Greedy Algorithms

"Greed is good. Greed is right. Greed works." — Oliver Stone "Wall Street" 1987

Principle:

Greedy principle: A global optimal is reached by doing local greedy choices

Optimal substructure (like dynamic programming)

Notes: won't talk about theory of Matroids

- They usually provide good approximate solutions to NP-complete problems
- Good for Min/Max problems

Activity Selection Problem

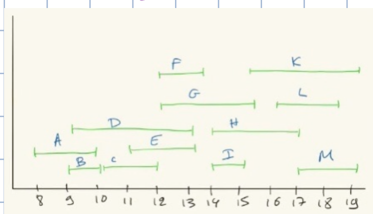
There's a set of (possibly) overlapping class but only one classroom.

How to maximize the # of classes to fit in the classroom?

Algorithm:

1. Sort by finish time
2. Schedule with earliest finish time possible → being greedy

Time: $O(n\log n)$



1.	2.
B	B
A	~~A~~
C	C
E	~~E~~
D	~~D~~
F	F
I	I
G	~~G~~
H	~~H~~
L	L
M	~~M~~
k	~~K~~

⇒ max of 5 classes: B C F I L

but can also do: A C F I L

⇒ solution may not be unique

Proof of correctness: Why greedy G is optimal?

ATAC that G is not optimal but some other solution O is.

$g_1\ g_2\ g_3 \cdots g_m$

$\cancel{O_1}\ \cancel{O_2}\ \cancel{O_3} \cdots \cancel{O_m} \ldots O_n$

$g_1\ g_2\ g_3 \quad g_m$

Can I replace O_1 with g_1? Yes, b/c greedy selected the class g_1 $f(g_1) \leq f(O_1)$ by definition

Does those classes exist? No, as greedy would've selected them

Above, we managed to "transform" the optimal to the greedy without sacrificing optimality. Hence greedy is optimal.

The knapsack problem

A thief enters a store that has n items. Item i values V_i and weights w_i. Theif can carry W weight only.

1. Fractional knapsack (greedy)

   can take a potion of an item (rice, beans etc.)

2. 0-1 knapsack (dynamic)

   can only take or leave a whole item

1. sort items per $\frac{V_i}{w_i}$ and will keep on taking in descending order maximizing the value for the w he can carry

   Time: $O(n\log n)$

2. Sort in any order to consider

$c[i,w]$ = the max value taking from item f...i with "leftover" weight W

$= \begin{cases} 0 & \text{if } w = 0 \end{cases}$

$$\begin{cases} c[i-1,w] & \text{if } w_i > w \\ \max\{c[i-1,w],\ c[i-1,w-w_i]\}+v_i & \text{if } w_i \leq w \end{cases}$$

Time: $O(nW)$