Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Techniques for the design of Algorithms

The classical techniques are as follows:

1. Divide and Conquer
2. Dynamic Programming
3. Greedy Algorithms
4. Backtracking Algorithms
5. Branch and Bound Algorithms

An activity-selection problem

Definition

- Given a set $S = \{a_1, a_2, \cdots, a_n\}$ of n activities that wish to use a resource.
- Each activity a_i has a start time s_i and a finish time f_i, where $0 \le s_i < f_i < \infty$. If selected, activity a_i takes place during the time interval $[s_i, f_i)$.
- Activities a_i and a_j are compatible if the intervals $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap (i.e. $s_i \ge f_j$ or $s_j \ge f_i$).

The activity-selection problem is to select a maximum-size subset of compatible activities.

An activity-selection problem

Example

Consider the following activities:

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

Some of the compatible activities are as follows:

- $\{a_3, a_9, a_{11}\}$
- $\{a_1, a_4, a_8, a_{11}\}$
- $\{a_2, a_4, a_9, a_{11}\}$

An activity-selection problem: Dynamic Programming

- $S \longleftarrow S \cup \{a_0, a_{n+1}\}$, where $f_0 = 0$ and $s_{n+1} = \infty$.
- Sort finishing times so that $f_0 \leq f_1 \leq f_2 \leq \cdots \leq f_n \leq f_{n+1}$.
- Define $S_{ij} = \{a_k \in S \ : \ f_i \leq s_k < f_k \leq s_j\}$, where $0 \leq i,\ j \leq n+1$.
- Let $C[i,j]$ be the number of activities in a maximum-size subset of mutually compatible activities in S_{ij}.

An activity-selection problem: Dynamic Programming

- $S \longleftarrow S \cup \{a_0, a_{n+1}\}$, where $f_0 = 0$ and $s_{n+1} = \infty$.
- Sort finishing times so that $f_0 \leq f_1 \leq f_2 \leq \cdots \leq f_n \leq f_{n+1}$.
- Define $S_{ij} = \{a_k \in S \ : \ f_i \leq s_k < f_k \leq s_j\}$, where $0 \leq i, \ j \leq n+1$.
- Let $C[i,j]$ be the number of activities in a maximum-size subset of mutually compatible activities in S_{ij}.
- Compute $C[i,j]$ as follows:

$$C[i,j] = \begin{cases} 0 & \text{if } i \geq j, \\ \max\limits_{\substack{a_k \in S_{ij} \\ i \leq k \leq j}} \{C[i,k] + C[k,j] + 1\} & \text{if } i < j. \end{cases}$$

An activity-selection problem: Greedy Algorithm

Theorem

Consider any nonempty subproblem S_{ij}, and let a_m be the activity in S_{ij} with the earliest finish time, i.e. $f_m = min\{f_k \ : \ a_k \in S_{ij}\}$. Then

1. *Activity a_m is used in some maximum-size subset of compatible activities of S_{ij}.*
2. *The subproblem S_{im} is empty, so that choosing a_m leaves the subproblem S_{mj} as the only one that may be nonempty.*

An activity-selection problem: Greedy Algorithm

Theorem

Consider any nonempty subproblem S_{ij}, and let a_m be the activity in S_{ij} with the earliest finish time, i.e. $f_m = min\{f_k \ : \ a_k \in S_{ij}\}$. Then

1. *Activity a_m is used in some maximum-size subset of compatible activities of S_{ij}.*
2. *The subproblem S_{im} is empty, so that choosing a_m leaves the subproblem S_{mj} as the only one that may be nonempty.*

Proof.

2. If $S_{im} \neq \emptyset$ then there exists $a_k \in S_{im}$ such that $f_i \leq s_k < f_k \leq s_m < f_m$. So $a_k \in S_{ij}$ and $f_k < f_m$. $\boxtimes$

1. Suppose that A_{ij} is a maximum-size subset of compatible activities of S_{ij}. Let a_k be the first activity in A_{ij}.
   - If $a_k = a_m$, we are done.
   - If $a_k \neq a_m$, we construct the subset $A'_{ij} = A_{ij} - \{a_k\} \cup \{a_m\}$. Now, a_m is the first activity in A'_{ij} to finish, and $f_m \leq f_k$. Note that A'_{ij} has the same number of activities as A_{ij}, so A'_{ij} is a maximum-size subset of compatible activities of S_{ij} that includes a_m.

$\square$

An activity-selection problem: Recursive Algorithm

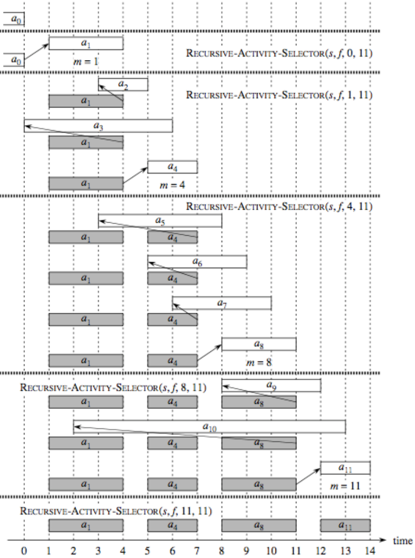
RECURSIVE-ACTIVITY-SELECTOR(s, f, i, n)

$m \leftarrow i + 1$
while $m \leq n$ and $s_m < f_i$ $\quad \triangleright$ Find the first activity in $S_{i,n+1}$.
    do $m \leftarrow m + 1$
if $m \leq n$
  then return $\{a_m\} \cup$ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
  else return $\emptyset$

Where the initial call is RECURSIVE-ACTIVITY-SELECTOR$(s, f, 0, n)$.

An activity-selection problem: Recursive Algorithm

k	s_k	f_k
0	–	0
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9
7	6	10
8	8	11
9	8	12
10	2	13
11	12	14
12	∞	–



An activity-selection problem: Greedy Algorithm

GREEDY-ACTIVITY-SELECTOR(s, f)

$n \leftarrow length[s]$

$A \leftarrow \{a_1\}$

$i \leftarrow 1$

for $m \leftarrow 2$ **to** n

do if $s_m \geq f_i$

then $A \leftarrow A \cup \{a_m\}$

$i \leftarrow m$

return A

Exercises

1. Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm and prove that it yields an optimal solution.
2. Suppose that we have a set of activities to schedule among a large number of lecture halls. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall.