Greedy Algorithms

We looked at divide and conquer where we solved an identical problem on a smaller subset of input.

We will now look at greedy algorithm solutions.

Greedy algorithms – Overview

- Algorithms for solving (optimization) problems typically go through a sequence of steps, with a set of choices at each step.

- A greedy algorithm always makes the choice that *looks best at the moment*, without regard for future consequence,
  i.e., *"take what you can get now"* strategy

- Greedy algorithms do not always yield optimal solutions,

$$\text{Local optimum} \quad \overset{?}{\not\Rightarrow} \quad \text{Global optimum}$$

- but for many problems they do.

Activity-selection problem

Problem statement:

Input: *Set* $S = \{1, 2, \ldots, n\}$ *of* n *activities*
s_i = *start time of activity* i
f_i = *finish time of activity* i

Output: *Maximum-size subset* $A \subseteq S$ *of compatible activities*

Notes:

- Activities i and j are compatible if the intervals $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.
- Without loss of generality, assume

$$f_1 \leq f_2 \leq \cdots \leq f_n$$

Activity-selection problem

Greedy algorithm:

- *pick the compatible activity with the earliest finish time.*

Why?

- Intuitively, this choice leaves as much opportunity as possible for the remaining activities to be scheduled
- That is, the greedy choice is the one that maximizes the amount of unscheduled time remaining.

Activity-selection problem

```
Greedy_Activity_Selector(s,f)
n = length(s)
A = {1}
j = 1
for i = 2 to n
    if s[i] >= f[j]
       A = A U {i}
       j = i
    end if
end for
return A
```

Remarks

- Assume the array `f` already sorted
- Complexity: $T(n) = O(n)$

Activity-selection problem

Question: Does `Greedy_Activity_Selector` work?
Answer: Yes!

Theorem. Algorithm `Greedy_Activity_Selector` produces a solution of the activity-selection problem.

Activity-selection problem

The proof of **Theorem** is based on the following two properties:

Property 1.

There exists an optimal solution A such that the greedy choice "1" in A.

Proof:

- *let's order the activities in A by finish time such that the first activity in A is "k_1".*
- *If $k_1 = 1$, then A begins with a greedy choice*
- *If $k_1 \neq 1$, then let $A' = (A - \{k_1\}) \cup \{1\}$.*
  Then
  1. *the sets $A - \{k_1\}$ and $\{1\}$ are disjoint*
  2. *the activities in A' are compatible*
  3. *A' is also optimal, since $|A'| = |A|$*
- *Therefore, we conclude that* there always exists an optimal solution that begins with a greedy choice.

Activity-selection problem

Property 2.

If A is an optimal solution, then $A' = A - \{1\}$ is an optimal solution to $S' = \{i \in S, s[i] \geq f[1]\}$.

Proof: By contradiction. If there exists B' to S' such that $|B'| > |A'|$, then let

$$B = B' \cup \{1\},$$

we have

$$|B| > |A|,$$

which is contradicting to the optimality of A.

Activity-selection problem

Proof of **Theorem**: By Properties 1 and 2, we know that

- After each greedy choice is made, we are left with an optimization problem of the same form as the original.
- *By induction* on the number of choices made, making the greedy choice at every step procedures an optimal solution.

Therefore, the `Greedy_Activity_Selector` produces a solution of the activity-selection problem.

Activity-selection problem

- Property 1 is called **the greedy-choice property**, generally casted as

  a globally optimal solution can be arrived at by making a locally optimal (greedy) choice.

- Property 2 is called **the optimal substructure property**, generally casted as

  an optimal solution to the problem contains within it optimal solution to subprograms.

These are two key properties for the success of greedy algorithms.