Notes, 6(a)

ECE 606

Algorithm Design Strategy III: Greedy

Some optimization problems possess a *greedy choice property*: there exists a locally-optimal, i.e., "greedy," choice, a sequence of which leads to a globally-optimal solution.

Example: change-making with coin-values 1, 5, 10 and 25.

Suppose we have a positive integer amount $a \in \mathbb{Z}^+$. And we have an unlimited number of coins of each value. And we want to minimize the number of coins that make up the amount a. That is:

- Input: $a \in \mathbb{Z}^+$.
- Output: a vector $\langle a_1, a_5, a_{10}, a_{25} \rangle$ such that:
  (1) $a = 1 \times a_1 + 5 \times a_5 + 10 \times a_{10} + 25 \times a_{25}$, and,
  (2) $a_1 + a_5 + a_{10} + a_{25}$ is the minimum across all possibilities that satisfy property (1).

Example: two possibilities for input $a = 63$ are: (i) $\langle 3, 0, 1, 2 \rangle$ and (ii) $\langle 3, 0, 6, 0 \rangle$. Option (i) is better, and indeed, the optimal — we claim that we cannot do with fewer than 6 coins.

The above problem possesses a greedy choice property: first hand out as many coins of value 25, then as many of value 10, and so on.

However, if we change the values of the coins, then the problem does not necessarily possess *that particular* greedy choice. So, suppose we change the problem to:

- Inputs: (i) $a \in \mathbb{Z}^+$, and, (ii) $C = \langle c_0, c_1, \ldots, c_{k-1} \rangle$ where each $c_i \in \mathbb{Z}^+$, $c_0 = 1$ and every $c_i < c_{i+1}$.
- Output: $A = \langle a_0, a_1, \ldots, a_{k-1} \rangle$ coin-values of each type that satisfy properties (1) and (2) from the previous problem definition.

Example: $a = 6$, $C = \langle 1, 3, 4 \rangle$. Then, the above greedy-choice would suggest $A = \langle 2, 0, 1 \rangle$. But a better solution is $A' = \langle 0, 2, 0 \rangle$.

Example: fractional knapsack.

We have n piles of gold dust, each a pair $\langle w_i, v_i \rangle$, where w_i is the weight of the pile and v_i is the total value of the pile. We also have a knapsack which is a positive integer W, which is the weight-capacity of the knapsack.

Which partial/full piles of gold dust should we take to maximize the value in the knapsack while not exceeding its weight capacity, W?

Solution: the above problem possesses a greedy choice. And that is, greedily pick as much of the pile which maximizes v_i/w_i as possible. E.g., if we have the piles $p_1 = \langle 4, 8 \rangle$, $p_2 = \langle 3, 5 \rangle$, $p_3 = \langle 3, 4 \rangle$. Then p_1's fractional value is $8/4 = 2$, p_2's is $5/3 = 1.67$ and p_3's is $4/3 = 1.33$.

So given a knapsack of capacity $W = 6$, we take all of p_1, and 2 units of weight of p_2 to yield a total value of $\frac{8}{4} \times 4 + \frac{5}{3} \times 2 = 8 + 3.33 = 11.33$.

The above problem is called fractional knapsack because we are allowed to take fractions of items, i.e., of the piles of gold.

A discretized version of the problem: 0-1 knapsack.

We have bars of gold which cannot be sub-divided, and not piles of gold. We must either take an entire bar, or none of it.

0-1 knapsack does not possess the same greedy choice property. E.g., if we are given the three gold bars $b_1 = \langle 4, 8 \rangle$, $b_2 = \langle 3, 5 \rangle$, $b_3 = \langle 3, 4 \rangle$. The knapsack has capacity $W = 6$. Now, if we greedily pick b_1 which has highest fractional value, we cannot take either of the other bars, and we end up with a value of 8 in the knapsack.

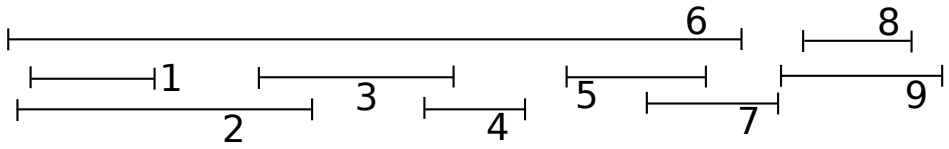
A better option would have been to take both of b_2, b_3 thereby yielding a total value of 9.

The contrast between fractional and 0-1 knapsack is a good example of when discrete versions of problems can be much harder than continuous versions. Another example of this: linear programming.

Interval scheduling

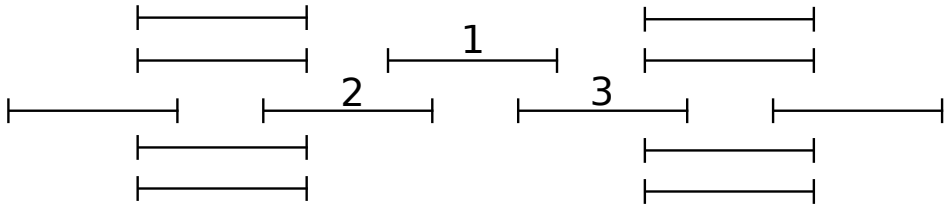
We have one room in which we can schedule meetings. We have n meeting requests, each a pair, $\langle s(i), f(i)\rangle$, where $s(i)$ is the start-time of the ith meeting and $f(i)$ its finish time, and $s(i), f(i) \in \mathbb{Z}^+$ with $s(i) < f(i)$ for every i. We seek to schedule the maximum number of meetings none of which is *in conflict* with another.

Two meeting requests, i and j are in conflict if one of them starts before the other, and ends only after the other starts. That is, $s(i) \leq s(j) \leq f(i)$, or $s(j) \leq s(i) \leq f(j)$.



In the above example of an input, requests (5) and (7) are in conflict, as are requests (1) and (6). However, requests (6) and (9) are not in conflict.

It turns out that the problem possesses greedy choice, but not necessarily an obvious one. E.g., a request of shortest duration does not always work, nor does a request with fewest conflicts nor a request with earlier start-time.



In the above example, if we greedily schedule request (1), we can schedule neither request (2) nor (3). Thus, we end up scheduling 3 meetings only; 4 is the optimum.

It turns out that a valid greedy choice is the ~~task~~ request with earliest finish time. That is, our algorithm is:

while *more requests remain* **do**
    Pick a request r with earliest finish time
    Remove all requests that are in conflict with r

The above algorithm results in a set of non-conflicting requests. To prove that it yields an optimal set, we do the following. Let $T = \{t_1, t_2, \ldots, t_m\}$ be an optimal set of requests. Suppose our greedy choice outputs $G = \{g_1, g_2, \ldots, g_k\}$. We seek to prove that $k = m$. We progressively "cut" or "remove" an item from T, and "paste" an item from G, till we end up exactly with G as an optimal set. This is called a "cut and paste" proof strategy.

Assume: $s(t_1) \leq s(t_2) \leq \ldots \leq s(t_m)$ and $s(g_1) \leq \ldots \leq s(g_k)$. Because the requests in each of T and G are non-conflicting, this implies that $f(t_1) \leq \ldots \leq f(t_m)$ and $f(g_1) \leq \ldots \leq f(g_k)$.

Claim 1. *For every* $i = 1, \ldots, k$, $f(g_i) \leq f(t_i)$.

Proof. By induction on i. Base case: $i = 1$, and the proof is by our greedy choice.

For the step: assume true for all $i = 1, \ldots, p-1$. Then:

- $f(g_{p-1}) \leq f(t_{p-1}) \leq s(t_p)$, and therefore,
- t_p non-conflicting with $g_{p-1} \implies f(g_p) \leq f(t_p)$.

□

Claim 2. $k = m$

Proof. By contradiction, and then same proof strategy as above claim. □