COMP108 Algorithmic Foundations — Tutorial 11

w/c 8th May 2017

Name: ____

Hand this in to the demonstrator at the end of the tutorial even if you haven't finished it. You will get feedback next week. Tutorial participation contributes to 5% of overall marks.

1. [**Do this before tutorial**] Consider the Knapsack problem with a knapsack of capacity 10. Suppose we have four items I_1, I_2, I_3, I_4. The following table lists the value and weight of each item.

Item	Weight	Value
I_1	2	20
I_2	4	30
I_3	6	35
I_4	8	40

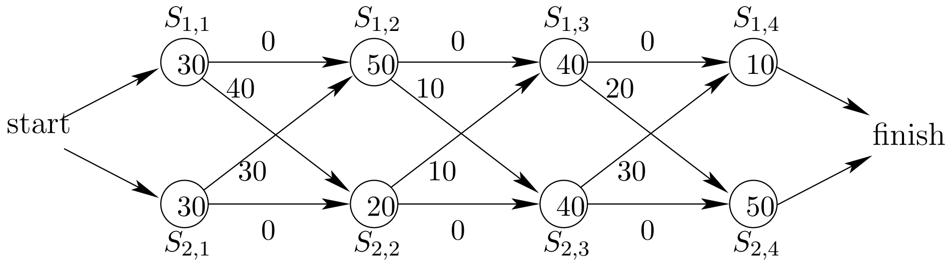
   (a) Fill in the following table to find the value and weight of all possible subsets of items.

Subset	Weight	Value	Subset	Weight	Value
$\{I_1\}$					
$\{I_2\}$					
$\{I_3\}$					
$\{I_4\}$					
$\{I_1, I_2\}$					

   (b) Which is the subset with maximum value such that the weight does not exceed the knapsack capacity?

   (c) Consider the following greedy method. *Start from the item with the largest value, select the item if adding this item to the selected set does not exceed the knapsack capacity.* Which subset of items is found by the above greedy method? Is the subset found the best solution?

2. **[Do this during tutorial]** Suppose there are two assembly lines each with 4 stations, $S_{i,j}$. The assembly time is given in the circle representing the station and the transfer time is given next to the arrow from one station to another.



(a) Using dynamic programming, fill in the table of the minimum time $f_i[j]$ needed to get through station $S_{i,j}$. You should also show **all** the intermediate steps in computing these values, e.g., in computing $f_1[2]$, you need to specify that $f_1[2] = \min\{\_\_, \_\_\}$.

Intermediate steps:

j	$f_1[j]$	$f_2[j]$
1		
2		
3		
4		

(b) What is the minimum time f^* needed to get through the assembly line?

(c) Which stations should be chosen to achieve the minimum time?

3. [**Do this during tutorial**] Consider the following recurrence.

$$T(n) = \begin{cases} 1 & \text{if } n == 0 \text{ or } n == 1 \\ 2 & \text{if } n == 2 \\ T(n-1) + T(n-3) & \text{if } n > 2 \end{cases}$$

(a) Design and write a pseudo code for a recursive procedure to compute $T(n)$.

(b) Draw the execution tree for $T(7)$.

(c) Using the concept of dynamic programming, rewrite your recursive procedure into a non-recursive one.

4. [**Puzzle**] Forty-five Minutes: How do we measure forty-five minutes using two identical wires, each of which takes an hour to burn, but the wires burn **non-uniformly**. You can use as many matchsticks as you like.