Correctness

Algorithm SGREEDY-ACTIVITY-SELECTOR produces solutions of maximum size for the activity-selection problem.

We use INDUCTION to prove that

> "We can find an optimal selection if we are given n activities".

Remember that, after the first step of SGREEDY-ACTIVITY-SELECTOR, activities are in order of finish time. We thus assume w.l.o.g. that activity A has the earliest finish time.

Simple case: there is only one activity ... we win (there is nothing to choose)!!

Complicated case: there's many activities.

Induction tells us we need to prove that we can find an optimal selection if we are given n activities assuming we can solve the same problem but with only $n-1$ activities.

(1) We apply SGREEDY-ACTVITY-SELECTOR and select A as the first activity. We argue that there must be an optimal solution whose first activity is A.

> Suppose $X \subseteq \mathcal{S}$ is an optimal solution (we write the activities in X by increasing finishing time). Suppose the first activity in X is A'. If $A = A'$ we are done, otherwise we could replace A' with A and obtain another solution with the same number of activities as X.

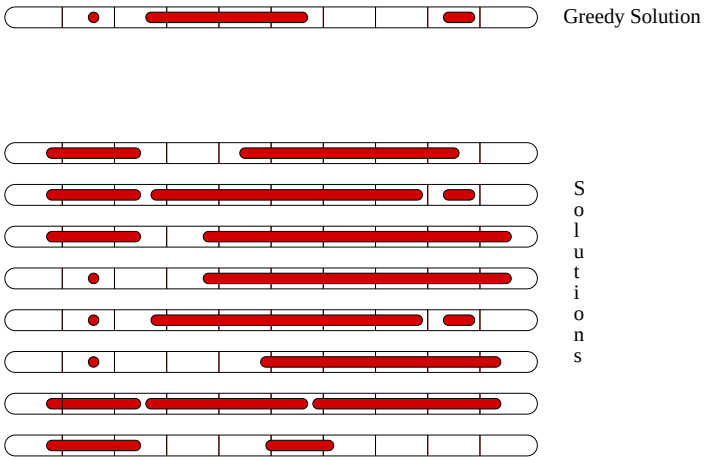
(2) Now we are left with $\mathcal{S} \setminus \{A\}$ (in fact we can discard all activities starting strictly before the end of A). This is a set of (at most) $n-1$ activities: we know how to solve the given problem if we only have $n-1$ activities!! So let Y be the set of activities chosen from $\mathcal{S} \setminus \{A\}$. The set of activities selected from $\mathcal{S}$ will be:

$$\{A\} \cup Y$$

AND THIS IS AN OPTIMAL CHOICE! (i.e. combining A with an optimal solution for the problem involving the "remaining" activities we get an optimal solution for the whole problem).

> Suppose that the best solution is $\{A\} \cup Z$. But then $|Z| > |Y|$. So Y would not be optimal solution for the "remaining" activities!.

Why does it all work?



Fundamental Properties

Greedy-choice an optimal solution can be found by making locally optimal greedy choices at each step.

Optimal substructure an optimal solution contains within it optimal solutions to subproblems.

Some (simple) greedy algorithm will always work as long as the solution sets for a given problem have the "greedy-choice" and "optimal-substructure" properties.

Writing down a good program to solve the given problem is less important than studying the combinatorial properties of its solutions.

Another example

i	A_1	A_2	A_3	A_4	A_5	A_6	A_7
s_i	1	1	5	8	7	11	3
f_i	2	4	6	9	10	12	13

(to speed things up activities are already sorted by non-decreasing finish time).

Detailed simulation

instructions processed	j	X	i
first two instructions before **for** *loop*	1	$\{A_1\}$	
$i \leftarrow 2$			2
$(s_2 = 1, f_1 = 2, s_2 \not\geq f_1)\ i \leftarrow i + 1$			3
$(s_3 = 5, f_1 = 2, s_3 \geq f_1)\ X \leftarrow X \cup \{A_3\}$		$\{A_1, A_3\}$	
$j \leftarrow i$	3		
$i \leftarrow i + 1$			4
$(s_4 = 8, f_3 = 6, s_4 \geq f_3)\ X \leftarrow X \cup \{A_4\}$		$\{A_1, A_3, A_4\}$	
$j \leftarrow i$	4		
$i \leftarrow i + 1$			5
$(s_5 = 7, f_4 = 9, s_5 \not\geq f_4)\ i \leftarrow i + 1$			6
$(s_6 = 11, f_4 = 9, s_6 \geq f_4)\ X \leftarrow X \cup \{A_6\}$		$\{A_1, A_3, A_4, A_6\}$	
$j \leftarrow i$	6		
$i \leftarrow i + 1$			7

Exercise: Timetabling

Lecture rooms: $R_1, R_2, \ldots, R_k$, each room is available for the same x hours. No room can be booked by two lecturers as the same time.

Lectures: $C_1, C_2, \ldots, C_l$, each taking one hour.

Attendability constraints: a set of pairs (C_i, C_j) meaning that lecture C_i cannot take place at the same time as lecture C_j.

Avoidability constraints: for each lecture C_i, a set of times $T_i = \{t : t \in \{1, \ldots, x\}\}$ such that C_i should not take place at time $t \in T_i$.

Goal. Assign lectures to lecture rooms, satisfying as many attendability constraints and avoidability constraints as possible.

Questions. can you find a greedy algorithm that returns *some* solution? Can you prove/disprove this algorithm gives an optimal solution?

A correct solution (a working program or pseudo-code) for any of the problems I am suggesting is worth $\frac{1}{4}$ of the coursework mark.

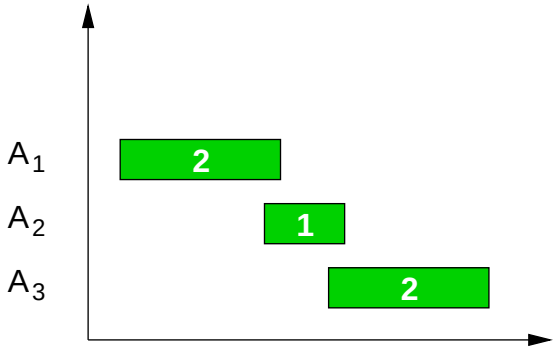
Greedy: is it all as simple as that?

Weighted Activity-selection problem. We are given a set of proposed activities $\mathcal{S} = \{A_1, A_2, \ldots, A_n\}$ that wish to use a resource, which can be used by only one activity at a time. Each activity is defined by a *start time* s_i, a *finish time* f_i, with $0 \leq s_i < f_i < +\infty$, and a positive integer value w_i. If selected, activity A_i takes place during the time interval $[s_i, f_i)$. Two activities A_i and A_j are *compatible* if $s_i \geq f_j$ or $s_j \geq f_i$. The *weighted* activity-selection problem is to select a number of mutually compatible activities of the largest possible total weight.

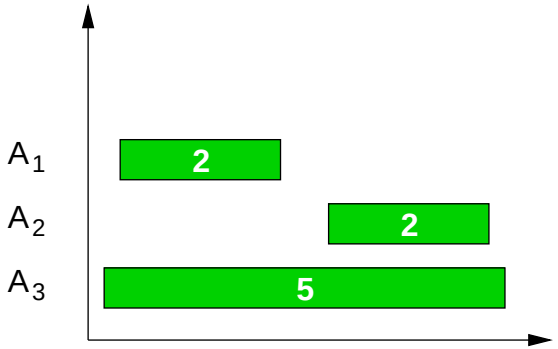
The problem makes perfect sense (it is even more realistic than the original problem). Perhaps the resource manager wants to maximise his profit (take v_i as the profit gained from running A_i).

How would you solve the problem?

Example



One more example



... and again