Greedy Algorithms

- It is an algorithm design technique

- A Greedy algorithm always makes the Choice that looks best at the moment.
  i.e., it makes a locally optimal Choice in the hope that this choice will lead to a globally optimal solution.

- Generally greedy algorithm are simple, however Proving that greedy algorithm Produces an optimal solution to a Problem is challenging sometimes.

Fractional Knapsack

It is Similar to 0-1 knapsack, however here we are allowed Put any fraction of an item into the knapsack.

Formally,

Given n items with weights $w_1, \ldots w_n$ and value $v_1, \ldots v_n$, Knapsack of capacity W.

Select fractions $p_1, p_2, \ldots p_n$ to maximize $p_1v_1 + \cdots + p_nv_n$

such that $p_1w_1 + p_2w_2 + \cdots + p_nw_n \leq W$,

where $0 \leq p_i \leq 1$.

Example: $W = 10$

$w_1 = 4, \quad w_2 = 5, \quad w_3 = 7$

$v_1 = 2, \quad v_2 = 3, \quad v_3 = 4$

Optimal Solution: item 2 $+ \frac{5}{7}$ item 3

$$\text{Value} = 3 + \frac{5}{7} \times 4 = 5 + \frac{6}{7}$$

$$\text{Weight} = 5 + \frac{5}{7} \times 7 = 10 \leq W$$

If it were 0-1 Knapsack:

Optimal solution: item 1 + item 2

Value: $2 + 3 = 5$

Weight: $4 + 5 = 9 \leq W$.

Greedy Algorithm

Consider the items in non-increasing value-per-weight ratio. Add items to Knapsack one at a time, in this order, until we reach an item whose addition would cause knapsack's capacity W to be exceeded. Add the largest fraction of that item that fits into the knapsack and stop.

Above Example: $W = 10$

$w_1 = 4, \quad w_2 = 5, \quad w_3 = 7$

$v_1 = 2, \quad v_2 = 3, \quad v_3 = 4$

value per weight $\frac{2}{4} = 0.5 \qquad \frac{3}{5} = 0.6 \qquad \frac{4}{7} = 0.57$

$$\frac{3}{5} \geq \frac{4}{7} \geq \frac{2}{4}$$

Opt Soln: Pick item 2 + largest possible fraction of item 3

Running time:

$O(n \log n)$ time to sort the items by the ratio in non-increasing order.

$O(n)$ time to traverse and pick from the list of items until knapsack is full.

Proof of Correctness :

Proof by Contradiction

Suppose there is an instance of fractional knapsack such that the solution of the algorithm (ALG) is not optimal.

Let OPT denote the optimal solution.

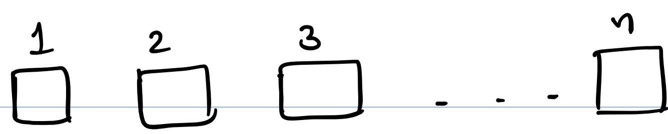
WLOG we assume no items have same $\frac{v_i}{w_i}$.

and items are sorted in decreasing order of v_i/w_i

Let $ALG = \{p_1, p_2, \dots p_n\}$ be the fraction of items picked by Algorithm

$OPT = \{q_1, q_2, \dots q_n\}$ optimal soln.

By assumption $\sum_{i=1}^{n} p_i v_i < \sum_{i=1}^{n} q_i v_i$.



let i be the first index at $P_i \neq q_i$.

By our greedy algorithm $P_i > q_i$

[because our algo takes maximum possible fraction of an item]

By the optimality, there must exist an item $j > i$

such that $P_j < q_j$

Define new soln $q' = \{q_1', q_2', \ldots q_n'\}$

where $q_k' = q_k$ for all $k \neq i, j$

q' will take a little more of item i and a little less of item j compared to OPT.

$q_i' = q_i + \varepsilon_i$, ε_i is the fraction of item i

$q_j' = q_j - \varepsilon_j$ ε_j is the fraction of item j

where $\varepsilon_i = \varepsilon_j$

Total weight doesn't change.

Total value $\sum_{i=1}^{n} q_i' v_i = \sum_{i=1}^{n} q_i v_i + \text{Value}(\epsilon_i) - \text{Value}(\epsilon_j)$

$$> \sum_{i=1}^{n} q_i v_i$$

ie., q' is a valid & better solution than OPT, which is a contradiction.

Hence, the solution from our algorithm is optimal.

Q Can we solve the fractional knapsack in $O(n)$ time? HINT: MEDIAN

Activity Selection Problem [CLRS 16.1]

Given a Set S of n activities

s_i = Start time of activity a_i

f_i = finish time of activity a_i

Activity a_i takes Place during the interval $[s_i, f_i)$

Two activities a_i and a_j are Compatible if the intervals $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Find max-size Subset A of Compatible activities.

Example:

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

$\{a_1, a_4, a_8, a_{11}\}$ and $\{a_2, a_4, a_9, a_{11}\}$

are largest subset of mutually compatible activities.

For the rest of the discussion

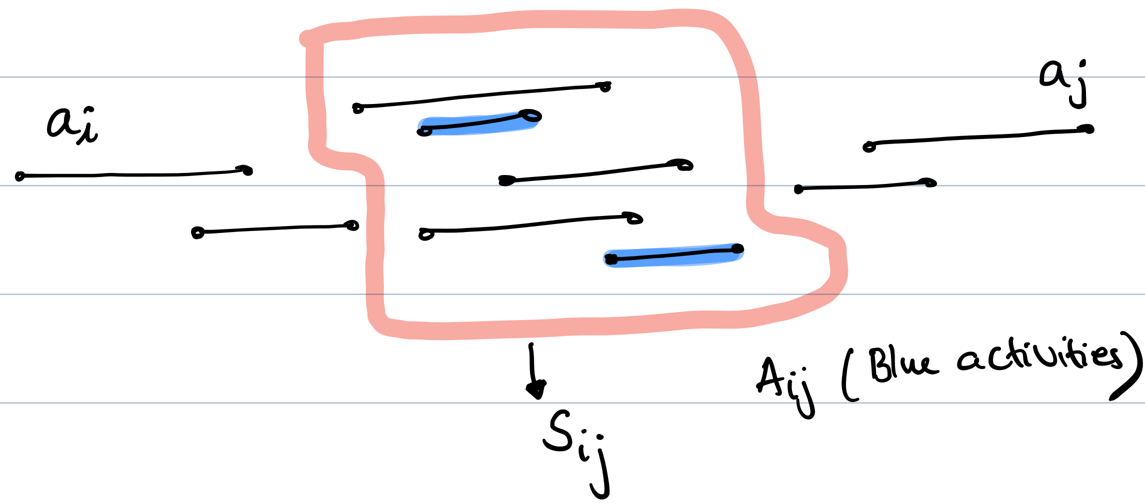
We assume that $f_1 \le f_2 \le \dots f_n$.

Possible ways

- Brute Force
- Dynamic Programming
- Greedy algorithm

Outline of DP - Algorithm

let S_{ij} be the set of activities that starts after activity a_i finishes and finish before the activity a_j starts



let A_{ij} be a maximum set of mutually compatible activities in S_{ij}

Optimal Substructure

Suppose $a_k \in A_{ij}$

Then we get two subproblems.

Finding mutually compatible activities in S_{ik} and

" " " " " S_{kj}

let $A_{ik} = A_{ij} \cap S_{ik}$

$A_{kj} = A_{ij} \cap S_{kj}$

$\therefore \quad A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$

$|A_{ij}| = |A_{ik}| + |A_{kj}| + 1$

let $c[i,j]$ denote the size of the optimal solution for the set S_{ij}, then we get

$$c[i,j] = \begin{cases} 0 & \text{if } S_{ij} = \phi \\ \max\limits_{a_k \in S_{ij}} \{c[i,k] + c[k,j] + 1\} & \text{if } S_{ij} \neq \phi \end{cases}$$

From here onwards the procedure is similar to the other DP problems we have seen so far.

Greedy Algorithms

Warmup

Possible Greedy Choices

1. Select the activity which takes least time

2. Select the activity which starts first

3. Select the activity which has minimal overlaps with other activities

Greedy Choice

Select the activity with earliest finish time.

Proof of Correctness :

Theorem 16.1
Consider any nonempty subproblem S_k, and let a_m be an activity in S_k with the earliest finish time. Then a_m is included in some maximum-size subset of mutually compatible activities of S_k.

Proof:
$$S_k = \{a_i \in S : s_i \geq f_k\}$$

let A_k be a maximum-size subset of mutually compatible activities in S_k.

if $a_m \in A_k$ then we are done.

if $a_m \notin A_k$

then let a_j be the activity in A_k with the earliest finish time.

let $A'_k = A_k - \{a_j\} \cup \{a_m\}$

ⓐ $|A'_k| = |A_k|$

ⓑ The activities in A'_k are disjoint, because the activities in A_k are disjoint, a_j is the first activity in A_k to finish and $f_m \leq f_j$.

$\therefore$ A'_k is a maximum-size subset of mutually compatible activities of S_k containing a_m.

Algorithm

Input activities are ordered by increasing finish time.

```
GREEDY-ACTIVITY-SELECTOR(s, f)
n = s.length
A = {a_1}
k = 1
for m = 2 to n
    if s[m] ≥ f[k]
        A = A ∪ {a_m}
        k = m
return A
```

Running time: $\Theta(n)$. (By assuming that the activities were already sorted by their finish times)

Exercise ①

Suppose we select the last activity to start that is compatible with all previously selected activities then show that it yields an optimal solution.

Exercise ②:

In the activity selection Problem, each activity a_i has in addition to a Start and finish time, a value v_i.

The Objective is to maximize the total value of the activities scheduled.

i.e., choose a set A of compatible activities such that $\sum_{a_k \in A} v_k$ is maximized.

Give a Polynomial-time algorithm for this Problem.