

Last time we may solve the scheduling of activities for a conference room.

Objective: maximize the # of activities scheduled to a conf. room (that are compatible).

Given S = start time of availability of room
list of activities w/ (s_i, f_i)
start time finish time

example:

$(1,4), (3,6), (1,7), (5,8), (3,9), (5,10), (6,11), (8,12)$
= 8 activ.

$S = 1$

Last time we did the dyn. prog. solution but an easier Greedy solution exists:

Schedule $(S, \text{activ. list})$:

1. $\text{maxAct} \in \emptyset$
2. sort activ. list by finish times (low to high)
3. for $i \leftarrow 1$ to n
4. if $\text{act list}[i].s \geq S$
5. sched. $\text{act list}[i]$
6. $\text{maxAct}++$
7. $S \leftarrow \text{act list}[i].f$
8. return maxAct

Running time? $\Theta(n \lg n)$ for sort + $\Theta(n)$

$\Rightarrow \Theta(n \lg n)$.

Will the greedy sol. always work?

Defn Greedy Choice = take the earliest finish time that is still compat. w/ the first set of greedy choices.
that is always give us MAX ACTS.

Proof by induction:

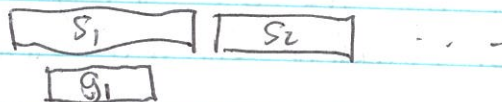
BASE CASE: We can always start w/ greedy choice.

Let S be any optimal sol. (that is it has MAX # of compat. acts.)
ordered by finish time.

$\langle S_1, S_2, \dots, S_t \rangle$

S_1 can be replaced by the greedy choice

Notice



g_1 can replace S_1 (g_1 is the 1st greedy choice, i.e. it ends the earliest)
so g_1 can replace S_1 which ends later.

Ind. hypothesis Assume S is an optimal sol. that begins w/ k greedy choices.

$S = \langle g_1, g_2, g_3, \dots, g_k, S_{k+1}, \dots, S_t \rangle$



can replace S_{k+1} w/ g_{k+1} b/c by defn it is compatible w/ g_k & it ends earlier than S_{k+1}

New topic.

Binary Min Heaps. What are they?

a binary tree w/ parent \leq children at every node

no relationship between L & R children.

also always complete. (balanced).

Binary Min Heap is a common implementation of a PQ which needs these operations:

1. Extract Min (get & remove the min)
2. Insert
3. Decrease Key

Compare a Binary Min Heap to a BST.

a heap has lighter restrictions on ordering than a BST. We give up search but have easy balancing.