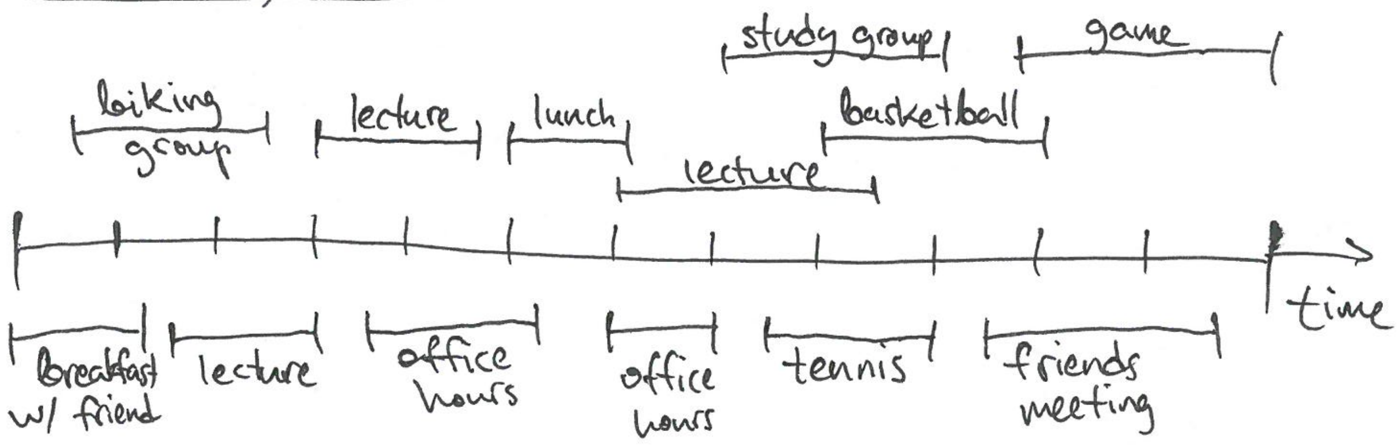
Planning Your Time

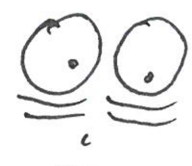


Input Activities $i = 1 \dots n$; each has a start time s_i & a finish time f_i.

A schedule Subset of non-overlapping activities $i_1 \dots i_k$

$$s_{i_1} < f_{i_1} \le s_{i_2} < f_{i_2} \dots < f_{i_k}$$

(A) "I want to do as many activities as possible!" (i.e., max k).



People often use greedy strategies:
pick an activity that looks "best" now, then "deal with the consequences" move on to next. E.g.

* Pick shortest activity (i.e., min $f_i - s_i$)
* Pick most urgent activity (i.e., min s_i)
* Pick earliest ~~deadline~~ finish (i.e., min f_i)

<u>Advantage</u> Easy to implement, efficient to find next activity.

Run-time for the three algorithms we suggested: $O(n \lg n)$

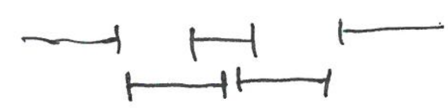
1 sort the activities ((i) according to $f_i - s_i$ or $\frac{s_i}{f_i}$)
(ii) sort $s_1 \ldots s_n, f_1 \ldots f_n$

2 pick min activity, rule out overlapping activities.
(i.e, start time between start & finish of activity)

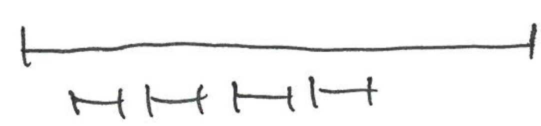
Step 1 takes $O(n \lg n)$ time. Step 2 takes $O(n)$ time overall. *

<u>Disadvantage</u> May not find optimal solution.

* Shortest activity - one short activity may rule out two activities

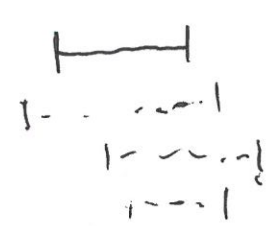


* Most urgent activity - urgent activity may rule out many activities that start later

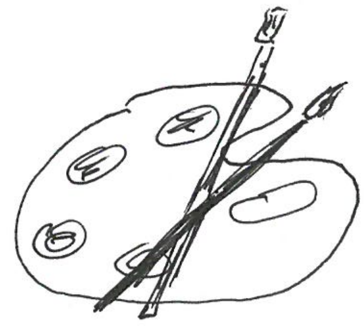


* Earliest finish time - works for previous examples, and in fact works in general as we'll see later in the term.

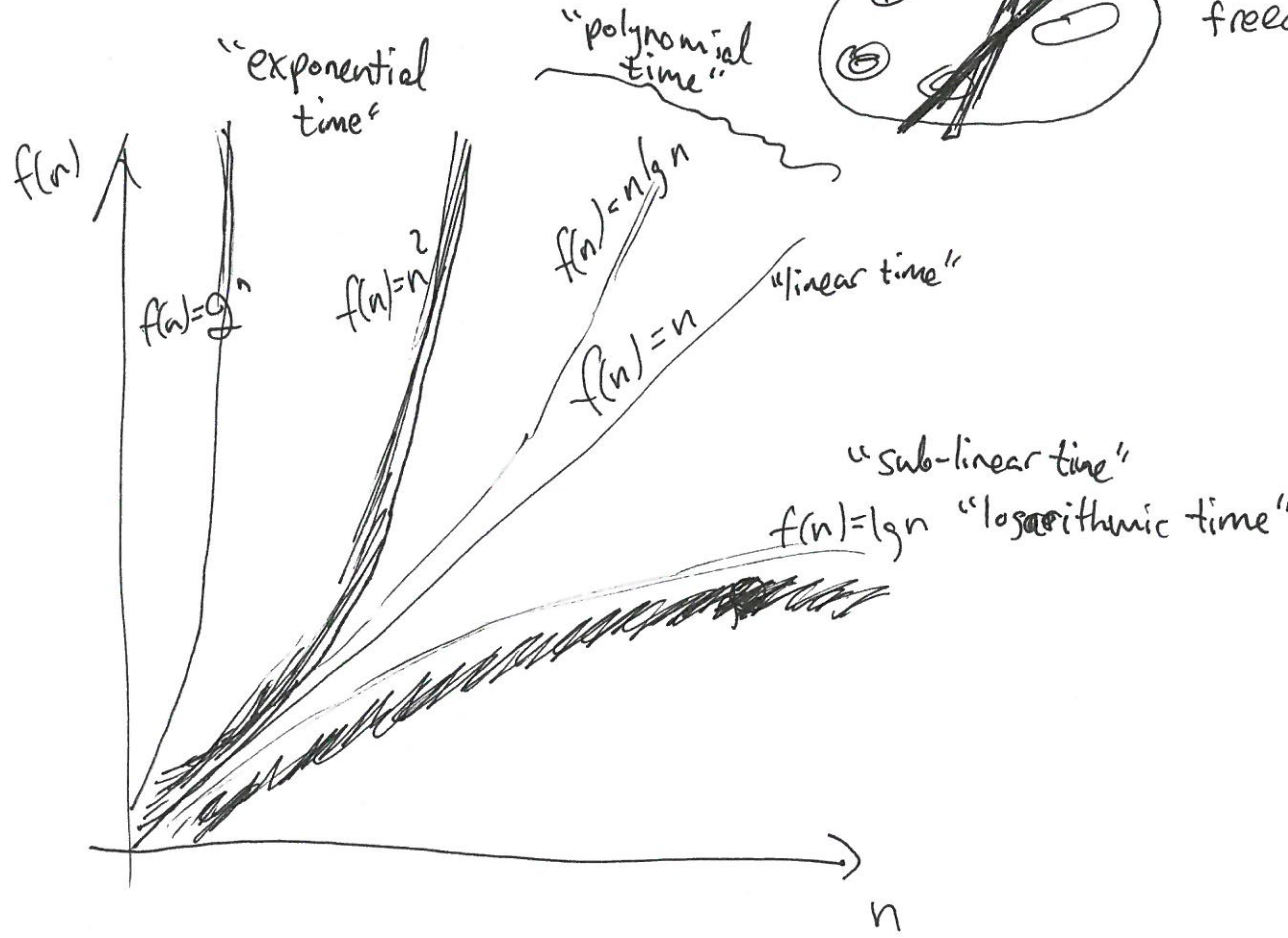
Intuition: put yourself ahead of the game for the next activities.



* Run-times



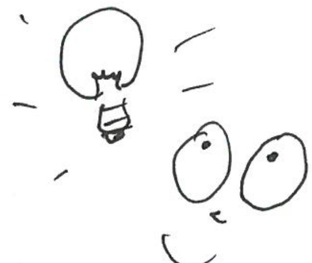
with some artistic freedom...



* We use "O" notation : $f(n) = O(g(n))$ if $\exists m, n_0\ \forall n \geq n_0\ f(n) \leq m \cdot g(n)$

Rationale : one operation can count as two (or more) in a different computational model. We want to count # operations while ignoring such differences.

B "I have priorities"

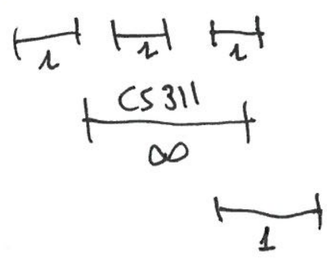


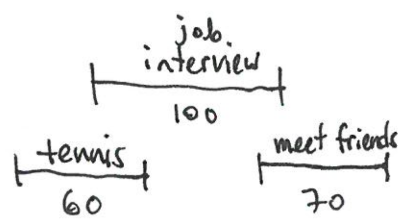
Input Activities as before but now every activity has a weight w_i

Goal $\max \sum_{j=1}^{k} w_{i_j}$ for a schedule $i_1 - i_k$.

Note The no-weights case is a special case with $w_i = 1 \ \forall i$.

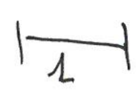
Examples





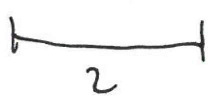
Observation The greedy approach fails!

What's better ??



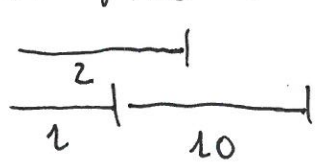
get less, free fast

OR

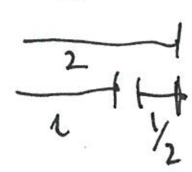


get more, free later

Depends... Are there exciting oppurtunities ahead?

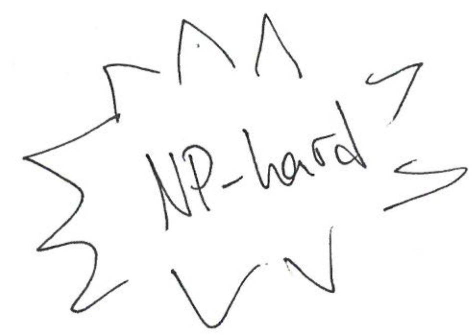


OR



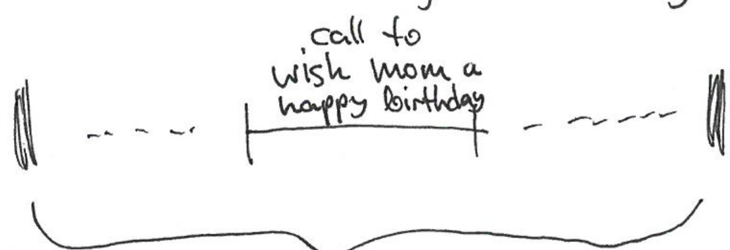
Later in the semester we'll discuss "dynamic programming", a powerful algorithmic paradigm that records information about different alternatives.

<u>Note</u> There can be up to 2^n different schedules. Going through them one by one — using "brute force" — would be extraordinarily inefficient. In contrast — Dynamic programming gives an $O(n \lg n)$ time algo for the problem!!



The most efficient algo we know takes exponential time!!

Ⓒ "I'm flexible in my scheduling"



need half an hour any time during this period.

<u>Input</u> n activities; each has a start time s_i, a finish time f_i and a duration ℓ_i.

<u>Goal</u> Maximize k with $i_1 \sim i_k$ & s.t. $s_{i_j} \le s'_{i_j} < f'_{i_j} \le f_{i_j}$

$s'_{i_1} \le f'_{i_1} \le \cdots \le s'_{i_k} \le f'_{i_k}$