2 October 2019

<u>So far</u>: Dynamic Programming
Divide & Conquer
Randomized
Sorting

this week: Greedy Algorithms

<u>Making Change</u>: US = 1¢, 5¢, 10¢, 25¢

$$83¢ = \underbrace{25 + 25 + 25}_{75¢} + 5¢ + 1¢ + 1¢ + 1¢$$

Assuming, 1¢ exists, ~~what~~ does "this" always work? Not optimally.

- eg: what if no nickles?

  $30¢ = 25 + 1 + 1 + 1 + 1 + 1$ } 6 coins

  but $10 + 10 + 10$ } 3 coins

- UK currency: 1, 2¢, 5¢, 10¢, 20¢, 25¢

  try to make 40¢

Greedy Make Change

→ example of an optimization problem, that is, a problem that has many solutions, but you wish to min/max some function defined over those sol'ns

- many sol'ns: different ways to make change
- opt: min number of coins used
- note: might not be a unique solution.

Algorithm: assume $1¢ \in d$

```
GMC (val, d=[d_1, ..., d_k])
    sort d from large to small.
    for i = 1...k
    |  * add as many d_i as possible to S
    endfor
    return set S of coins we "collected"
```

Q = post condition

$= \sum_{c \in S} value(c) = val$

and is done in the least # of coins.

What is the loop invariant here?

L_i = what is true at * in the i^{th} loop.

S_i = sum of values in S after i^{th} ~~ith~~ iteration
(w/ $S_0 = 0$)

$L_i = \{ S_i \geq S_{i-1}$ and $S_i \leq val$ and no more $d_1 \ldots d_i$ can be added w/out going over val
and S is a subset of an optimal solution $\}$

Loop Inv:	Init
	Maint.
	End

Termination : $\neg G \wedge L \Rightarrow Q$

$\neg G \wedge L \Rightarrow i > k$ = the # of denominations
and $L_i \Rightarrow S$ is a subset of the optimal sol'n.

$\Rightarrow$ 1¢ has already been considered

$\Rightarrow$ Sum of values in S = val

$\Rightarrow S$ is optimal. ∎

$val = 83¢$, $d = [25, 10, 5, 1]$

What if I add however many I want w/out going over val?

$\underline{\underline{i=1}}$

$S = \{25, 25\}$

L_i: ✓ $\Sigma \emptyset \leq 50¢$

✓ $50¢ \leq 83¢$

✓ S is a subset of an optimal sol'n

X no more d_i can be added.

$\underline{\underline{i=2}}$ $S = \{25, 25, 10, 10, 10\}$

no longer the subset of an optimal solution!

All ways to make 83¢

$$\left.\begin{array}{l} 3 \cdot 25 + 1 \cdot 5 + 3 \cdot 1 \\ 83 \cdot 1 \\ 10 \cdot 5 + 3 \cdot 10 + 3 \cdot 1 \end{array}\right\} \text{enumerating} = \text{listing all ways}$$

on proving optimality for greedy:

→ think "stays ahead"

i.o.w., given <u>any</u> other sol'n, the one I am building (in this loop) is better in some way.

---

scheduling:

Given tasks $T = \{[b_i, f_i)\}_{i=1}^{n}$

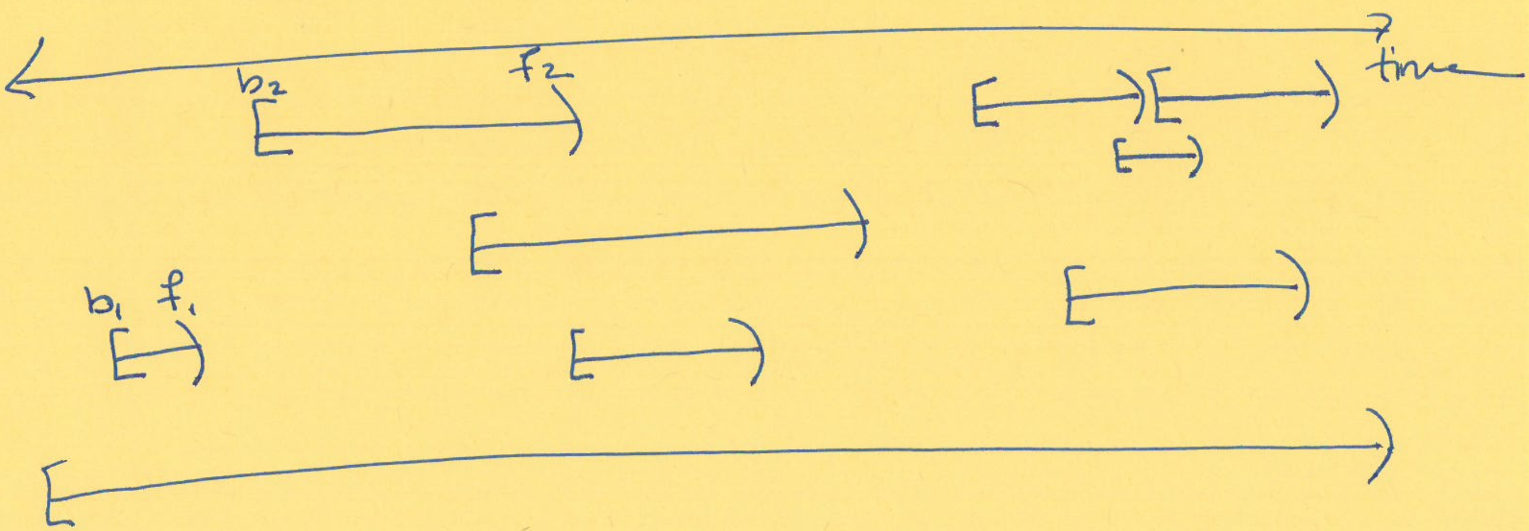
beginning time — finishing time

Want: $S \subseteq T$ s.t. $\forall [b_i, f_i) \neq [b_j, f_j)$,

$f_i \le b_j$ or $f_j \le b_i$

(i.e., the intervals are disjoint)

such that $|S|$ is maximized.



greedy-1: pick the one that ends first
greedy-2: pick the one that starts last

for greedy-1, how does this "stay ahead?"

- S = set of all optimal solutions
- G = our greedy solution.
- claim: the ith interval in G ends befor or at the same time as any solution in S.