Greedy Algorithms

Making change

$3,47 => 1 × $2
1 × $1
1 × 25¢
2 × 10¢
2 × 1¢
7 coins

Claim: This is the minimum # of coins

Exercise: Prove greedy choice works for Canadian Coin System

Greedy Alg: always choose largest coin possible

Does it work for all coin systems?

1¢ 6¢ 7¢

Make change 12¢

- greedy: 7 1 1 1 1 1 (5 * 1¢) => 6 coins
- better: 2 × 6¢ => 2 coins

Claim: Greedy change alg can be implemented in polynomial time using quotients & remainders

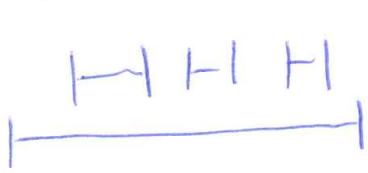
Interval Scheduling or "Activity Selection"

Given set of activities (intervals)
- start time, finish time, length

Select a maximum set of disjoint intervals

Greedy Approach
- pick one activity greedily
- remove conflicts
- repeat

(1) Select activity that starts earliest.

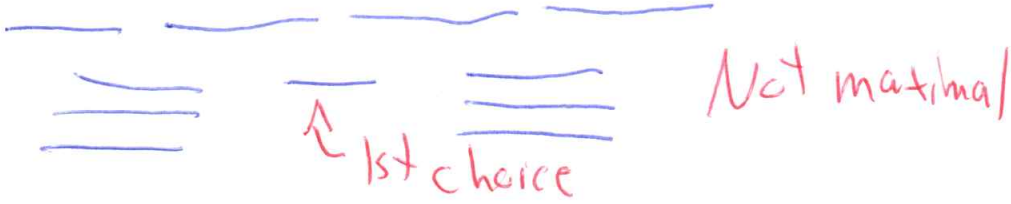


Not maximal

(2) Select shortest interval

Not maximal

(3) Select interval with fewest conflicts



Not maximal

(4) Select interval that ends earliest

Sort activities 1..n by end time

$A = \emptyset$

for $i = 1$ to n

  if activity i does not overlap with any activities in A then

    $A = A \cup \{i\}$

only needs to check last one

Analysis: $O(n \log n)$ to sort

$O(n)$ loop

$\Rightarrow O(n \log n)$

Correctness: 2 approaches

1. Greedy stays ahead all the time
2. "Exchange" proofs

Interval Scheduling - Correctness

① Suppose greedy alg returns: $a_1, a_2, \ldots, a_K$
- sorted by end time

Suppose an optimal solution is: $b_1, b_2, \ldots b_K b_{K+1}, \ldots, b_\ell$
- sorted by end time

Claim: $a_1, b_2, \ldots, b_K, b_{K+1}, b_{K+2}, \ldots, b_\ell$ is an optimal solution

Why? $end(a_1) \leq end(b_1)$

so, a_1 doesn't intersect with b_2

Claim: $a_1, a_2, \ldots, b_K, b_{K+1}, b_{K+2}, \ldots, b_\ell$ is an optimal solution

- b_2 does not intersect with a_1 so, greedy alg could have chosen it

Instead, it chose a_2 so $end(a_2) \leq end(b_2)$
- intervals are disjoint

⋮

Claim $a_1, a_2 \ldots a_K b_{K+1}, \ldots, b_\ell$ is an optimal solution

Claim: $K = \ell$ otherwise greedy alg would have continued to choose more intervals

Scheduling to minimize lateness

Task	time req'd	deadline
CS341	4 hrs	in 9 hrs
Math	2 h	6 hrs
Phil	3 h	14
CS350	10 h	25

Can you do everything by its deadline?

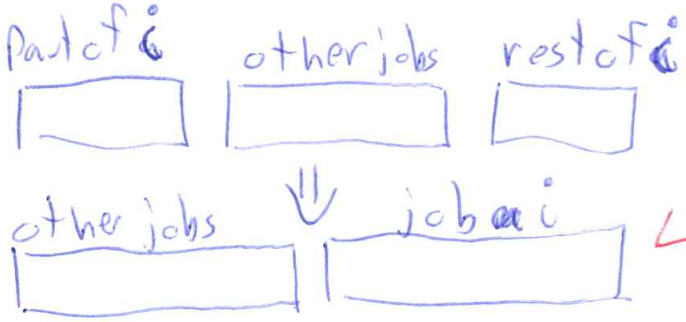
- ignore sleep!

Why is the optimization problem more general?

- a schedule completes all jobs on time

iff its maximum lateness is 0

Observation 1: Once you start a job, you might as well finish it



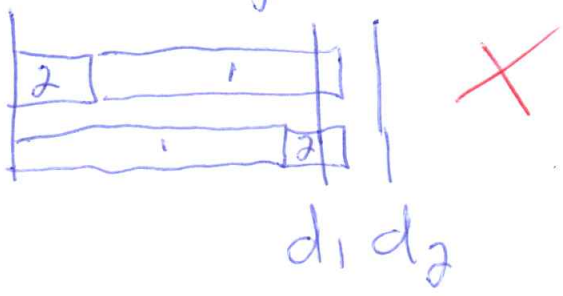
This is at least as good
- other jobs complete earlier
- job i completes same time

=> each job should be done contiguously.

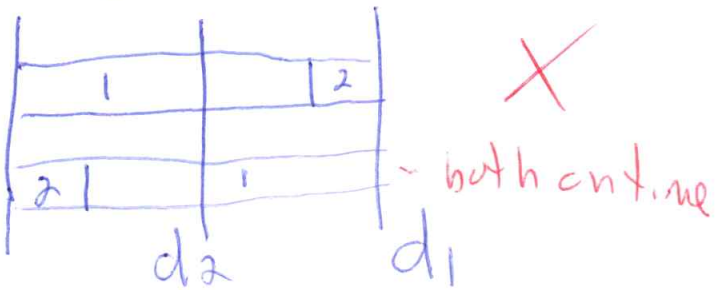
Observation 2: There is never any value taking a break

Greedy approaches:

- do short jobs first



- do jobs with less slack first: slack = $d_i - t_i$



do jobs in order of deadline:

- order jobs (relabel) so $d_1 \leq d_2 \leq \dots \leq d_n$
- seems to work on previous examples

Greedy alg: order job by deadline

- will show minimizes lateness

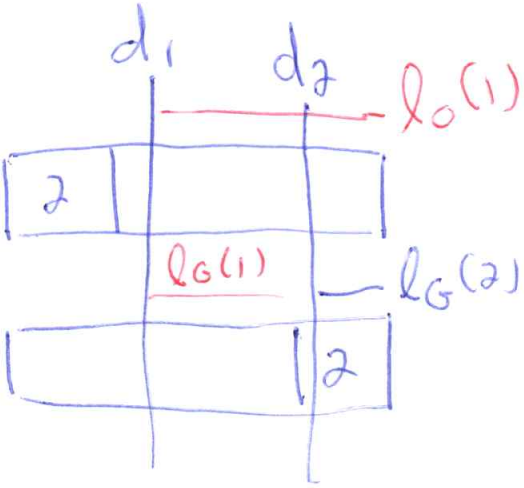
Proof Advice

- Don't be general at first! Try special cases!

Ex: $n=2$, $d_1 < d_2$

Wrong/Other solution O:

Greedy Sol G:



O has job 2 before 1

G has job 1 before 2

$\ell_O(1)$ = lateness of job 1 in O, ...

ℓ_G - maximum lateness of greedy $= \max\{\ell_G(1), \ell_G(2)\}$

ℓ_O other $= \max\{\ell_O(1), \ell_O(2)\}$

see next

$\ell_G(1) \leq \ell_O(1)$ because we moved 1 earlier

$\ell_G(2) \leq \ell_O(1)$ because $d_1 \leq d_2$

$\therefore \ell_G \leq \ell_O(1) \leq \ell_O$

Generalize this?

- This idea allows us to swap a pair of consecutive jobs if their dead lines are out of order, making the solution better or at least no worse

"Exchange" Proof

Thm The greedy alg gives an optimal solution - one that minimizes the maximum lateness

Proof: an "exchange proof" that converts any solution to the greedy one without increasing the maximum lateness.

Let 1..n be ordering of jobs by greedy alg.

- $d_1 \leq d_2 \leq \ldots \leq d_n$

Consider an optimal job ordering.

If it matches greedy => done.

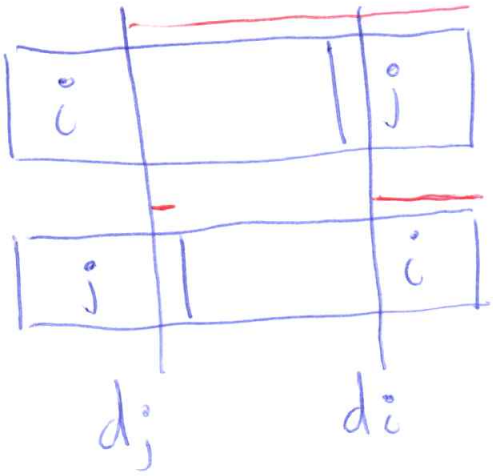
Otherwise, there must be 2 jobs that are consecutive in this ordering but in wrong order for greedy: i, j with $d_j \leq d_i$

Claim: Swapping i and j gives a new optimal ordering.

- new order has fewer inversions
  - => repeated swaps will eventually produce the greedy ordering
  - => new order is no worse, so greedy is also optimal.

Proof of claim:

Consider swapping jobs i and j



- swapping - both still finish at same time (as a pair)

$\ell_N(j) \leq \ell_O(j)$ we do j first now so it ends no later

$\ell_N(i) \leq \ell_O(j)$ because $d_j \leq d_i$

Note: all other jobs have same lateness

Thus $\ell_N \leq \ell_O$ but ℓ_O was minimum.

So $\ell_N = \ell_O$.

We can keep swapping until we obtain greedy solution and ℓ is unchanged.

Fractional Knapsack

- it makes sense to order items by value per weight

Greedy: choose item with largest $\frac{v_i}{w_i}$
choose as much of this item as possible

Correct?

Claim: The greedy alg gives the optimal sol for F.K.

Proof: greedy soln: $x_1 \ldots x_{k-1}\ x_k \ldots x_e \ldots x_n$
optimal soln: $y_1 \ldots y_{k-1}\ y_k \ldots y_e \ldots y_n$

Suppose y is an optimal soln that matches x on maximum # of indices, say M.

Assume, to arrive at a contradiction that greedy is not optimal: $M < n$

Show $\exists$ optimal soln that matches x on at least $M+1$ indices.

Let k be the first index where $x_k \neq y_k$

Then $x_k > y_k$ since greedy maximizes x_k

Since $\sum y = \sum x = W$, there is a later index $\ell > k$

s.t. $y_\ell > x_\ell$

- greedy took more of item k than opt
- opt took more of item ℓ than greedy

Exchange weight Δ of item ℓ for equal weight of item k in optimal soln.

$y'_k \leftarrow y_k + \Delta$

$y'_\ell \leftarrow y_\ell - \Delta$

choose $\Delta \leftarrow \min \begin{cases} y_\ell - x_\ell \\ x_k - y_k \end{cases}$

Then $x_k = y'_k$ or $x_\ell = y'_\ell$ and $\Delta > 0$

Change in value: $\Delta\left(\frac{v_k}{w_k}\right) - \Delta\left(\frac{v_\ell}{w_\ell}\right) = \Delta\left(\frac{v_k}{w_k} - \frac{v_\ell}{w_\ell}\right) \geq 0$

since $\frac{v_k}{w_k} \geq \frac{v_\ell}{w_\ell}$ (how we sorted)

y was optimal so, this can't be better.

New opt soln matches on 1 (at least) more index k or ℓ

Stable Marriage - Gale-Shapley Alg

rephrased: match employers with jobs to co-op students

- Once employed a student only changes jobs to one they prefer more.

- An employer can lose a student $O(n)$ times.

  Can they then make $O(n^2)$ offers?

  No, only try each student once.
  - start with highest preference
  - never offer to same student twice

Terminates:

Each employer offers to each student at most once.

$\Rightarrow$ at most $O(n^2)$ iterations

Correctness:

① we get n pairs. no employer can be rejected by all n students
∴ if one is unemployed, they accept

- once employed, a student stays employed.
- n employers, n students

② Stable Matching? Consider pair (e, s)
- if employer e prefers a student ss over s, they already offered to ss but were rejected.
- rejected because ss matched with a higher preference than e.

=> stable matching

Data Structure

matrices? Need to store index where each employer is at.

priority queues

Aside: lookup Gale & Shapley on wiki

=> Doctoral Advisor: Albert Tucker

=> John Nash (Beautiful Mind) 1998

Jack Edmonds coined term "greedy"? Is this true?