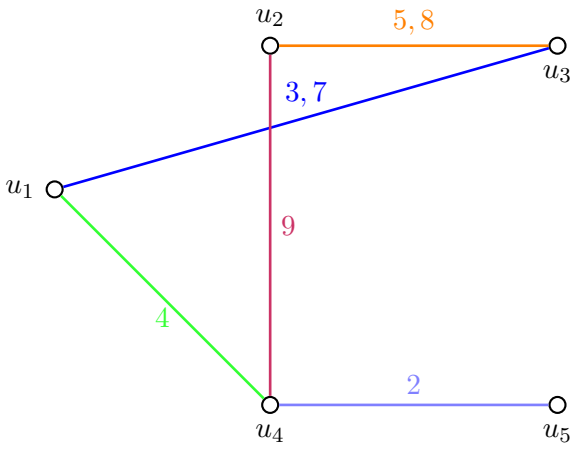


Σχημα 3.11: Παράδειγμα εκτέλεσης αλγορίθμου εύρεσης πρωτίστου ταξιδίου για όλα τα ζεύγη κορυφών σε πολυεπισημασμένο μη κατευθυνόμενο χρονικό γράφημα. (1)

Εφαρμογή 2 Ας εκτελέσουμε ξανά τον αλγόριθμο FJML_EXTENDED για το γράφημα G του Σχήματος 3.10 με *μηδενική start_time*, $t_0 = 0$.

Είναι προφανές πως στην περίπτωση αυτή προκύπτουν περισσότερα ταξίδια στο χρονικό γράφημα, αφού μας δίνεται η δυνατότητα να "ταξιδέψουμε" από την αρχή του χρόνου. Αυτά φαίνονται στον πίνακα που επιστρέφει ο αλγόριθμος, όπως αυτός φαίνεται ακολούθως:

$$S = \begin{bmatrix} \{\} & \{u_3, u_2\} & \{u_3\} & \{u_4\} & \{\} \\ \{u_3, u_1\} & \{\} & \{u_3\} & \{u_4\} & \{\} \\ \{u_1\} & \{u_2\} & \{\} & \{u_1, u_4\} & \{\} \\ \{u_1\} & \{u_1, u_3, u_2\} & \{u_1, u_3\} & \{\} & \{u_5\} \\ \{u_4, u_1\} & \{u_4, u_1, u_3, u_2\} & \{u_4, u_1, u_3\} & \{\} & \{\} \end{bmatrix}.$$

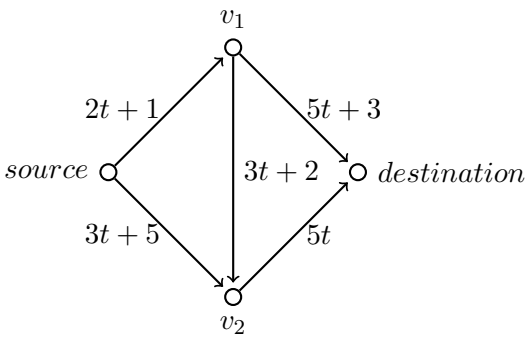


Σχημα 3.12: Παράδειγμα εκτέλεσης αλγορίθμου εύρεσης πρωτίστου ταξιδίου για όλα τα ζεύγη κορυφών σε πολυεπισημασμένο μη κατευθυνόμενο χρονικό γράφημα. (2)

Στις ακόλουθες υποενότητες θα μελετήσουμε αλγορίθμους που βρίσκουν πρώτιστα ταξίδια σε περιοδικά γραφήματα, καθώς οι αλγόριθμοι που είδαμε μέχρι τώρα δεν έχουν εφαρμογή στην περίπτωση των περιοδικών γραφημάτων.

3.3 Εύρεση πρωτίστου ταξιδίου σε περιοδικά χρονικά γραφήματα

Ας θυμηθούμε τι είναι τα περιοδικά χρονικά γραφήματα με τη βοήθεια ενός παραδείγματος (Σχήμα 3.13).



Σχημα 3.13: Παράδειγμα διαθεσιμότητας ακμών σε περιοδικά χρονικά γραφήματα.

Οι αλγόριθμοι που μελετήσαμε πιο πάνω για την εύρεση πρωτίστων μονοπατιών σε μονοεπισημασμένα χρονικά γραφήματα προφανώς δεν έχουν εφαρμογή σε περιοδικά χρονικά γραφήματα. Όμως, ούτε και ο FJML ή ο FJML_EXTENDED, έτσι όπως έχουν γραφεί, μπορούν να βρουν πρώτιστα ταξίδα σε περιοδικά γραφήματα, καθώς στις ακμές των τελευταίων ανατίθενται άπειρες το πλήθος χρονικές ετικέτες. Προκειμένου να λύσουμε αυτό το πρόβλημα, κατασκευάζουμε και χρησιμοποιούμε έναν ελαφρώς αλλαγμένο αλγόριθμο.

3.3.1 Αλγόριθμος FJP

Ο αλγόριθμος FJP (**F**oremost **J**ourney in a **P**eriodic temporal graph) έχει εφαρμογή σε περιοδικά χρονικά γραφήματα και βρίσκει πρώτιστα ταξίδια δεδομένης αρχικής και τελικής κορυφής, με επίσης δεδομένη αρχική χρονική στιγμή.

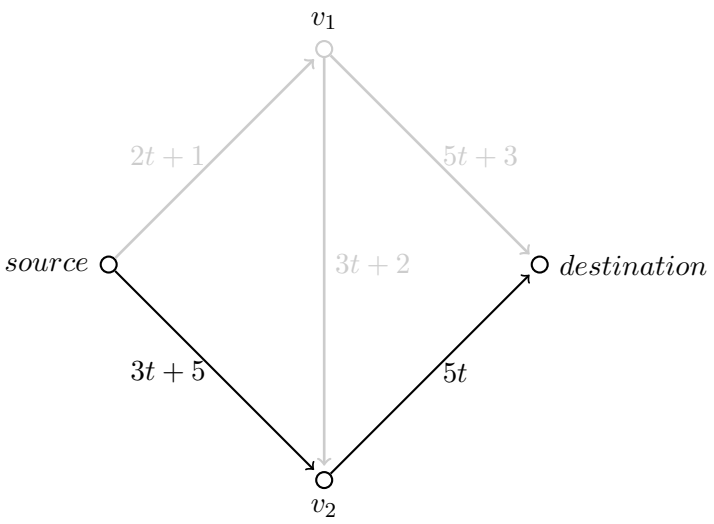
Algorithm 5 The foremost journey finding algorithm - periodic graphs FJP

```
1:  procedure FOREMOST (SOURCE,DESTINATION)-JOURNEY IN PERIODIC GRAPHS(Graph,
    source, destination, start_time)
2:      for all vertices v ∈ V(Graph) do                          ▷ Initializations
3:          time_to[v] := infinity;                               ▷ Unknown time function
4:                                                                    from source to v
5:          previous[v] := undefined;                             ▷ Previous node in optimal
6:                                                                    journey from source
7:      end for
8:      time_to[source] := start_time;                            ▷ Time to go from source
9:                                                                    to source is the time
10:                                                                   we start counting
11:     Q := the set of all nodes in Graph;                       ▷ All nodes in Graph are
12:                                                                   unoptimized - thus are
13:                                                                   in Q
14:     while Q is not empty do                                   ▷ The main loop
15:         u := vertex in Q with smallest time in time_to[];     ▷ Start node in first case
16:         remove u from Q;
17:         if u == destination then
18:             t := u;
19:             S := empty sequence;                              ▷ Save the foremost
20:                                                                   journey from source
21:                                                                   to destination
22:             while previous[t] is defined do
23:                 insert t at the beginning of S;
24:                 t := previous[t];
25:             end while
26:         end if
```

Algorithm 5 The foremost journey finding algorithm - periodic graphs FJP (continued)

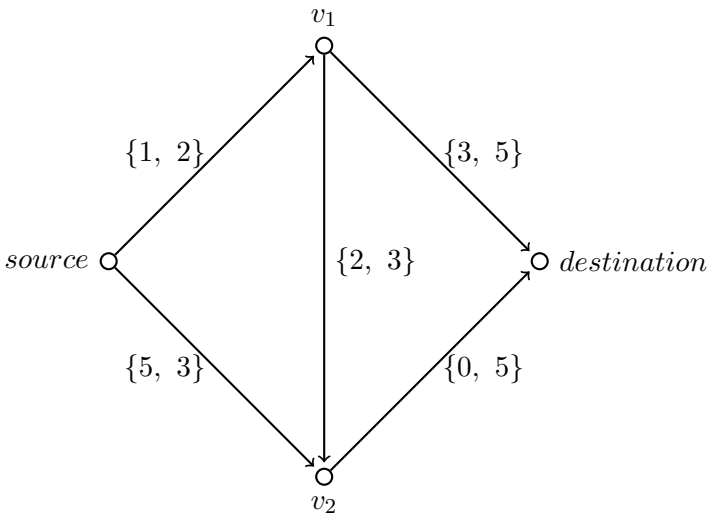
```
27:         for all neighbors v of u do
28:             if time_to[u] not equal to infinity then
29:                 temp_0 := max{0, ⌊(time_to[u] − b_{u,v}) / a_{u,v}⌋ + 1};
30:                 temp := a_{u,v} · temp_0 + b_{u,v};
31:                 if temp < time_to[v] then
32:                     time_to[v] := temp;
33:                     previous[v] := u;
34:                     decrease-key v in Q;                ▷ Reorder v in the queue Q
35:                 end if
36:             end if
37:         end for
38:     end while
39:     write "The foremost journey from", source, "to", destination, "is:", source, S;
40:     return time_to[destination];
41: end procedure
```

Εφαρμογή Ένα παράδειγμα εκτέλεσης του Αλγορίθμου 5 στο κατευθυνόμενο χρονικό γράφημα του Σχήματος 3.13 φαίνεται στο Σχήμα 3.14 (αποτέλεσμα εκτέλεσης). Ως start_time λογίζεται η χρονική στιγμή ένδεκα (11). Έτσι, μπορούμε να διασχίσουμε την ακμή $\{s, v_2\}$ τη στιγμή 38 και την ακμή $\{v_2, destination\}$ τη στιγμή 55, ώστε το ταξίδι να έχει τον ελάχιστο χρόνο άφιξης t=55.



Σχημα 3.14: Παράδειγμα εκτέλεσης αλγορίθμου εύρεσης πρωτίστου ταξιδίου σε περιοδικό γράφημα με δεδομένη αρχική και τελική κορυφή.

Παραλλαγή αλγορίθμου Μπορεί κανείς εύκολα να δει ότι ο ορισμός που δώσαμε για τα περιοδικά χρονικα γραφήματα ισοδυναμεί με αυτόν που αντιστοιχίζει σε κάθε ακμή e του γραφήματος ένα διατεταγμένο ζεύγος ετικετών $\{t\_0_e, period_e\}$, όπου t_0_e η πρώτη χρονική στιγμή στην οποία η e είναι διαθέσιμη και $period_e$ η περίοδος ανά την οποία έχουμε επαναδιαθεσιμότητα της ακμής. Έτσι, το γράφημα του Σχήματος 3.13 θα μπορούσε να παρασταθεί όπως φαίνεται στο Σχήμα 3.15



Σχημα 3.15: Παράδειγμα διαθεσιμότητας ακμών σε περιοδικά χρονικά γραφήματα με χρήση των t_0_e και $period_e$.

Είναι, βέβαια, προφανές πως για κάθε ακμή $e \in E(G)$ ενός περιοδικού γραφήματος ισχύουν:

$$\begin{cases} t\_0_e = b_e \\ period_e = a_e \end{cases}$$

Προκειμένου να εφαρμόζεται ο Αλγόριθμος 5 και σε περιοδικά χρονικά γραφήματα, των οποίων οι ακμές χαρακτηρίζονται από το διατεταγμένο ζεύγος της αρχικής στιγμής διαθεσιμότητας και της περιόδου, αρκεί να αλλάξουμε τον εσωτερικό του βρόχο (βλ. γραμμή 27) και να γίνει:

for all neighbors v of u **do**
  if time_to[u] **not equal to** infinity **then**
    temp$_0$:= $max\{0, \lfloor \frac{time\_to[u]-t\_0_{\{u,v\}}}{period_{\{u,v\}}} + 1 \rfloor\}$;
    temp := $period_{\{u,v\}} \cdot temp_0$ + $t\_0_{\{u,v\}}$;
    if $temp < time\_to[v]$ **then**
      time_to[v] := temp;
      previous[v] := u;
      decrease-key v in Q;
    end if
  end if
end for

3.3.2 Αλγόριθμος FJP_EXTENDED

Όπως και στις προηγούμενες περιπτώσεις χρονικών γραφημάτων, έτσι και εδώ, με μερικές προσθήκες και αλλαγές ο αλγόριθμος FJP μετατρέπεται στον αλγόριθμο FJP_EXTENDED, ο οποίος βρίσκει πρώτιστα ταξίδια για όλα τα διατεταγμένα ζεύγη κορυφών του χρονικού γραφήματος που δέχεται στην είσοδό του.

Ο αλγόριθμος λαμβάνει ως είσοδο το γράφημα και την αρχική χρονική στιγμή και επιστρέφει τετραγωνικό πίνακα S τάξης n. Το στοιχείο $s_{i,j}$ του S είναι η ακολουθία των κορυφών, εκτός της v_i, που περιλαμβάνονται στο πρώτιστο (v_i, v_j)-ταξίδι, με αντίστροφη σειρά.

Τα βήματά του αλγορίθμου περιγράφονται ακολούθως:

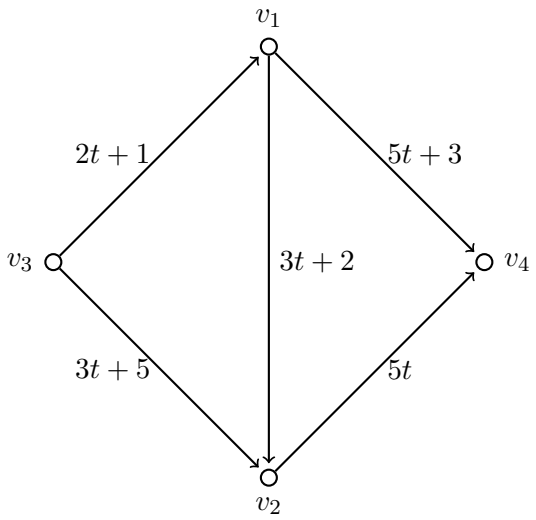
Algorithm 6 The foremost journey finding algorithm - periodic graphs FJP_EXTENDED

1: **procedure** FOREMOST JOURNEY IN PERIODIC GRAPHS($Graph$, $start\_time$)
2: **for all** vertices $v \in V(Graph)$ **do**
3: **for all** vertices $w \in V(Graph)$ **do** ▷ Initializations
4: S[v,w] := empty sequence;
5: **end for**
6: **end for**
7: **for all** vertices $w \in V(Graph)$ **do**
8: source := w; ▷ Current source
9: **for all** vertices $z \in V(Graph) - \{w\}$ **do**
10: destination := z; ▷ Current destination
11: **for all** vertices $v \in V(Graph)$ **do** ▷ Initializations
12: time_to[v] := infinity; ▷ Unknown time function
13: from source to v
14: previous[v] := undefined; ▷ Previous node in optimal
15: journey from source
16: **end for**

Algorithm 6 The foremost journey finding algorithm - periodic graphs FJP_EXTENDED (continued)

```
17:             time_to[source] := start_time;          ▷ Time to go from source
18:                                                       to source is the time
19:                                                       we start counting
20:             Q := the set of all nodes in Graph;     ▷ All nodes in Graph are
21:                                                       unoptimized - thus are
22:                                                       in Q
23:             while Q is not empty do                 ▷ The main loop
24:                 u := vertex in Q with smallest time in time_to[];   ▷ Start node
25:                                                                     in first case
26:                 remove u from Q;
27:                 if u == destination then
28:                     t := u;
29:                     while previous[t] is defined do
30:                         insert t at the beginning of S[source,destination];
31:                         t := previous[t];
32:                     end while
33:                 end if
34:                 for all neighbors v of u do
35:                     if time_to[u] not equal to infinity then
36:                         temp_0 := max{0, ⌊(time_to[u] − b_{u,v}) / a_{u,v} + 1⌋};
37:                         temp := a_{u,v} · temp_0 + b_{u,v};
38:                         if temp < time_to[v] then
39:                             time_to[v] := temp;
40:                             previous[v] := u;
41:                             decrease-key v in Q;    ▷ Reorder v in the queue Q
42:                         end if
43:                     end if
44:                 end for
45:             end while
46:             write "The foremost journey from", source, "to", destination, "is:",
    source, S[source,destination];
47:             return time_to[destination];
48:         end for
49:     end for
50:     return S;
51: end procedure
```

Εφαρμογή Αλλάζοντας τα ονόματα των κορυφών του Σχήματος 3.13, ώστε να μην έχουμε πια source ή destination, εκτελούμε τον Αλγόριθμο 6 στο κατευθυνόμενο χρονικό γράφημα του Σχήματος 3.16. Ως start_time λογίζεται η χρονική στιγμή ένδεκα (11).



Σχημα 3.16: Αλγόριθμος FJP_EXTENDED εύρεσης πρωτίστου ταξιδίου σε περιοδικό γράφημα για όλα τα διατεταγμένα ζεύγη κορυφών.

Ο αλγόριθμος FJP_EXTENDED επιστρέφει τον πίνακα των πρωτίστων ταξιδίων μεταξύ όλων των ζευγών των κορυφών του γραφήματος, όπως αυτός φαίνεται ακολούθως:

$$S = \begin{bmatrix} \{\} & \{v_2\} & \{\} & \{v_4\} \\ \{\} & \{\} & \{\} & \{v_4\} \\ \{v_1\} & \{v_1, v_2\} & \{\} & \{v_1, v_4\} \\ \{\} & \{\} & \{\} & \{\} \end{bmatrix}.$$

3.4 Το πρόβλημα REACHABILITY&AGE

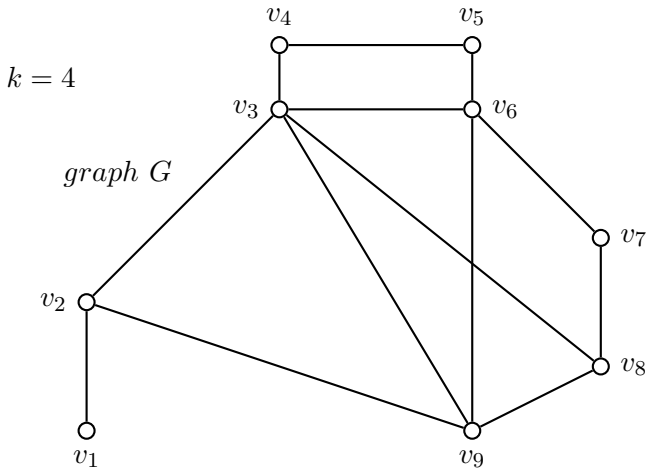
Στην παρούσα ενότητα, θα μελετήσουμε το λεγόμενο πρόβλημα της Προσβασιμότητας και της Ηλικίας ή απλούστερα *πρόβλημα "Προσβασιμότητα και Ηλικία" (Reachability&Age)*. Ας θυμηθούμε ότι λέμε πως μια ανάθεση χρονικών ετικετών, L, στις ακμές ενός γραφήματος G διατηρεί την προσβασιμότητα (reachability) του γραφήματος, εάν

$$\forall u, v \in V(G),\ \exists (u,v) - path\ in\ G \Rightarrow \exists (u,v) - journey\ in\ (G, L)$$

και ας προχωρήσουμε στον ορισμό του προβλήματος που θα μας απασχολήσει.

Ορισμός ***Πρόβλημα Reachability&Age***
Στιγμιότυπο Γράφημα G και θετικός ακέραιος k
Ερώτημα Υπάρχει ανάθεση L στις ακμές του G, η οποία διατηρεί την προσβασιμότητα και δίνει $age(G, L) \leq k$;

Με τη βοήθεια του σχήματος 3.17, ίσως μπορούμε ευκολότερα να αντιληφθούμε το πρόβλημα.



Σχημα 3.17: Παράδειγμα ζητήματος εύρεσης ανάθεσης που διατηρεί την προσβασιμότητα, δίνοντας ηλικία μικρότερη από δεδομένο αριθμό k.

Τώρα είναι περισσότερο κατανοητό πως δεν μπορούμε απλώς κοιτώντας το γράφημα του στιγμιοτύπου να μαντέψουμε την απάντηση στο ερώτημα που τίθεται στο πρόβλημα $Reachability\&Age$. Είναι εύλογο, επομένως, να αναρρωτηθούμε τι είδους πρόβλημα είναι αυτό. Είναι άραγε ένα απλό πρόβλημα, για την επίλυση του οποίου μπορεί να κατασκευαστεί ένας αλγόριθμος που εκτελείται σε πολυωνυμικό χρόνο;

Ή μήπως είναι πιο δύσκολο πρόβλημα, λ.χ. ανήκει στο NP; Μήπως μπορούμε να κάνουμε αναγωγή κάποιου γνωστού NP-πλήρους προβλήματος σε αυτό;

3.4.1 Πολυπλοκότητα προβλήματος Reachability&Age

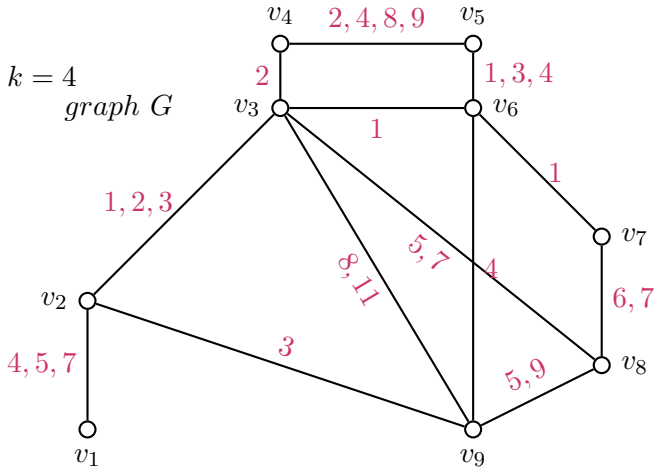
Το πρόβλημά μας ανήκει στο NP. Πράγματι, δεδομένου στιγμιοτύπου του προβλήματος και ανάθεσης L στις ακμές του γραφήματος G του προβλήματος, μπορώ σε πολυωνυμικό χρόνο να ελέγξω αν η L διατηρεί την προσβασιμότητα και δίνει ηλικία μικρότερη από k ή όχι.

Ο έλεγχος της ηλικίας είναι ο πιο απλός, αφού συνίσταται στην απλή εύρεση του μεγίστου ενός συνόλου τιμών. Όμως, και ο έλεγχος της διατήρησης της προσβασιμότητας δεν είναι τόσο δύσκολος τώρα που έχουμε ως εργαλείο τους αλγορίθμους των προηγούμενων υποενοτήτων!
Αρκεί να χρησιμοποιήσουμε τον κατάλληλο αλγόριθμο[1] ανάλογα με τον τύπο του χρονικού γραφήματος που μας δίνει η ανάθεση L (μονοεπισημασμένο, πολυεπισημασμένο, κλπ.) για να βρούμε όλα τα ζεύγη κορυφών u, v του (G,L), για τα οποία υπάρχει ταξίδι από τη u προς τη v, και εν συνεχεία να συγκρίνουμε τα τελευταία με τα ζεύγη κορυφών u', v' του G, για τα οποία υπάρχει μονοπάτι από τη u' προς τη v'[2]. Αν τα ζεύγη που θα βρούμε στις δύο περιπτώσεις ταυτίζονται, αποφαινόμαστε ότι η δεδομένη ανάθεση L διατηρεί την προσβασιμότητα. Διαφορετικά, απαντάμε πως κάτι τέτοιο δεν ισχύει.
Άρα, πράγματι, *το πρόβλημα Reachability&Age ανήκει στο NP*.

[1] Όλοι οι αλγόριθμοι είναι πολυωνυμικοί στο μέγεθος n των κορυφών.

[2] Υπάρχουν διάφοροι πολυωνυμικοί στο μέγεθος n των κορυφών αλγόριθμοι που απαντούν στο ερώτημα της σύνδεσης κορυφών σε ένα γράφημα, λ.χ. ο Floyd-Warshall.

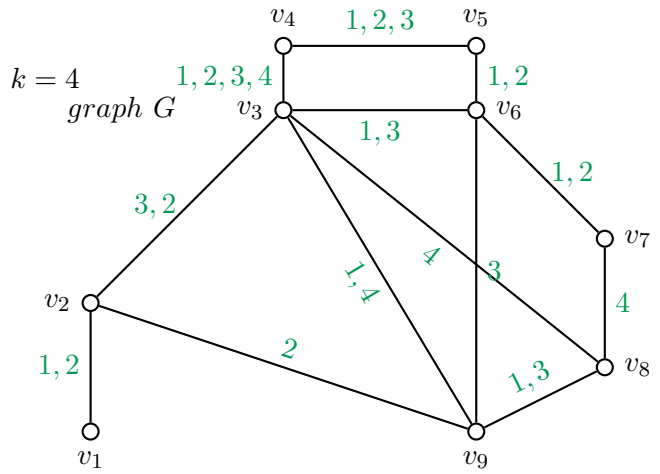
Παράδειγμα 1 Ας υποθέσουμε ότι μας δίνεται ανάθεση L στις ακμές του γραφήματος G του σχήματος 3.17, όπως φαίνεται στο σχήμα 3.18.



Σχημα 3.18: Παράδειγμα ελέγχου δεδομένης ανάθεσης για το αν διατηρεί την προσβασιμότητα και αν δίνει ηλικία μικρότερη από δεδομένο αριθμό k.

Η απάντηση στο ερώτημα αν η ηλικία του (G,L) χρονικού γραφήματος είναι το πολύ 4 είναι προφανώς "όχι".

Παράδειγμα 2 Ας υποθέσουμε τώρα ότι μας δίνεται ανάθεση L′ στις ακμές του γραφήματος G του σχήματος 3.17, όπως φαίνεται στο σχήμα 3.19



Σχημα 3.19: Παράδειγμα ελέγχου δεδομένης ανάθεσης για το αν διατηρεί την προσβασιμότητα και αν δίνει ηλικία μικρότερη από δεδομένο αριθμό k.

Στην περίπτωση αυτή, η απάντηση στο ερώτημα αν η ηλικία του (G,L) χρονικού γραφήματος είναι το πολύ 4 είναι "ναι". Όμως, διατηρεί η ανάθεση L′ την προσβασιμότητα του G; Η απάντηση είναι επίσης εύκολη, αφού παρατηρούμε ότι ενώ στο γράφημα G υπάρχει μονοπάτι (v_8, v_1), στο χρονικό γράφημα (G,L′) δεν υπάρχει κανένα (v_8, v_1)-ταξίδι. Συνεπώς, και πάλι η ανάθεση που δίνεται δε διατηρεί την προσβασιμότητα, δίνοντας ηλικία το πολύ 4.

3.4.1.1 Αλγόριθμος επίλυσης προβλήματος Reachability&Age

Αποδείξαμε πως το πρόβλημα $Reachability\&Age$ ανήκει στο NP. Τώρα, θα δείξουμε πως ανήκει στο P.
Σημειώνεται πως ο αλγόριθμος Floyd-Warshall που αναφέρεται ακολούθως επιστρέφει ένα διάνυσμα μεγέθους $|V(G)| \times |V(G)|$ που περιέχει τα μήκη των συντομότερων μονοπατιών μεταξύ όλων των ζευγών κορυφών του G.

Algorithm 7 Αλγόριθμος επίλυσης προβλήματος Reachability&Age

```
1: procedure REACHABILITY&AGE SOLVING(G,k)
2:     run Floyd-Warshall's algorithm on G, assuming unit edge weights
3:     l := maximum of all lengths stored in the array returned by Floyd-Warshall's
       algorithm
4:     L' := {{1, 2, ..., l}, ∀e ∈ E(G)}
5:     if k < l then
6:         Answer := no;
7:     else
8:         Answer := yes;
9:     end if
10:    return L', Answer;
11: end procedure
```

Ισχυριζόμαστε πως ο αλγόριθμος 7 επιλύει το πρόβλημα $Reachability\&Age$. Πριν, όμως, αποδείξουμε τον ισχυρισμό μας, ας διατυπώσουμε την ακόλουθη πρόταση:

ΠΡΟΤΑΣΗ

Η ανάθεση L' που κατασκευάζεται στη γραμμή 4 του αλγορίθμου διατηρεί την προσβασιμότητα.

Απόδειξη

Η ανάθεση L' δίνει όλες τις χρονικές ετικέτες $1, 2, \ldots, l$ σε κάθε ακμή του G. Έστω ζεύγος κορυφών $u, v \in V(G)$ τέτοιο, ώστε στο G υπάρχει (u, v)-μονοπάτι. Το συντομότερο (u, v)-μονοπάτι, έστω $path_0$, έχει μήκος $l(path_0) \leq l$.[3]

[3] Εξ' ορισμού του l, οποιοδήποτε μονοπάτι που είναι το συντομότερο μονοπάτι μεταξύ των άκρων του έχει μήκος το πολύ l.

Επιλέγοντας τις χρονικές ετικέτες $1, 2, \ldots, l(path_0)$ με τη σειρά αυτή στις $l(path_0)$ το πλήθος ακμές του $path_0$, βρίσκουμε ένα (u, v)-ταξίδι στο χρονικό γράφημα (G,L).

Άρα, η L' πράγματι διατηρεί την προσβασιμότητα του G.

Επιπλέον, η ανάθεση L' που κατασκευάζει ο αλγόριθμος 7 από κατασκευής της δίνει στο χρονικό γράφημα που προκύπτει ηλικία $age(G, L') = l$. Αν το δοθέν k στο στιγμιότυπο του προβλήματος είναι ένας αριθμός μεγαλύτερος από l ($l < k$), τότε η απάντηση στο ερώτημα του προβλήματος είναι θετική. Δηλαδή, υπάρχει ανάθεση L στις ακμές του γραφήματος G, η οποία διατηρεί την προσβασιμότητα του G και δίνει $age(G, L) \leq k$ και αυτή είναι η L' που κατασκεύασε ο αλγόριθμος.

Ο αλγόριθμος που κατασκευάσαμε, λοιπόν, απαντάει σωστά στο "ναι" του προβλήματος. Τι συμβαίνει με το "όχι"; Δηλαδή, αν δεν υπάρχει τέτοια ανάθεση, ο αλγόριθμος θα απαντήσει πάντα "όχι"; Η απάντηση είναι πως ναι, ο αλγόριθμος θα απαντήσει "όχι" στην περίπτωση που δεν υπάρχει μια ανάθεση στις ακμές του γραφήματος, η οποία πληρεί τις ζητούμενες προϋποθέσεις , αλλά πριν το αποδείξουμε, θα διατυπώσουμε ακόμη μία πρόταση.

ΠΡΟΤΑΣΗ

Δεν υπάρχει ανάθεση L στις ακμές γραφήματος G, η οποία διατηρεί την προσβασιμότητα του G, που να δίνει ηλικία $age(G, L) < l$, όπου l το μήκος του μεγαλύτερου συντομότερου μονοπατιού του G.

Απόδειξη

Έστω L ανάθεση ετικετών στις ακμές του G, η οποία διατηρεί την προσβασιμότητα. Έτσι, για κάθε ζεύγος κορυφών $u, v \in V(G)$, για τις οποίες στο G υπάρχει (u, v)-μονοπάτι, στο (G,L) υπάρχει (u, v)-ταξίδι. Αυτό το (u, v)-ταξίδι έχει τουλάχιστον τόσες χρονικές ακμές, όσες ακμές έχει το συντομότερο (u, v)-μονοπάτι.[4]
Επειδή τα παραπάνω ισχύουν για κάθε ζεύγος $u, v \in V(G)$, για τα οποίες στο G υπάρχει (u, v)-μονοπάτι, είναι προφανές ότι η L έχει μέγιστη χρησιμοποιούμενη χρονική ετικέτα τουλάχιστον ίση με το μήκος του μεγαλύτερου συντομότερου μονοπατιού που συναντάμε μεταξύ όλων των ζευγών κορυφών του G.

Δηλαδή, ισχύει $age(G, L) \geq l$.

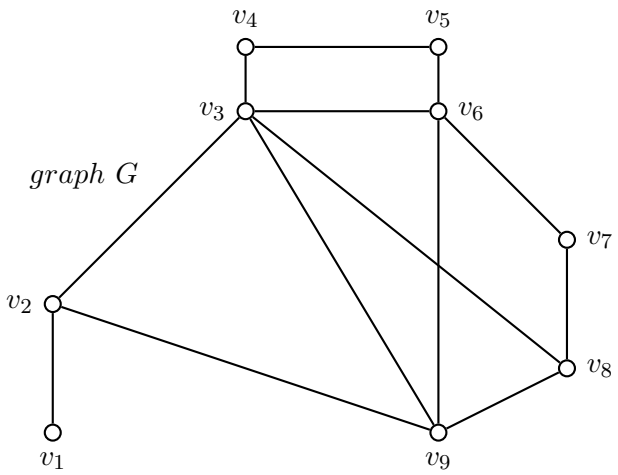
Επομένως, το l που ορίζεται στη γραμμή 3 του αλγορίθμου 7 είναι η μικρότερη δυνατή ηλικία που μπορεί να δώσει μια ανάθεση που διατηρεί την προσβασιμότητα

[4] Αυτό συμβαίνει γιατί το τυχαίο (u, v)-ταξίδι χρησιμοποιεί ακριβώς τις ακμές κάποιου (u, v)-μονοπατιού.

του G. Έτσι, αν $k < l$, τότε δεν υπάρχει ανάθεση L στις ακμές του G, η οποία διατηρεί την προσβασιμότητα του G και δίνει ηλικία $age(G, L) \leq l$.

Άρα, *ο αλγόριθμος 7 πράγματι επιλύει το πρόβλημα Reachability&Age.*

Εφαρμογή Ας δούμε τώρα μια εφαρμογή του αλγορίθμου επίλυσης του προβλήματος *Reachability&Age* με είσοδο το γράφημα G του Σχήματος 3.20 και το θετικό ακέραιο $k = 4$.

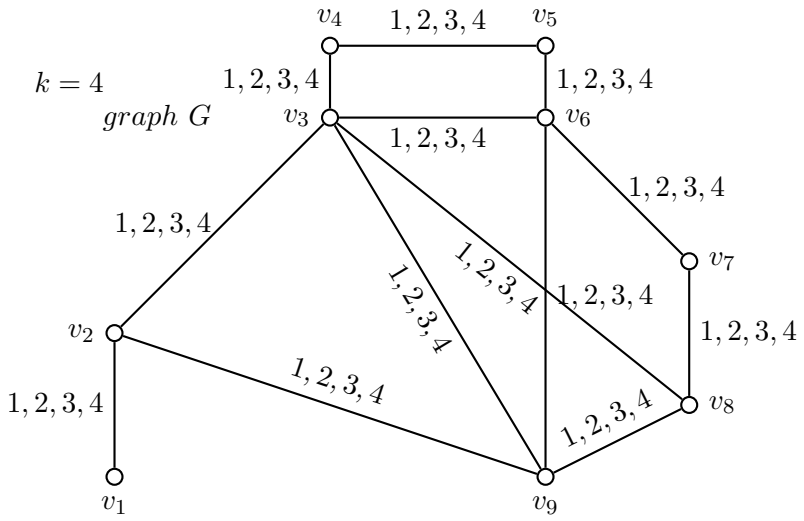


Σχημα 3.20: Εφαρμογή αλγορίθμου Reachabilit&Age

Για ευκολία στην ανάγνωση, γράφουμε το διάνυσμα που επιστρέφει ο αλγόριθμος Floyd-Warshall στη γραμμή 2 του Αλγορίθμου Επίλυσης του προβλήματος *Reachability&Age*, με μορφή πίνακα όπως φαίνεται ακολούθως:

$$dist = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 3 & 4 & 3 & 2 \\ 1 & 0 & 1 & 2 & 3 & 2 & 3 & 2 & 1 \\ 2 & 1 & 0 & 1 & 2 & 1 & 2 & 1 & 1 \\ 3 & 2 & 1 & 0 & 1 & 2 & 3 & 2 & 2 \\ 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 2 \\ 3 & 2 & 1 & 2 & 1 & 0 & 1 & 2 & 1 \\ 4 & 3 & 2 & 3 & 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 2 & 3 & 2 & 1 & 0 & 1 \\ 2 & 1 & 1 & 2 & 2 & 1 & 2 & 1 & 0 \end{bmatrix}.$$

Στη γραμμή 3, ο αλγόριθμος υπολογίζει το μέγιστο στοιχείο του παραπάνω πίνακα, $l = 4$, και στη γραμμή 4 κατασκευάζει την ανάθεση $L' := \{\{1,2,3,4\}, \forall e \in E(G)\}$ (βλ. Σχήμα 3.21).



Σχημα 3.21: Παράδειγμα εφαρμογής αλγορίθμου επίλυσης προβλήματος Reachability&Age - Ανάθεση $L' := \{\{1,2,3,4\}, \forall e \in E(G)\}$.

Τέλος, επειδή βρισκόμαστε στην περίπτωση $k \geq l$, ο αλγόριθμος απαντά "ναι" και επιστρέφει την L'.

Κεφάλαιο 4

Η Στατιστική στα Χρονικά Γραφήματα

4.1 Εισαγωγή

Όπως μαρτυρά και ο τίτλος του, το κεφάλαιο αυτό είναι μια προσπάθεια εφαρμογής των γνώσεών μας γύρω από τη Στατιστική στα Χρονικά Γραφήματα. Θα μελετήσουμε στατιστικά μέτρα κάποιων χαρακτηριστικών των χρονικών γραφημάτων, πιθανότητες να συμβούν ενδιαφέροντα γεγονότα κ.ά.

Στο εξής, εάν δεν αναφέρεται διαφορετικά, θεωρούμε πως κάθε ακμή οποιουδήποτε γραφήματος λαμβάνει ακριβώς μία ετικέτα διαθεσιμότητας και οι ετικέτες αυτές επιλέγονται τυχαία, ανεξάρτητα η μία από την άλλη από το σύνολο L_0=$\{1, 2, \ldots, a\}$, όπου $a \in \mathbb{N}$, με την πιθανότητα η ετικέτα μιας ακμής να είναι i, $\forall i \in L_0$, να είναι ίση με $\frac{1}{a}$. **(ΥΠΟΘΕΣΗ-UNI)**

Σημείωση Θα μπορούσαν μελλοντικά να μελετηθούν περιπτώσεις, στις οποίες κάθε ακμή ενός γραφήματος δύναται να πάρει περισσότερες από μία χρονικές ετικέτες διαθεσιμότητας και οι ετικέτες αυτές επιλέγονται τυχαία, ανεξάρτητα η μία από την άλλη από το σύνολο L_0=$\{1, 2, \ldots, a\}$, όπου $a \in \mathbb{N}$, με την επιλογή των ετικετών διαθεσιμότητας μιας ακμής να ακολουθεί μια άλλη κατανομή, έστω F.
(ΥΠΟΘΕΣΗ-F)

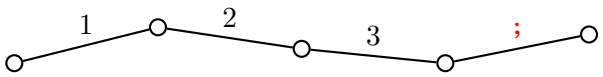
Στις ενότητες που ακολουθούν, θα εξετάσουμε το αναμενόμενο πλήθος ταξιδίων που δημιουργούνται πάνω σε μονοπάτια k ακμών σε ειδικές περιπτώσεις γραφημάτων που ικανοποιούν την ΥΠΟΘΕΣΗ-UNI. Χάριν συντομίας, θα καλούμε τέτοιου είδους ταξίδια *"χρονικά μονοπάτια" k ακμών*.

Θα μελετήσουμε, επίσης, τη χρονική διάμετρο ενός γραφήματος, όπως αυτή θα οριστεί ως ο μέγιστος αναμενόμενος συντομότερος χρόνος μετάβασης από μια κορυφή s του γραφήματος σε μια κορυφή t του γραφήματος, για όλα τα ζεύγη s,t κορυφών του γραφήματος.

4.2 Αναμενόμενο πλήθος ταξιδίων σε πλήρες γράφημα

Στην παρούσα ενότητα θα αναζητήσουμε το αναμενόμενο πλήθος χρονικών μονοπατιών k ακμών σε πλήρες γράφημα n κορυφών (κλίκα K_n) που ικανοποιεί την ΥΠΟΘΕΣΗ-UNI.

Είναι προφανές ότι για να υπάρχει κάποιο χρονικό μονοπάτι k ακμών σε **οποιοδήποτε** γράφημα, πρέπει το πλήθος ακμών, k, να είναι το πολύ ίσο με τη μέγιστη ετικέτα, a, του συνόλου L_0 των ετικετών που μπορούμε να αναθέσουμε στις διάφορες ακμές. Σε αντίθετη περίπτωση, είναι αδύνατη η ύπαρξη χρονικού μονοπατιού k ακμών (βλ. Σχήμα 4.1).



$L_0 = \{1, 2, 3(= a)\}$

$k = 4$

(Αδυναμία ύπαρξης χρονικού μονοπατιού 4 ακμών!)

Σχημα 4.1: Αδυναμία ύπαρξης χρονικού μονοπατιού, όταν $k > a$

4.2.1 Ειδική περίπτωση: $G = K_n,\ k = n-1,\ a = n-1$

Επικεντρώνουμε αρχικά το ενδιαφέρον μας στην περίπτωση της κλίκας (πλήρους γραφήματος) n κορυφών K_n, που ικανοποιεί την ΥΠΟΘΕΣΗ-UNI με $a = n-1$ (δηλαδή, με $L_0 = \{1, 2, , \ldots, n-1\}$), στην οποία αναζητούμε το αναμενόμενο πλήθος χρονικών μονοπατιών $n-1$ ακμών.

Προφανώς, στις $k = n-1$ ακμές ενός τυχαίου μονοπατιού της κλίκας K_n με αρχική κορυφή μια τυχαία $v_0 \in V(K_n)$, μόνο μία ανάθεση των ετικετών του L_0 μπορεί να γίνει ώστε να προκύπτει χρονικό μονοπάτι. Η ανάθεση αυτή δίνει στην 1^{η} ακμή την ετικέτα 1, στη 2^{η} ακμή την ετικέτα 2, ... , στη $(n-1)^{οστή}$ ακμή την ετικέτα $n-1$.

Κάθε ακμή έχει τη δυνατότητα να λάβει ακριβώς μία ετικέτα από ένα σύνολο $n-1$ το πλήθος ετικετών. Επομένως, το συνολικό πλήθος αναθέσεων ετικετών που μπορούν να γίνουν στις $n-1$ αυτές ακμές είναι:

$$\#assignments = (n-1)^{n-1}$$

Συνεπώς, δοθέντος μονοπατιού $n-1$ ακμών που ξεκινά από τη v_0, η πιθανότητα να υπάρξει το αντίστοιχο χρονικό μονοπάτι (δηλαδή, αυτό που προκύπτει πάνω στο απλό μονοπάτι μετά την ανάθεση των χρονικών ετικετών) είναι:

$$P(temporal\_path\_of\_length\_n-1\_starting\_from\_v_0) = \frac{1}{(n-1)^{n-1}}$$

Το πλήθος των μονοπατιών που ξεκινούν από τη v_0 στην κλίκα K_n και έχουν μήκος $n-1$ είναι ίσο με το πλήθος των διατάξεων των $n-1$ κορυφών που απομένουν -εκτός, δηλαδή, της αρχής v_0- για να κατασκευαστεί τέτοιο μονοπάτι. Δηλαδή, το πλήθος των μονοπατιών που ξεκινούν από τη v_0 στην κλίκα K_n και έχουν μήκος $n-1$ είναι:

$$(n-1)!$$

Άρα, αφού η κλίκα K_n έχει n κορυφές, και λόγω της γραμμικότητας της μέσης τιμής, το αναμενόμενο πλήθος χρονικών μονοπατιών $k = n-1$ ακμών σε αυτήν είναι:

$$E(\#\_temporal\_paths\_of\_length\_n-1) = n \cdot (n-1)! \cdot \frac{1}{(n-1)^{n-1}} = \frac{n!}{(n-1)^{n-1}}$$

Παρατηρήσεις Ας παρατηρήσουμε ότι όταν το n είναι πολύ μεγάλο ($n \to +\infty$), τότε από τον τύπο του Stirling προκύπτει ότι:

$$\begin{aligned} E(\#temporal\_paths\_of\_length\_n-1) &= \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{(n-1)^{n-1}} \\ &= \frac{\sqrt{2\pi n} n^n}{e^n (n-1)^{n-1}} \xrightarrow[n\to+\infty]{} 0 \end{aligned}$$

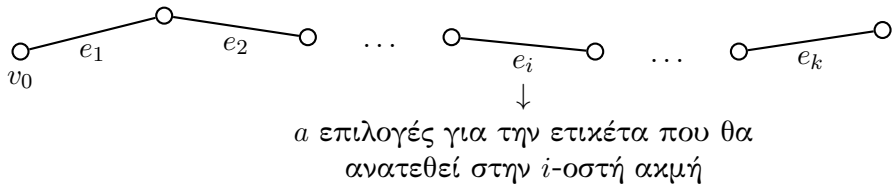
Βέβαια, κάτι τέτοιο είναι λίγο ή πολύ προφανές αν σκεφτούμε ότι είναι δύσκολο να βρούμε χρονικά μονοπάτια μήκους $n-1$ ακμών σε κλίκα n κορυφών όταν το n είναι πολύ μεγάλο. Αυτό συμβαίνει γιατί για να υπάρχει ένα χρονικό μονοπάτι αυτού του μήκους θα πρέπει να μπορέσουν να ανατεθούν έτσι οι (τόσες πολλές) χρονικές ετικέτες που να τηρούν την επιθυμητή αυστηρά αύξουσα σειρά, κάτι που συμβαίνει με όλο και μικρότερη πιθανότητα όσο το n αυξάνει.

4.2.2 Ειδική περίπτωση: $G = K_n,\ k < a,\ a \geq n$

Ας δούμε τώρα τι συμβαίνει στην περίπτωση της κλίκας K_n, που ικανοποιεί την ΥΠΟΘΕΣΗ-UNI, όταν αναζητούμε το αναμενόμενο πλήθος χρονικών μονοπατιών $k < a$ ακμών και η μέγιστη ετικέτα που μπορεί να ανατεθεί σε κάποια ακμή της κλίκας είναι $a \geq n$.

Ξεκινώντας από μια τυχαία κορυφή $v_0 \in V(K_n)$ και κατά μήκος μονοπατιού k ακμών, μπορούμε, όπως εξηγεί το Σχήμα 4.2, να κατασκευάσουμε πλήθος αναθέσεων ίσο με:

$$\#assignments = a^k$$



Σχημα 4.2: Πλήθος αναθέσεων που μπορούν να γίνουν σε μονοπάτι k ακμών, όταν $k < a$

Το πλήθος των αναθέσεων που μπορούν να γίνουν στις k ακμές, στις οποίες οι ετικέτες που ανατίθενται είναι διακριτές (διαφορετικές μεταξύ τους) είναι:

$$\#non\_equal\_(distinct)\_times\_assignments = a \cdot (a-1) \cdot \ldots \cdot (a-k+1) = \frac{a!}{(a-k)!}$$

Το πλήθος των μονοπατιών μήκους k που μπορεί να ξεκινούν από την τυχαία $v_0 \in V(K_n)$ είναι:

$$\#paths\_of\_length\_k\_starting\_from\_v_0 = (n-1) \cdot (n-2) \cdot \ldots \cdot (n-k) = \frac{(n-1)!}{(n-k-1)!}$$

καθώς έχουμε $n-1$ επιλογές για την κορυφή v_1 που θα επιλεγεί ως επόμενη της v_0 στο μονοπάτι, $n-2$ επιλογές για την κορυφή v_2 που θα επιλεγεί ως επόμενη της v_1 στο μονοπάτι, κ.ο.κ., και τέλος $n-k$ επιλογές για την κορυφή v_k που θα επιλεγεί ως τελευταία στο μονοπάτι.

Καλούμε A το ενδεχόμενο να "έχουμε σωστή ανάθεση ετικετών στις k ακμές τυχαίου μονοπατιού μήκους k που ξεκινά από τη v_0".
Δηλαδή, αν $l_1, l_2, \ldots, l_k$ οι χρονικές ετικέτες που ανατίθενται στην 1^η, τη 2^η, ... , την $k^{οστή}$ ακμή του μονοπατιού αντιστοίχως, με $l_i \in L_0 = \{1, 2, \ldots, a\},\ \forall i = 1, 2, \ldots, k,$

το A είναι το ενδεχόμενο να ισχύει:

$$l_1 < l_2 < \ldots < l_k$$

Καλούμε ϕ την πιθανότητα να συμβεί το ενδεχόμενο A:

$$\phi = P(A) = P(l_1 < l_2 < \ldots < l_k)$$

Ας σημειωθεί εδώ ότι όλες οι διατάξεις k ετικετών της μορφής

$$l_{a_1} < l_{a_2} < \ldots < l_{a_k}$$

είναι $k!$ το πλήθος και ισοπίθανες με πιθανότητα ίση με $P(A)$.
Επομένως, αν θεωρήσουμε B το ενδεχόμενο να "υπάρχει ισότητα *τουλάχιστον* δύο ετικετών από αυτές που έχουν ανατεθεί στις k ακμές του μονοπατιού", μπορούμε να πούμε πως ισχύει το εξής:

$$k! \cdot P(A) + P(B) = 1 \Leftrightarrow$$

$$k! \cdot \phi + 1 - P(\urcorner B) = 1 \tag{4.1}$$

Η πιθανότητα να συμβεί το ενδεχόμενο $\urcorner B$, δηλαδή να μην υπάρχει καμία ισότητα στις ετικέτες που έχουν ανατεθεί στις k ακμές του μονοπατιού, είναι:

$$P(\urcorner B) = \frac{\#non\_equal\_(distinct)\_times\_assignments}{\#assignments} =$$

$$= \frac{\frac{a!}{(a-k)!}}{a^k}$$

$$= \frac{a!}{a^k \cdot (a-k)!}$$

Επομένως, η σχέση (4.1) γίνεται:

$$k! \cdot \phi + 1 - \frac{a!}{a^k \cdot (a-k)!} = 1 \Leftrightarrow$$

$$\Leftrightarrow \phi = \frac{a!}{k! \cdot a^k \cdot (a-k)!}$$

Υπενθυμίζουμε ότι ϕ είναι η πιθανότητα να έχουμε σωστή ανάθεση στις k ακμές του τυχαίου μονοπατιού μήκους k που ξεκινά από την τυχαία v_0 κορυφή της κλίκας K_n. Υπενθυμίζουμε, επίσης, πως το πλήθος των μονοπατιών μήκους k που μπορεί να

ξεκινούν από τυχαία v_0 κορυφή της κλίκας K_n είναι $\frac{(n-1)!}{(n-k-1)!}$.
Επομένως, το αναμενόμενο πλήθος μονοπατιών μήκους k που ξεκινούν από την τυχαία v_0 και στα οποία ανατίθενται σωστά ετικέτες, ώστε να προκύπτουν χρονικά μονοπάτια, είναι:

$$E(\#temporal\_paths\_of\_length\_k\_starting\_from\_v_0) = \frac{(n-1)!}{(n-k-1)!} \cdot \phi$$

Τέλος, αφού η κλίκα K_n έχει n το πλήθος κορυφές, το αναμενόμενο πλήθος μονοπατιών μήκους k, στα οποία ανατίθενται σωστά ετικέτες, ώστε να προκύπτουν χρονικά μονοπάτια στην κλίκα K_n, είναι:

$$E(\#temporal\_paths\_of\_length\_k) = n \cdot \frac{(n-1)!}{(n-k-1)!} \cdot \phi$$

$$= \frac{n \cdot (n-1)!}{(n-k-1)!} \cdot \frac{a!}{k! \cdot a^k \cdot (a-k)!}$$

$$= \frac{n! \cdot a!}{(n-k-1)! \cdot k! \cdot a^k \cdot (a-k)!}$$

Παρατηρήσεις Ας παρατηρήσουμε ότι η πιθανότητα ϕ είναι:

$$\phi = \frac{1}{k!} \cdot \frac{\overbrace{a(a-1)\dots(a-k+1)}^{\text{k παράγοντες}}}{\underbrace{a \cdot \dots \cdot a}_{\text{k παράγοντες}}}$$

και έτσι, αν το a είναι πολύ μεγάλο εν συγκρίσει με το k, τότε γίνεται $\phi \approx \frac{1}{k!}$.
Συνεπώς, αν το a είναι πολύ μεγάλο σε σχέση με το k, τότε το αναμενόμενο πλήθος μονοπατιών μήκους k, στα οποία ανατίθενται σωστά ετικέτες, ώστε να προκύπτουν χρονικά μονοπάτια στην κλίκα K_n, είναι:

$$E(\#temporal\_paths\_of\_length\_k) \approx \frac{n!}{k!(n-k-1)!} = \frac{n \cdot (n-1) \cdot \dots \cdot (n-k)}{k!}$$

4.3 Η Χρονική Διάμετρος

Στις ενότητες που ακολουθούν στο κεφάλαιο αυτό θα ορίσουμε και θα μελετήσουμε μια νέα έννοια, αυτή της *χρονικής διαμέτρου (temporal diameter)*.
Πριν όμως συνεχίσουμε, ας συμφωνήσουμε ότι ένα χρονικό γράφημα -κατευθυνόμενο ή μη- $G = \{V, E, L\}$ που ικανοποιεί την ΥΠΟΘΕΣΗ-UNI, όπως αυτή ορίστηκε στην

ενότητα 4.1, θα καλείται στο εξής *Ομοιόμορφο Τυχαίο Χρονικό Γράφημα (Uniform Random Temporal Digraph or Graph, U-RTD or U-RTG respectively).*

Συνεπώς, δίδονται οι ορισμοί:

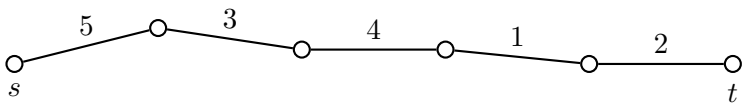
Ομοιόμορφο Τυχαίο Χρονικό Γράφημα: Χρονικό γράφημα $G = \{V, E, L\}$, σε κάθε ακμή, $e \in E(G)$, του οποίου ανατίθεται μοναδική ετικέτα διαθεσιμότητας και οι ετικέτες αυτές επιλέγονται τυχαία, ανεξάρτητα η μία από την άλλη από το σύνολο L_0=$\{1, 2, \ldots, a\}$, όπου $a \in \mathbb{N}$, με την πιθανότητα η ετικέτα μιας ακμής να είναι i, να είναι ίση με $\frac{1}{a}$, $\forall i \in L_0$.

Κανονικοποιημένο (Normalized) Ομοιόμορφο Τυχαίο Χρονικό Γράφημα: Ομοιόμορφο Τυχαίο Χρονικό Γράφημα $G = \{V, E, L\}$, στις ακμές του οποίου η μέγιστη ετικέτα που δύναται να ανατεθεί είναι $a = n = |V(G)|$. Δηλαδή, είναι L_0=$\{1, 2, \ldots, n\}$.

Έστω $G(L)$ στιγμιότυπο ενός U-RTG ή ενός U-RTD. Δεδομένων κορυφών $s, t \in V\big(G(L)\big)$, ορίζουμε:

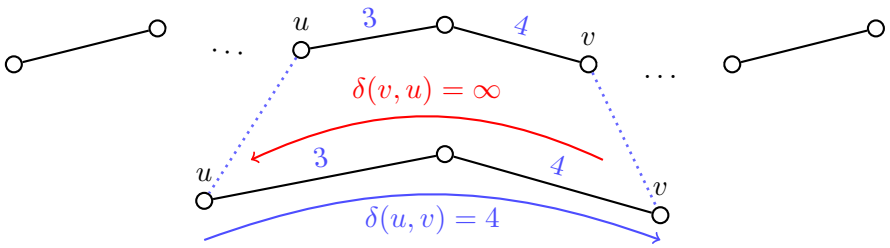
- $\delta(s,t) = a(j)$, όπου j ένα πρώτιστο $(s,t)-$ταξίδι: καλείται ***χρονική απόσταση*** της t από την s και είναι ο ελάχιστος χρόνος που απαιτείται για να φθάσω στην κορυφή t, ξεκινώντας από την κορυφή s (υπό την ανάθεση L). Αν δεν υπάρχει $(s,t)-$ταξίδι, τότε $\delta(s,t) = +\infty$
- $d = max_{s,t \in V(G)} E\big(\delta(s,t)\big)$: καλείται ***Χρονική Διάμετρος (Temporal Diameter)*** του G

Παρατηρήσεις Αν το $U-RTG$ ή $U-RTD$ γράφημα G είναι το ίδιο ένα μονοπάτι, τότε η χρονική διάμετρος του G προφανώς τείνει στο άπειρο ($d \to +\infty$).



Σχημα 4.3: Χρονική Διάμετρος $U-RTG$ γραφήματος που είναι το ίδιο ένα μονοπάτι

Αυτό γίνεται εύκολα κατανοητό αν σκεφτούμε ότι για οποιεσδήποτε κορυφές u και v του μονοπατιού, αν υπάρχει (u,v)-ταξίδι, τότε οι χρονικές ετικέτες στις ακμές του ταξιδίου αυτού τηρούν αυστηρά αύξουσα σειρά και συνεπώς στο ίδιο γράφημα (μονοπάτι) δεν υπάρχει (v,u)-ταξίδι (βλ. Σχήμα 4.4).

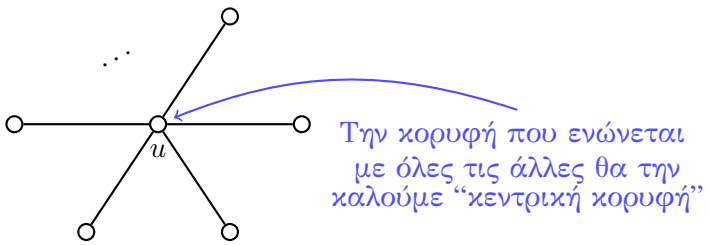


Σχημα 4.4: Γιατί η χρονική διάμετρος του μονοπατιού είναι άπειρη;

Η διάμετρος ενός γραφήματος είναι η μέγιστη απόσταση που συναντάται στο γράφημα μεταξύ δύο κορυφών του, όπου απόσταση της κορυφής v από την κορυφή u είναι το μήκος του συντομότερου μονοπατιού από τη u προς τη v.
Κατ' αντιστοιχία, η χρονική διάμετρος ενός χρονικού γραφήματος είναι η μέγιστη από τις αναμενόμενες χρονικές αποστάσεις μεταξύ των ζευγών των κορυφών του χρονικού γραφήματος.

4.3.1 Η Χρονική Διάμετρος γνωστών γραφημάτων

Ακολούθως, θα μελετήσουμε τη χρονική διάμετρο δύο γνωστών μας γραφημάτων, του γραφήματος-αστεριού (star graph) n κορυφών, που θα συμβολίσουμε G_{star} (βλ. Σχήμα 4.5) και του πλήρους γραφήματος ή κλίκας n κορυφών, K_n (βλ. Σχήμα 4.6).

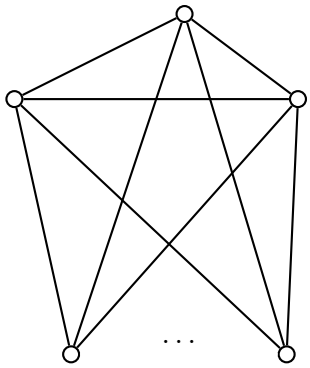


Σχημα 4.5: Ένα γράφημα-αστέρι (star graph)

4.3.1.1 Περίπτωση: $G = G_{star}$

Γνωρίζουμε ότι η διάμετρος οποιουδήποτε γραφήματος-αστεριού είναι:

$$diam(G_{star}) = 2$$



Σχημα 4.6: Ένα πλήρες γράφημα n κορυφών (clique)

Τι συμβαίνει με τη χρονική διάμετρο ενός U-RTG που είναι γράφημα-αστέρι;

Είναι εύκολο να αντιληφθεί κανείς ότι, ακόμη και αν το τυχαίο χρονικό γράφημα-αστέρι δεν ικανοποιεί την ΥΠΟΘΕΣΗ-UNI, αλλά την ΥΠΟΘΕΣΗ-F όπως αυτή ορίστηκε στην 4.1, είναι:

$$max_{s,t\in V(G_{star})} E_F\big(\delta(s,t)\big) \geq 2, \text{ για οποιαδήποτε κατανομή } F$$

Συνεπώς, *η χρονική διάμετρος ενός χρονικού γραφήματος-αστεριού είναι τουλάχιστον όση η διάμετρος ενός γραφήματος-αστεριού.*

Θα υπολογίσουμε ακριβώς τη Χρονική Διάμετρο ενός ομοιόμορφου τυχαίου χρονικού γραφήματος-αστεριού. Είναι:

$$\begin{aligned}
d &= max_{s,t\in V(G_{star})} E\big(\delta(s,t)\big) \\
&= E\big(\delta(s,t)\big), \text{ για οποιεσδήποτε κορυφές } s,t \in V(G_{star}) \\
&= E(l_2|\ l_2 > l_1) \\
&= \sum_{i=1}^{a} E(l_2|\ l_2 > i)\cdot P(l_1 = i) \\
&= \sum_{i=1}^{a} \Big(\sum_{i'=i}^{a} \big(P(l_2 = i'+1)\cdot(i'+1)\big)\Big)\cdot P(l_1 = i) \\
&= \sum_{i=1}^{a} \Big(\sum_{i'=i}^{a} (i'+1)\cdot\frac{1}{a}\Big)\cdot\frac{1}{a} \\
&= \frac{1}{a^2}\cdot\sum_{i=1}^{a}\sum_{i'=i}^{a}(i'+1) \\
&= \frac{1}{a^2}\cdot\Big(\sum_{i'=1}^{a}(i'+1) + \sum_{i'=2}^{a}(i'+1) + \ldots + \sum_{i'=a}^{a}(i'+1)\Big)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{a^2} \cdot \Big(\big(2+3+\ldots+(a+1)\big) + \big(3+4+\ldots+(a+1)\big) + \ldots + \big((a+1)\big) \Big) \\
&= \frac{1}{a^2} \cdot \Big(1\cdot 2 + 2\cdot 3 + 3\cdot 4 + 4\cdot 5 + \ldots + a\cdot(a+1) \Big) \\
&= \frac{1}{a^2} \cdot \sum_{i=1}^{a} \Big(i\cdot(i+1) \Big) \\
&= \frac{1}{a^2} \cdot \sum_{i=1}^{a} \Big(i^2 + i \Big) \\
&= \frac{1}{a^2} \cdot \sum_{i=1}^{a} i^2 + \sum_{i=1}^{a} i \\
&= \frac{1}{a^2} \cdot \Big(\frac{a\cdot(a+1)\cdot(2a+1)}{6} + \frac{a\cdot(a+1)}{2} \Big) \\
&= \frac{1}{a^2} \cdot \frac{a\cdot(a+1)\cdot(2a+1) + 3\cdot a\cdot(a+1)}{6} \\
&= \frac{a\cdot(a+1)\cdot(2a+4)}{6\cdot a^2}
\end{aligned}
$$

Τελικά, προκύπτει ότι η χρονική διάμετρος του γραφήματος-αστεριού είναι:

$$d = \frac{(a+1)(a+2)}{3a}$$

4.3.1.2 Περίπτωση: $G = K_n$

Στην περίπτωση του πλήρους γραφήματος n κορυφών, K_n, η διάμετρος είναι ίση με 1. Επομένως, και εδώ ισχύει ότι *η χρονική διάμετρος της χρονικής κλίκας είναι τουλάχιστον όση η διάμετρος της κλίκας*, αφού στις ακμές της πρώτης ανατίθενται θετικές ακέραιες χρονικές ετικέτες.

Θα μελετήσουμε τώρα την περίπτωση της κλίκας σε μεγαλύτερο βάθος. Για αρχή, ας παρατηρήσουμε ότι σε ένα U-RTG που είναι κλίκα, η χρονική διάμετρος είναι πάντα πραγματικός αριθμός, καθώς για οποιεσδήποτε κορυφές s, t της κλίκας, είναι $\delta(s,t) \leq a$. Συνεπώς:

$$d = max_{s,t \in V(K_n)} E\big(\delta(s,t)\big) \leq a$$

Κανονικοποιημένη ομοιόμορφη τυχαία χρονική κλίκα Έστω $G = K_n$ μια κλίκα n κορυφών και ας θεωρήσουμε την κανονικοποιημένη U-εκδοχή της. Δηλαδή, σε κάθε ακμή $e \in E(K_n)$ ανατίθεται μοναδική ετικέτα διαθεσιμότητας και οι ετικέτες αυτές επιλέγονται τυχαία, ανεξάρτητα η μία από την άλλη από το σύνολο L_0=$\{1, 2, \ldots, n\}$, με την πιθανότητα η ετικέτα μιας ακμής να είναι i, να είναι ίση με $\frac{1}{n}$, $\forall i \in L_0$.

Για οποιεσδήποτε κορυφές s, t της κλίκας, είναι:

$$E\big(l(e=\{s,t\})\big) = \frac{n}{2}$$

και επομένως είναι:

$$d = max_{s,t\in V(K_n)} E\big(\delta(s,t)\big) \leq \frac{n}{2}$$

Ακολούθως, παρουσιάζουμε τον Αλγόριθμο 8, ισχυριζόμενοι ότι βρίσκει ένα σύντομο ταξίδι από δοθείσα κορυφή s προς δοθείσα κορυφή t της κανονικοποιημένης ομοιόμορφης τυχαίας χρονικής κλίκας, δίνοντας , έτσι, ένα καλύτερο άνω όριο για τη χρονική διάμετρό της.

Algorithm 8 The normalized U-RTG clique short journey finding algorithm

1: **procedure** EXTEND-TRY(*clique* K_n, s, t, c_1, k)
2: **for** i = 0 ... $c_1\sqrt{n}logn$ **do**
3: s_i := undefined;
4: **end for**
5: s_0 := s;
6: **for** i = 0 ... $c_1\sqrt{n}logn$ **do**
7: **if** $l(\{s_i,t\}) \in (c_1\sqrt{n}lognk, c_1\sqrt{n}lognk+\sqrt{n})$ **then**
8: Follow directly the edge $\{s_i,t\}$; Success!
9: **else**
10: **if** $\exists u \in U \setminus \{t\}$ (where U stands for the set of the unvisited vertices) such,
11: that $l(\{s_i,u\}) \in \big(k\cdot i, k(i+1)\big)$ **then**
12: $s_{i+1} = u$;
13: go to line 6
14: **else**
15: follow directly the edge $\{s_i,u\}$ with the smallest $l(\{s_i,u\})$ among
16: all $u \in U$; Failure!
17: **end if**
18: **end if**
19: **end for**
20: **for** i = 0 ... $c_1\sqrt{n}logn$ **do**
21: **return** s_i;
22: **end for**
23: **end procedure**

Ακολουθεί μια ανάλυση του αλγορίθμου 8, στην οποία ελέγχεται η πιθανότητα με την οποία αυτός επιτυγχάνει.

Η πιθανότητα η χρονική ετικέτα της ακμής $\{s_i,t\}$ να βρίσκεται στο διάστημα $(c_1\sqrt{n}k, c_1\sqrt{n}k+\sqrt{n})$ και συνεπώς να έχουμε επιτυχία στην $(i+1)^{\text{οστή}}$ επανάληψη του αλγορίθμου, είναι:

$$P\Big(l(\{s_i,t\}) \in (c_1\sqrt{n}lognk, c_1\sqrt{n}lognk+\sqrt{n})\Big) = \frac{\sqrt{n}}{n} = \frac{1}{\sqrt{n}}$$

Έστω ε_1^j το ενδεχόμενο:

"Ο αλγόριθμος βρίσκει ένα σωστό ταξίδι $s_0s_1, s_1s_2, s_2, s_3, \ldots, s_{j-1}s_j$"

με την έννοια ότι βρίσκει χρονικό μονοπάτι, στις χρονικές ακμές του οποίου ακολουθείται αυστηρά αύξουσα σειρά χρονικών ετικετών και μάλιστα η ετικέτα της i-οστής χρονικής ακμής ανήκει στο σωστό διάστημα $((i-1)k, ik)$. Επειδή οι ακμές λαμβάνουν τις ετικέτες τους ανεξάρτητα η μία από την άλλη, η πιθανότητα να συμβεί το ε_1^j είναι το γινόμενο των πιθανοτήτων:

$P\Big(\exists s_1$ unvisited vertex : η ακμή $\{s_0, s_1\}$ έχει χρονική ετικέτα $l(\{s_0, s_1\}) \in (0, k)\Big)$
$P\Big(\exists s_2$ unvisited vertex : η ακμή $\{s_1, s_2\}$ έχει χρονική ετικέτα $l(\{s_1, s_2\}) \in (k, 2k)$
$\vdots$
$P\Big(\exists s_j$ unvisited vertex : η ακμή $\{s_{j-1}, s_j\}$ έχει χρονική ετικέτα $l(\{s_{j-1}, s_j\}) \in \big((j-1)k, jk\big)$

Για την τυχαία i-οστή από τις παραπάνω πιθανότητες, ισχύει:

$$
\begin{aligned}
&P\Big(\exists s_i \text{ unvisited vertex}: \text{ η ακμή } \{s_{i-1}, s_i\} \text{ έχει } l(\{s_{i-1}, s_i\}) \in ((i-1)k, ik)\Big) \\
&= 1 - P\Big(\nexists s_i \text{ unvisited vertex}: \text{ η ακμή } \{s_{i-1}, s_i\} \text{ έχει } l(\{s_{i-1}, s_i\}) \in ((i-1)k, ik)\Big) \\
&= 1 - P\Big(\forall s_i \text{ unvisited vertices}: \text{ η ακμή } \{s_{i-1}, s_i\} \text{ έχει } l(\{s_{i-1}, s_i\}) \notin ((i-1)k, ik)\Big) \\
&= 1 - \Big(P\big(\text{η ακμή } \{s_{i-1}, s_i\} \text{ έχει } l(\{s_{i-1}, s_i\}) \notin ((i-1)k, ik), s_i \text{ unvisited vertex}\big)\Big)^{n-i} \\
&= 1 - \Big(1 - P\big(\text{η ακμή } \{s_{i-1}, s_i\} \text{ έχει } l(\{s_{i-1}, s_i\}) \in ((i-1)k, ik), s_i \text{ unvisited vertex}\big)\Big)^{n-i} \\
&= 1 - \Big(1 - \frac{k}{n}\Big)^{n-i}
\end{aligned}
$$

Έτσι, η πιθανότητα να συμβεί το ε_1^j είναι:

$$
\begin{aligned}
P(\varepsilon_1^j) = &\left(1 - \Big(1 - \frac{k}{n}\Big)^{n-1}\right) \cdot \\
&\left(1 - \Big(1 - \frac{k}{n}\Big)^{n-2}\right) \cdot \ldots \cdot \left(1 - \Big(1 - \frac{k}{n}\Big)^{n-j}\right) \geq \\
&\geq \left(1 - \Big(1 - \frac{k}{n}\Big)^{n-j}\right)^j \geq \\
&\geq \left(1 - e^{-k}\Big(1 - \frac{k}{n}\Big)^{-j}\right)^j
\end{aligned}
$$

Για $j \leq c_1\sqrt{n}logn$, είναι:

$$\begin{aligned} \Big(1-\frac{k}{n}\Big)^{-j} &\leq \Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn} \quad \Leftrightarrow \\ \Leftrightarrow 1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-j} &\geq 1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn} \end{aligned}$$

και:

$$\left(1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-j}\right)^{j} \geq \left(1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn}\right)^{c_1\sqrt{n}logn}$$

Οπότε, για $j \leq c_1\sqrt{n}logn$, είναι:

$$P(\varepsilon_1^j) \geq \left(1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn}\right)^{c_1\sqrt{n}logn}$$

και επειδή στην τελευταία δύναμη η βάση $1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn}$ είναι μικρότερη του 1 και ο εκθέτης $c_1\sqrt{n}logn$ τουλάχιστον ίσος με 1, είναι:

$$P(\varepsilon_1^j) \geq 1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn}$$

Είναι:

$$\begin{aligned} c_1\sqrt{n}logn &\leq n \quad \Leftrightarrow \\ \Big(1-\frac{k}{n}\Big)^{c_1\sqrt{n}logn} &\geq \Big(1-\frac{k}{n}\Big)^{n} \quad \Leftrightarrow \\ \Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn} &\leq \Big(1-\frac{k}{n}\Big)^{-n} \quad \Leftrightarrow \\ 1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-c_1\sqrt{n}logn} &\geq 1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-n} \end{aligned}$$

Συνεπώς:

$$P(\varepsilon_1^j) \geq 1-e^{-k}\Big(1-\frac{k}{n}\Big)^{-n}$$

και αφού $k \geq 1$, είναι:

$$\begin{aligned} P(\varepsilon_1^j) &\geq 1-e^{-k}\Big(1-\frac{1}{n}\Big)^{-n} \\ &= 1-e^{-k}e \\ &= 1-e^{1-k} \end{aligned}$$

Για $k = rlogn,\ r > 1$, γίνεται:

$$\begin{aligned} P(\varepsilon_1^j) &\geq 1 - e^{1-rlogn} \\ &= 1 - en^{-r} \end{aligned}$$

Η πιθανότητα να αποτύχουμε σε κάθε δοκιμή $i = 0, \ldots, c_1\sqrt{n}logn$ να βρούμε s_i κορυφή τέτοια, ώστε $l(\{s_i, t\}) \in (c_1\sqrt{n}k, c_1\sqrt{n}k + \sqrt{n})$ είναι:

$$\begin{aligned} P(allfail) &= \overbrace{\Big(1 - \frac{1}{\sqrt{n}}\Big) \cdot \Big(1 - \frac{1}{\sqrt{n}}\Big) \cdot \ldots \Big(1 - \frac{1}{\sqrt{n}}\Big)}^{c_1\sqrt{n}logn \text{ το πλήθος παράγοντες}} \\ &= \Big(1 - \frac{1}{\sqrt{n}}\Big)^{c_1\sqrt{n}logn} \\ &= e^{-c_1logn} = n^{-c_1} \end{aligned}$$

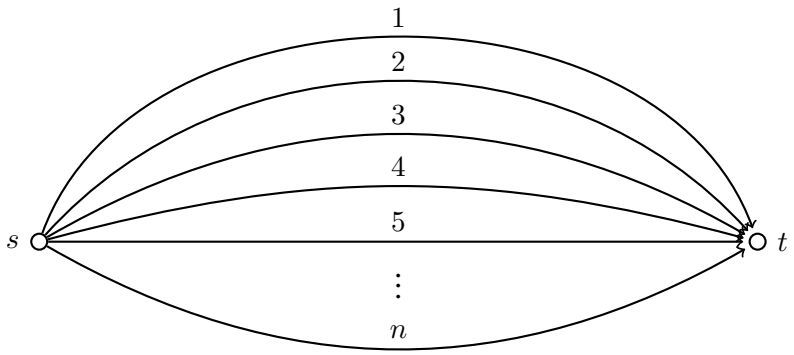
Η πιθανότητα να επιτύχουμε σε κάποια δοκιμή του αλγορίθμου είναι:

$$\begin{aligned} P(success) &= \Big(1 - P(allfail)\Big)P(\varepsilon_1^j) \\ &\geq \Big(1 - n^{-c_1}\Big)\Big(1 - en^{-r}\Big) \end{aligned}$$

Έτσι, με πιθανότητα ίση ή μεγαλύτερη από $\Big(1 - n^{-c_1}\Big)\Big(1 - en^{-r}\Big)$, επιτυγχάνουμε στιγμή άφιξης στην κορυφή t -ξεκινώντας από την s- το πολύ ίση με $c_1\sqrt{n}lognk + \sqrt{n}$, εκτελώντας διαδικασία που κοστίζει υπολογιστικό χρόνο το πολύ $\Theta(n\sqrt{n}logn)$. Το τελευταίο γίνεται εύκολα κατανοητό, αν σκεφτούμε ότι στη χειρότερη περίπτωση επιτυχίας, ο βασικός βρόγχος του αλγορίθμου εκτελείται $c_1\sqrt{n}logn + 1$ φορές, με την τελευταία επανάληψη να δίνει την επιθυμητή ακμή προς την κορυφή t. Σε κάθε μία από τις πρώτες $c_1\sqrt{n}logn$ επαναλήψεις, ελέγχει n το πολύ ακμές για να βρει αν υπάρχει ακμή με ετικέτα στο επιθυμητό διάστημα. Συνεπώς, στη χειρότερη περίπτωση επιτυχίας, ο αλγόριθμος χρειάζεται $\Theta(n\sqrt{n}logn)$ χρόνο για να βρει χρονικό μονοπάτι από την s στην t.

4.4 Ένα πρόβλημα βελτιστοποίησης: Το πρόβλημα των γεφυρών (The Bridges' problem)

Θα μελετήσουμε τώρα ένα πρόβλημα βελτιστοποίησης στο απλό χρονικό πολυγράφημα του Σχήματος 4.7.



Σχημα 4.7: Το πρόβλημα των γεφυρών (The bridges' problem)

Πρόβλημα

n το πλήθος άτομα βρίσκονται στη μία όχθη ενός ποταμού (βλ. κορυφή s, Σχήμα 4.7) και επιθυμούν να περάσουν στην απέναντι όχθη (βλ. κορυφή t, Σχήμα 4.7). Ο καθένας έχει τη δυνατότητα να διασχίσει μία από τις n γέφυρες που συνδέουν τις δύο όχθες του ποταμού, πληρώνοντας κόστος ίσο με $1 + \frac{i}{m}$, όπου i είναι ο αύξων αριθμός της γέφυρας και m το συνολικό πλήθος των ατόμων που θα περάσουν τη γέφυρα αυτή. Έτσι, το κόστος που πληρώνουν $\boldsymbol{m}$ άτομα για να διασχίσουν την ***i*-οστή** γέφυρα, $i = 1, 2, \ldots, n$, είναι:

$$cost[i] = m + i$$

Πώς πρέπει να μοιραστούν τα άτομα στις γέφυρες έτσι, ώστε να ελαχιστοποιείται το μέγιστο κόστος που συναντάται στις n γέφυρες;

4.4.1 Ο αλγόριθμος

Ισχυριζόμαστε ότι ο αλγόριθμος 9 επιλύει το πρόβλημα των γεφυρών.

Algorithm 9 The bridges problem solving algorithm

```
1:  procedure Bridges(n)
2:      cost[] is a 1×n array which holds the bridges' costs;
3:      content[] is a 1×n array which holds the bridges' contents;      ▷ a.k.a how
4:                                                                        many people
5:                                                                        are on each
6:                                                                        bridge
7:      m := n;                                          ▷ m is the number of bridges
8:      for i = 1 ... m do
9:          content[i] := 0;                                  ▷ Initializations
10:         cost[i] := i;
11:     end for
12:     for i = 1 ... n do
13:         bridge := 1;          ▷ Initialize the bridge that the i^th person will pass
14:         for  j = 2 ... m do ▷ Find the bridge that gives the minimum possible cost
15:             if  cost[j] < cost[bridge]  then
16:                 bridge := j;
17:             end if
18:         end for
19:         content[bridge] := content[bridge]+1;  ▷ Add the i^th person to the selected
20:                                                                   bridge's content
21:         cost[bridge] := cost[bridge]+1;          ▷ Calculate the right new cost
22:     end for
23:     for i = 1 ... m do
24:         if content[i] == 0 then
25:             cost[i] := 0;
26:         end if
27:         if content[i] == 1 then
28:             Write content[i] , " person passes bridge #", i , " who therefore has to
                pay cost equal to ", cost[i];
29:         else
30:             Write content[i] , " people pass bridge #", i , " who therefore have to
                pay cost equal to ", cost[i];
31:         end if                                        ▷ Print the bridges' costs
32:     end for
33: end procedure
```

Ο αλγόριθμος αναθέτει το τυχαίο i-οστό άτομο που θέλει να περάσει στην απέναντι όχθη στη γέφυρα εκείνη, η οποία αν της ανατεθεί ένα άτομο επιπλέον θα έχει κόστος, έστω a, μικρότερο από αυτό που θα είχε κάθε άλλη γέφυρα αν της ανατίθετο ένα άτομο επιπλέον. Αν υπάρχουν πολλές γέφυρες που μετά την προσθήκη ενός ατόμου θα είχαν κόστος a, ο αλγόριθμος αναθέτει το i-οστό άτομο σε εκείνη που έχει το μικρότερο αύξοντα αριθμό.

Εφαρμογή 1 Στους πίνακες που ακολουθούν φαίνονται οι καταστάσεις των γεφυρών και των διαφόρων δεικτών κατά την εκτέλεση του αλγορίθμου 9 για $n = 7$.

Βήμα 1^o		
#bridge	content	cost
1	0	1
2	0	2
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7

Βήμα 2^o		
#bridge	content	cost
1	1	2
2	0	2
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7

Βήμα 3^o		
#bridge	content	cost
1	2	3
2	0	2
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7

Βήμα 4^o		
#bridge	content	cost
1	2	3
2	1	3
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7

Βήμα 5^o		
#bridge	content	cost
1	3	4
2	1	3
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7

Βήμα 6^o		
#bridge	content	cost
1	3	4
2	2	4
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7

Βήμα 7^o		
#bridge	content	cost
1	3	4
2	2	4
3	1	4
4	0	4
5	0	5
6	0	6
7	0	7

Βήμα 8^o		
#bridge	content	cost
1	4	5
2	2	4
3	1	4
4	0	4
5	0	5
6	0	6
7	0	7

Στο σημείο αυτό, έχουν ολοκληρωθεί οι αναθέσεις των 7 ατόμων στις γέφυρες και μένει να γίνουν οι διορθώσεις κοστών στις γέφυρες με μηδενικό περιεχόμενο, όπως φαίνεται στον πίνακα του $9^{ου}$ βήματος.

Βήμα 9^o		
#bridge	content	cost
1	4	5
2	2	4
3	1	4
4	0	0
5	0	0
6	0	0
7	0	0

Εφαρμογή 2 Στους πίνακες που ακολουθούν φαίνονται οι καταστάσεις των γεφυρών και των διαφόρων δεικτών κατά την εκτέλεση του αλγορίθμου 9 για $n = 8$.

Προφανώς, τα βήματα στα οποία γίνονται οι αναθέσεις των πρώτων 7 ατόμων είναι ακριβώς ίδια με αυτά της Εφαρμογής 1, με τη διαφορά ότι υπάρχει επιπλέον μια 8^{η} γέφυρα, η οποία έχει συνεχώς μηδενικό περιεχόμενο.

Άρα, φθάνουμε στο $8^{ο}$ Βήμα και η κατάσταση των γεφυρών είναι αυτή που δείχνει ο ακόλουθος πίνακας:

Βήμα $8^{ο}$		
#bridge	content	cost
1	4	5
2	2	4
3	1	4
4	0	4
5	0	5
6	0	6
7	0	7
8	0	8

Έχουν ήδη ανατεθεί στις γέφυρες τα 7 πρώτα άτομα και στο επόμενο βήμα γίνεται η ανάθεση του $8^{ου}$ και τελευταίου ατόμου.

Βήμα $9^{ο}$		
#bridge	content	cost
1	4	5
2	3	5
3	1	4
4	0	4
5	0	5
6	0	6
7	0	7
8	0	8

Τέλος, γίνονται οι διορθώσεις κοστών στις γέφυρες με μηδενικό περιεχόμενο, όπως φαίνεται στον πίνακα του $10^{ου}$ βήματος.

Βήμα 10^o		
#bridge	content	cost
1	4	5
2	3	5
3	1	4
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0

Απόδειξη Ισχύος αλγορίθμου Θα αποδείξουμε την ισχύ του αλγορίθμου 9 με επαγωγή στο πλήθος n των γεφυρών και των ατόμων.

- Για $n = 1$, ο αλγόριθμος ορίζει το πλήθος των γεφυρών να είναι $m = 1$ και αρχικά θέτει μηδενικό περιεχόμενο και μοναδιαίο κόστος στη μοναδική γέφυρα. Στο βασικό βρόγχο του αλγορίθμου, το 1 άτομο ανατίθεται στη γέφυρα και το κόστος που πληρώνει γίνεται:

  $$cost[1] = 2$$

  Τέλος, τυπώνεται:

  "Η γέφυρα 1 έχει 1 άτομο που πληρώνει συνολικό κόστος 2"

  Άρα, πράγματι ο αλγόριθμος επιλύει το πρόβλημα για $n = 1$ άτομο.

- Έστω ότι ο αλγόριθμος επιλύει το πρόβλημα για $n = k$ άτομα.

- Θα δείξουμε ότι ο αλγόριθμος επιλύει το πρόβλημα για $n = k + 1$ άτομα.

  Προτού συνεχίσουμε, ας σκεφτούμε το εξής: Έστω $n_1, n_2 \in \mathbb{N}$, με $n_1 > n_2$. Είναι προφανές πως δεν μπορεί η λύση του προβλήματος για $n = n_1$ γέφυρες να είναι αριθμός μικρότερος της λύσης για $n = n_2$. Άρα, το ελάχιστο δυνατό μέγιστο κόστος για $n = n_1$ γέφυρες είναι τουλάχιστον όσο το ελάχιστο δυνατό μέγιστο κόστος για $n = n_2$ γέφυρες.

  Ας παρατηρήσουμε τώρα ότι οι διαδικασίες που εκτελεί ο αλγόριθμος στο βασικό βρόγχο για k άτομα, και τα αποτελέσματα που προκύπτουν μέσω αυτών, ταυτίζονται με αυτές που εκτελεί και τα αποτελέσματα που προκύπτουν για $k + 1$ άτομα, με τη διαφορά ότι στα $k + 1$ άτομα υπάρχει μια $(k + 1)$-οστή

γέφυρα, η οποία καθ' όλη την εκτέλεση των διαδικασιών αυτών έχει μηδενικό περιεχόμενο, και υπάρχει επίσης μια επιπλέον εκτέλεση του βρόγχου. Στην αρχή της $(k+1)$-οστής αυτής εκτέλεσης του βρόγχου, ο αλγόριθμος έχει αναθέσει τα k άτομα στις γέφυρες έτσι, ώστε να επιτυγχάνεται το ελάχιστο δυνατό μέγιστο κόστος.

Ο αλγόριθμος, από κατασκευής του, θέτει τα άτομα στις γέφυρες με τρόπο που ταξινομεί τα κόστη κατά (όχι αυστηρά) φθίνουσα σειρά και μάλιστα να συμβαίνει ένα από τα δύο ακόλουθα ενδεχόμενα:

$$\begin{cases} \text{όλες οι γέφυρες έχουν ίδιο κόστος, έστω } a \\ \text{ή} \\ \text{κάποιες γέφυρες έχουν κόστος } a \text{ και κάποιες έχουν κόστος } a-1. \end{cases}$$

Στη 2^η περίπτωση, όταν φθάνει το $(k+1)$-οστό άτομο, είναι προφανές ότι ο αλγόριθμος θα το αναθέσει στην 1^η κατά σειρά γέφυρα που έχει κόστος ίσο με $a-1$, διατηρώντας έτσι το μέγιστο κόστος που συναντάται στις γέφυρες στο ελάχιστο δυνατό, δηλαδή a.

Στην 1^η περίπτωση, αν $r \leq k+1$ ο αύξων αριθμός της τελευταίας γέφυρας που έχει θετικό περιεχόμενο, $content[r]$, τότε είναι:

$$\left.\begin{array}{l} r + content[r] = a \\ \text{Όμως: } content[r] \geq 1 \text{ και επομένως: } r + content[r] \geq r+1 \end{array}\right\} \Rightarrow a \geq r+1$$

Επίσης, αφού η $(r+1)$-οστή γέφυρα έχει μηδενικό περιεχόμενο, είναι:

$$cost[r+1] = r+1$$

Ο αλγόριθμος ελέγχει ποια από τις $k+1$ γέφυρες έχει το ελάχιστο κόστος για να αναθέσει σε αυτή το $(k+1)$-οστό άτομο. Αν $a = r+1$, τότε ο αλγόριθμος αναθέτει το τελευταίο άτομο στην 1^η γέφυρα. Διαφορετικά, το αναθέτει στη $(r+1)$-οστή γέφυρα. Εξασφαλίζει, έτσι, το ελάχιστο δυνατό μέγιστο κόστος που συναντάται στις $k+1$ γέφυρες.

Άρα, ο αλγόριθμος επιλύει το πρόβλημα για $n = k+1$ άτομα.

Κεφάλαιο 5

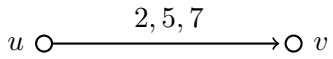
Σύνοψη

5.1 Αποτελέσματα

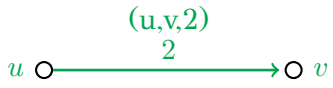
Κλείνοντας, θα κάνουμε μια ανασκόπηση της εργασίας. Τι είναι αυτό που πρέπει να θυμόμαστε; Τι ορίσαμε, τι μελετήσαμε και πού καταλήξαμε;

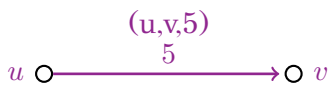
Γνωρίσαμε τα χρονικά γραφήματα, ένα νέο είδος γραφημάτων που φαίνεται να έχει ποικίλες εφαρμογές, πολλές απ' τις οποίες πιθανότατα ακόμη δεν έχουν ξεδιπλωθεί ολοκληρωτικά μπροστά μας. Ορίσαμε νέες έννοες, όπως:

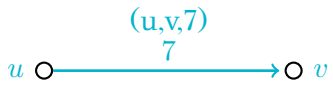
- οι χρονικές ακμές



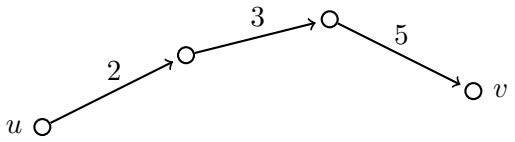
Η ακμή $\{u, v\}$, στην οποία ανατίθενται ετικέτες
2, 5 και 7 δίνει τις χρονικές ακμές:





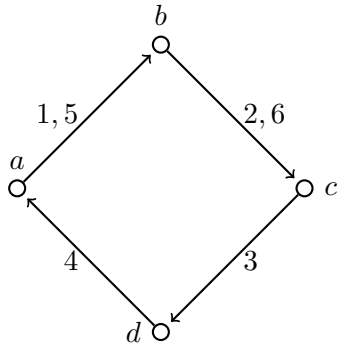


- τα ταξίδια ή χρονικά μονοπάτια στα χρονικά γραφήματα



Ένα (u, v)-ταξίδι με χρόνο άφιξης ίσο με 5

- ο αριθμός χρονικότητας ενός γραφήματος
- η ηλικία ενός χρονικού γραφήματος



Ένα γράφημα με αριθμό χρονικότητας ίσο με 2
και ηλικία ίση με 6

- η χρονική διάμετρος, κ.ά.

Κατασκευάσαμε αλγορίθμους εύρεσης πρωτίστων ταξιδίων σε διαφόρους τύπους χρονικών γραφημάτων, καθώς επίσης μελετήσαμε και αποδείξαμε την πολυπλοκότητα προβλημάτων που έχουν σχέση με τα χρονικά γραφήματα (βλ. $Reachability\&Age$ και $Bridges'\ problem$).

Αυτό που πρέπει να έχουμε στο νου μας είναι πως σε ένα χρονικό γράφημα, μια ακμή συμβολίζει μια *κοστοβόρα* απευθείας σύνδεση δύο "αντικειμένων". Πρέπει, λοιπόν, να περιορίσουμε τις στιγμές, κατά τις οποίες η σύνδεση αυτή υφίσταται. Μέσα σε ένα δίκτυο τέτοιων συνδέσεων, μας ενδιαφέρει:

- να ελαχιστοποιήσουμε τις στιγμές που υφίσταται η κάθε σύνδεση (πλήθος ετικετών / ακμή)
- να ελαχιστοποιήσουμε συνολικά τις στιγμές που υφίστανται οι διάφορες συνδέσεις (συνολικό πλήθος ετικετών)

- να ελαχιστοποιήσουμε την τελευταία στιγμή, κατά την οποία κάποια σύνδεση είναι υπαρκτή (ηλικία)
- κ.ά.

5.2 Τροφή για μελλοντικές μελέτες

Με την παρούσα εργασία αγγίζουμε ένα νέο αντικείμενο μελέτης στον κλάδο της Θεωρίας Γραφημάτων που ανοίγει δρόμους για εκτενή έρευνα, καθώς πολλά είναι εκείνα που μένουν να εξεταστούν και να μελετηθούν γύρω από τα Χρονικά Γραφήματα.

Για αρχή, θα μπορούσαν ίσως να επιτευχθούν καλύτεροι χρόνοι σε κάποιους από τους αλγορίθμους που κατασκευάστηκαν και παρουσιάστηκαν στην παρούσα, καθώς επίσης να βρεθούν καλύτερα άνω ή/και κάτω όρια για τις χρονικές διαμέτρους των γραφημάτων που μελετήσαμε στο κεφάλαιο 4, αλλά και άλλων γνωστών γραφημάτων.

Αξίζει να σημειωθεί ξανά ότι στο κεφάλαιο 4 έγινε επίσης μια νύξη για μελλοντική μελέτη πάνω σε γραφήματα που ικανοποιούν την ΥΠΟΘΕΣΗ-F (βλ. ενότητα 4.1), δηλαδή γραφήματα, οι ακμές των οποίων δύνανται να πάρουν περισσότερες από μία χρονικές ετικέτες διαθεσιμότητας και οι ετικέτες αυτές επιλέγονται τυχαία, ανεξάρτητα η μία από την άλλη από ένα σύνολο L_0=$\{1, 2, \ldots, a\}$, όπου $a \in \mathbb{N}$, με την επιλογή των ετικετών διαθεσιμότητας μιας ακμής να ακολουθεί μια κατανομή F.

Σημαντική μελέτη θα μπορούσε να γίνει πάνω στους αριθμούς χρονικότητας διαφόρων χρονικών γραφημάτων και μάλιστα να εξεταστεί κατά πόσο είναι εύκολο να βρούμε τον αριθμό χρονικότητας δεδομένου γραφήματος.Είναι, δηλαδή, το πρόβλημα της εύρεσης του αριθμού χρονικότητας τυχαίου γραφήματος ένα πρόβλημα που επιλύεται πολυωνυμικά ή όχι και αν όχι, ποια η κλάση πολυπλοκότητας στην οποία ανήκει;

Επίσης, ποια η συμπεριφορά του αριθμού χρονικότητας ενός γραφήματος, όταν περιορίσουμε την ηλικία που επιθυμούμε να έχει το γράφημα; Για παράδειγμα, αν δε θέλουμε η ηλικία δεδομένου γραφήματος να ξεπερνά τον αριθμό 5, ποιος είναι ο αριθμός χρονικότητας που έχει το γράφημα;

Ακόμη, προκύπτουν άλλα προβλήματα που παρουσιάζουν ενδιαφέρον ως προς την πολυπλοκότητα επίλυσής τους. Θα κλείσουμε με μια αναφορά σε ένα από αυτά, το λεγόμενο ***πρόβλημα ALL-PATHS&AGE***.

Πρόβλημα All-Paths&Age

Δεδομένου γραφήματος G και θετικού ακεραίου k, υπάρχει ανάθεση L στις ακμές του G που διατηρεί όλα τα απλά μονοπάτια του G και δίνει $age(G, L) \leq k$;

Το πρόβλημα αυτό έχει μια ομοιότητα με το γνωστό μας πλέον πρόβλημα *Reachability&Age*, αλλά η δυσκολία επίλυσής του έγκειται στο ότι ζητείται η ανάθεση να διατηρεί όλα τα απλά μονοπάτια του G. Δηλαδή, $\forall u, v \in V(G)$, αν p ένα (u, v)-μονοπάτι στο G, τότε στο (G, L) υπάρχει (u, v)-ταξίδι πάνω στις ακμές του p.

Αυτά και πολλά ακόμη παραμένουν ανοικτά θέματα γύρω από τα Χρονικά Γραφήματα. Μελλοντικά, σκοπός μας είναι να τα μελετήσουμε και να επεκτείνουμε τις σχετικές μας γνώσεις, ελπίζοντας -γιατί όχι;- πως θα αποδειχθούν ενδιαφέροντα και για άλλους.

Βιβλιογραφία

[1] Othon Michail, Ioannis Chatzigiannakis, Paul G. Spirakis *Causality, Influence, and Computation in Possibly Disconnected Synchronous Dynamic Networks.* OPODIS, volume 7702 of LNCS, pages 269-283. Springer, 2012

[2] Noga Alon, Joel H.Spencer and Paul Erdős *The Probabilistic Method* 1992.

[3] William Feller *An Introduction to Probability Theory and Its Applications, Vol. 1, 3rd Edition.*

[4] George Mertzios, Othon Michail, Ioannis Chatzigiannakis and Paul G. Spirakis. *Temporal Network Optimization Subject to Connectivity Constraints.* ICALP 2013, July 8-14. Springer LNCS ARCoSS Proceedings.

Παράρτημα

Ακολούθως παρουσιάζεται ο κώδικας σε C++ του αλγορίθμου επίλυσης του προβλήματος των γεφυρών (βλ. 4.4), καθώς και τα αποτελέσματα της εκτέλεσής του για διάφορες τιμές του πλήθους των ατόμων, n.

```
#include <iostream>
#include <vector>
#include <math.h>

using namespace std;

int main(){

   int i,j,n,m,bridge;

cout«"Give number of persons, n\n";
cin»n;
m=n;

int *content = new int[n];
int *cost = new int[n];

for (i=1;i<=m;i++){
content[i]=0;
cost[i]=i;
}

for (i=1;i<=n;i++){
bridge=1;
for (j=2;j<=m;j++){
if (cost[j]<cost[bridge]){
bridge=j;
}
```

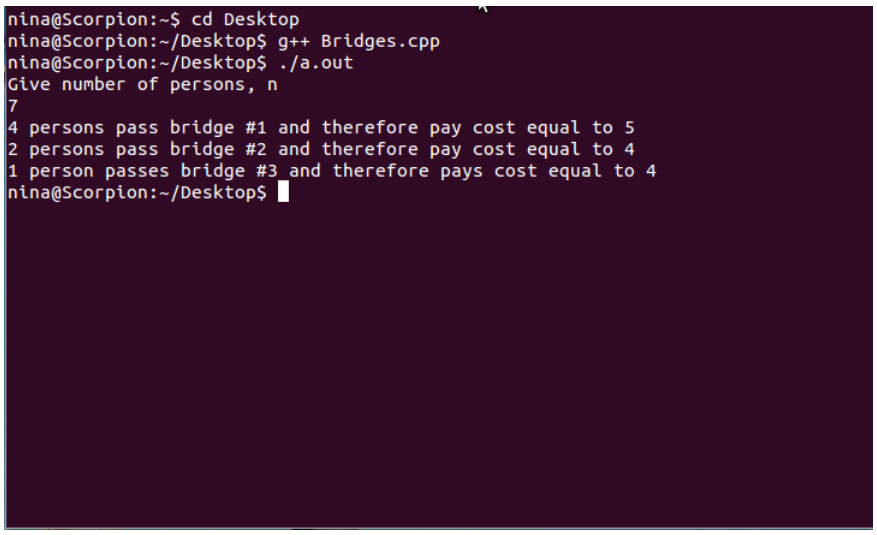
```
}
content[bridge]=content[bridge]+1;
cost[bridge]=cost[bridge]+1;
}

for (i=1;i<=m;i++){
if (content[i]==0)
cost[i]=0;
else if (content[i]==1)
cout«content[i]« " person passes bridge #" «i«" and therefore pays cost equal to "«cost[i]«"\n";
else
cout«content[i]« " persons pass bridge #"«i«" and therefore pay cost equal to "«cost[i]«"\n";
}

}
```

Εκτελέσεις

1. Για 7 άτομα:



2. Για 10 άτομα:

```
nina@Scorpion:~$ cd Desktop
nina@Scorpion:~/Desktop$ g++ Bridges.cpp
nina@Scorpion:~/Desktop$ ./a.out
Give number of persons, n
10
4 persons pass bridge #1 and therefore pay cost equal to 5
3 persons pass bridge #2 and therefore pay cost equal to 5
2 persons pass bridge #3 and therefore pay cost equal to 5
1 person passes bridge #4 and therefore pays cost equal to 5
nina@Scorpion:~/Desktop$
```

3. Για 15 άτομα:

```
nina@Scorpion:~$ cd Desktop
nina@Scorpion:~/Desktop$ g++ Bridges.cpp
nina@Scorpion:~/Desktop$ ./a.out
Give number of persons, n
15
5 persons pass bridge #1 and therefore pay cost equal to 6
4 persons pass bridge #2 and therefore pay cost equal to 6
3 persons pass bridge #3 and therefore pay cost equal to 6
2 persons pass bridge #4 and therefore pay cost equal to 6
1 person passes bridge #5 and therefore pays cost equal to 6
nina@Scorpion:~/Desktop$
```

4. Για 88 άτομα:

```
nina@Scorpion:~$ cd Desktop
nina@Scorpion:~/Desktop$ g++ Bridges.cpp
nina@Scorpion:~/Desktop$ ./a.out
Give number of persons, n
88
13 persons pass bridge #1 and therefore pay cost equal to 14
12 persons pass bridge #2 and therefore pay cost equal to 14
11 persons pass bridge #3 and therefore pay cost equal to 14
10 persons pass bridge #4 and therefore pay cost equal to 14
9 persons pass bridge #5 and therefore pay cost equal to 14
8 persons pass bridge #6 and therefore pay cost equal to 14
7 persons pass bridge #7 and therefore pay cost equal to 14
6 persons pass bridge #8 and therefore pay cost equal to 14
5 persons pass bridge #9 and therefore pay cost equal to 14
4 persons pass bridge #10 and therefore pay cost equal to 14
2 persons pass bridge #11 and therefore pay cost equal to 13
1 person passes bridge #12 and therefore pays cost equal to 13
nina@Scorpion:~/Desktop$
```