

CH-231-A

Algorithms and Data Structures

ADS

Lecture 29

Dr. Kinga Lipskoch

Spring 2022

Design Concepts

- ▶ We have been looking into different algorithms and, in particular, emphasized one design concept, namely, the Divide & Conquer strategy, which was based on recursions and whose analysis was given by recurrences.
- ▶ Now, we are going to look into further design concepts.

Activity-Selection Problem (1)

- ▶ Suppose we have a set $S = \{a_1, a_2, \dots, a_n\}$ of n activities.
- ▶ The activities wish to use a resource, which can only be used by one activity at a time.
- ▶ Each activity a_i has a start time s_i and a finish time f_i , where $0 \leq s_i < f_i < \infty$.
- ▶ Two activities a_i and a_j are compatible, if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint.
- ▶ The activity-selection problem is to select a maximum-size subset of mutually compatible activities.

Activity-Selection Problem (2)

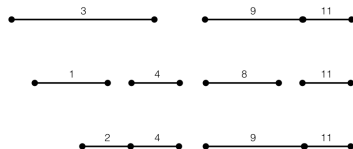
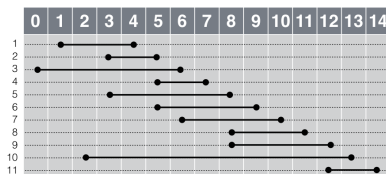
► Example:

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 8 9 10 11 12 13 14

- $\{a_3, a_9, a_{11}\}$ is a subset of mutually compatible activities.
- $\{a_1, a_4, a_8, a_{11}\}$ is a largest subset of mutually compatible activities.
- $\{a_2, a_4, a_9, a_{11}\}$ is another largest subset of mutually compatible activities.



Sorting

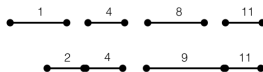
- ▶ We can apply a sorting algorithm to the finish times, which operates in $O(n \lg n)$ time.
- ▶ Then, we can assume that the activities are sorted, i.e.,
 $f_1 \leq f_2 \leq \dots \leq f_n$.

Greedy Algorithm

- ▶ 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 it will lead to a globally optimal solution.

Greedy Approach (1)

- ▶ After sorting, a_1 has the earliest finish time f_1 .
- ▶ A greedy approach starts with taking a_1 as a locally optimal choice.
- ▶ **Lemma:**
The greedy choice of picking a_1 as first choice is optimal.
- ▶ **Proof:**
 - ▶ Suppose A is a globally optimal solution for set S .
 - ▶ Let $a_k \in A$ be the activity with earliest finish time f_k in A .
 - ▶ If $k = 1$, then $a_1 \in A$ and we are done.
 - ▶ If $k > 1$, then we can replace A by $(A \setminus \{a_k\}) \cup \{a_1\}$.
 - ▶ Since $f_1 \leq f_k$, this is still an optimal solution.
 - ▶ Hence, we can always start with a_1 .



Greedy Approach (2)

- ▶ After the first step, we consider the subproblem $S' = \{a_i \in S : s_i \geq f_1\}$.
- ▶ We apply the same greedy strategy.
- ▶ **Lemma:**
 $A \setminus \{a_1\}$ is the optimal solution for S' .
- ▶ **Proof:**
 - ▶ Let B be a solution for S' that is larger than $A \setminus \{a_1\}$.
 - ▶ Then, $B \cup \{a_1\}$ would be solution for S that is larger than A .
 - ▶ Contradiction.

Greedy Approach (3)

Using the two lemmata we can prove by induction that the greedy approach delivers the globally optimal solution.

Greedy Algorithm

```
1 Greedy-Selector(S)
2   // Assume S = {a[1], ..., a[n]}
3   // with activities sorted by f[i].
4   A := {a[1]}
5   j := 1
6   for i := 2 to n do
7     if s[i] >= f[j]
8       then A := A union {a[i]}
9         j := i
10  return A
```

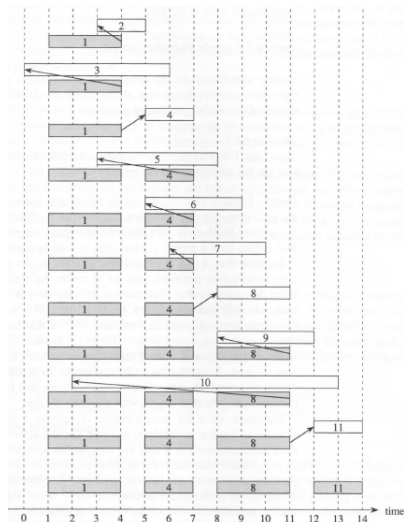
Example

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 8 9 10 11 12 13 14

Alternative:
Recursive function



Greedy Algorithm (Recursive)

RECURSIVE-ACTIVITY-SELECTOR(s, f, k, n)

```
1   $m = k + 1$ 
2  while  $m \leq n$  and  $s[m] < f[k]$       // find the first activity in  $S_k$  to finish
3       $m = m + 1$ 
4  if  $m \leq n$ 
5      return  $\{a_m\} \cup \text{RECURSIVE-ACTIVITY-SELECTOR}(s, f, m, n)$ 
6  else return  $\emptyset$ 
```

RECURSIVE-ACTIVITY-SELECTOR($s, f, 0, n$).

Time Complexity of Greedy Approach

- ▶ $O(n)$ – if already sorted
- ▶ $O(n \lg n)$ – if not sorted
- ▶ Comparison to **brute-force** approach:
 - ▶ Brute-force approach would try all combinations, reject all the combinations that have incompatibilities, and pick among the remaining ones one with maximum number of activities.
 - ▶ Time complexity: $O(2^n)$

Greedy Algorithm in General

- ▶ Greedy algorithms build upon solving subproblems.
- ▶ Greedy approaches make a locally optimal choice.
- ▶ There is no guarantee that this will lead to a globally optimal solution.
- ▶ Having found a global optimum requires a proof.