طراحی و تحلیل الگوریتم ها

دکتر امیر لکی زاده

استادیار گروه مهندسی کامپیوتر دانشگاه قم

Greedy Algorithms

ویژگی اصلی:

در هر گام بهترین تصمیم را می گیرد و لذا لزوما بهترین جواب را به دست نمی دهد.

1. Activity-Selection Problem

Suppose we have a set $S = \{a_1, a_2, \ldots, a_n\}$ of n proposed ***activities*** that wish to use a resource, such as a lecture hall, which can be used by only one activity at a time.

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 half-open time interval $[s_i, f_i)$.

1. Activity-Selection Problem

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., a_i and a_j are compatible if $s_i \geq f_j$ or $s_j \geq f_i$).

The ***activity-selection problem*** is to select a maximum-size subset of mutually compatible activities. For example, consider the following set S of activities, which we have sorted in monotonically increasing order of finish time:

1. Activity-Selection Problem

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

1. Activity-Selection Problem

$$S_{ij} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\} ,$$

so that S_{ij} is the subset of activities in S that can start after activity a_i finishes and finish before activity a_j starts. In fact, S_{ij} consists of all activities that are compatible with a_i and a_j and are also compatible with all activities that finish no later than a_i finishes and all activities that start no earlier than a_j starts.

1. Activity-Selection Problem

we add fictitious activities a_0 and a_{n+1} and adopt the conventions that $f_0 = 0$ and $s_{n+1} = \infty$. Then $S = S_{0,n+1}$, and the ranges for i and j are given by $0 \leq i, j \leq n + 1$.

We can further restrict the ranges of i and j as follows. Let us assume that the activities are sorted in monotonically increasing order of finish time:

$$f_0 \leq f_1 \leq f_2 \leq \cdots \leq f_n < f_{n+1} . \tag{16.1}$$

1. Activity-Selection Problem

We claim that $S_{ij} = \emptyset$ whenever $i \geq j$. Why?

1. Activity-Selection Problem

A_{ij}

$c[i, j]$

$c[i, j] = 0$ whenever $S_{ij} = \emptyset$;

$c[i, j] = 0$ for $i \geq j$.

1. Activity-Selection Problem

$$c[i, j] = c[i, k] + c[k, j] + 1 .$$

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

1. Activity-Selection Problem

Theorem 16.1

Consider any nonempty subproblem S_{ij}, and let a_m be the activity in S_{ij} with the earliest finish time:

$$f_m = \min \{ f_k : a_k \in S_{ij} \} .$$

Then

1. Activity a_m is used in some maximum-size subset of mutually 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.

1. Activity-Selection Problem

Proof We shall prove the second part first, since it's a bit simpler. Suppose that S_{im} is nonempty, so that there is some activity a_k such that $f_i \le s_k < f_k \le s_m < f_m$. Then a_k is also in S_{ij} and it has an earlier finish time than a_m, which contradicts our choice of a_m. We conclude that S_{im} is empty.

To prove the first part, we suppose that A_{ij} is a maximum-size subset of mutually compatible activities of S_{ij}, and let us order the activities in A_{ij} in monotonically increasing order of finish time. Let a_k be the first activity in A_{ij}. If $a_k = a_m$, we are done, since we have shown that a_m is used in some maximum-size subset of mutually compatible activities of S_{ij}. If $a_k \ne a_m$, we construct the subset $A'_{ij} = A_{ij} - \{a_k\} \cup \{a_m\}$. The activities in A'_{ij} are disjoint, since the activities in A_{ij} are, a_k is the first activity in A_{ij} to finish, and $f_m \le f_k$. Noting that A'_{ij} has the same number of activities as A_{ij}, we see that A'_{ij} is a maximum-size subset of mutually compatible activities of S_{ij} that includes a_m. ■

1. Activity-Selection Problem

```
RECURSIVE-ACTIVITY-SELECTOR(s, f, i, n)
m ← i + 1
while m ≤ n and s_m < f_i        ▷ Find the first activity in S_{i,n+1}.
      do m ← m + 1
if m ≤ n
   then return {a_m} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
   else return Ø
```

1. Activity-Selection Problem

GREEDY-ACTIVITY-SELECTOR(s, f)

```
n ← length[s]
A ← {a_1}
i ← 1
for m ← 2 to n
     do if s_m ≥ f_i
           then A ← A ∪ {a_m}
                i ← m
return A
```

1. Activity-Selection Problem

16.1-3

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.

(This is also known as the ***interval-graph coloring problem***. We can create an interval graph whose vertices are the given activities and whose edges connect incompatible activities. The smallest number of colors required to color every vertex so that no two adjacent vertices are given the same color corresponds to finding the fewest lecture halls needed to schedule all of the given activities.)

Greedy Steps

مراحل ارائه یک الگوریتم به روش حریصانه:

1. Determine the optimal substructure of the problem.
2. Develop a recursive solution.
3. Prove that at any stage of the recursion, one of the optimal choices is the greedy choice. Thus, it is always safe to make the greedy choice.
4. Show that all but one of the subproblems induced by having made the greedy choice are empty.
5. Develop a recursive algorithm that implements the greedy strategy.
6. Convert the recursive algorithm to an iterative algorithm.

Greedy Steps

- Greedy-choice property
- Optimal substructure