GREEDY ALGORITHMS

Analysis of Algorithms

Punjab University College of Information Technology (PUCIT)
University of the Punjab, Lahore, Pakistan.

Credit

- These notes contain material from Chapter 16 of Cormen, Leiserson, Rivest, and Stein (3rd Edition).

Greedy Algorithms

- A greedy algorithm always makes the choice that looks best at the moment.
- Locally optimal choice may lead to the globally optimal solution.
- Greedy algorithms works in way to maximize immediate benefit.
- Greedy algorithms do not always yield optimal solutions, but for many problems they do.

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 serve only one activity at a time

- Each activity a_i has a start time s_i and a finish time f_i, where $0 <= s_i < f_i$

- If selected, activity a_i takes place during the half-open 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. That is

a_i and a_j are compatible if $s_i >= f_j$ or $s_j >= f_i$

Activity Selection Problem (Cont.)

- We assume that the activities are sorted in monotonically increasing order of finish time:

$$f_1 \le f_2 \le f_3 \le \cdots \le f_{n-1} \le f_n$$

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	9	9	10	11	12	14	16

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

Are these combinations true?

1, 4, 8, 11,
2, 4, 8, 11,
3, 7, 11,
4, 1, 6, 11,
5, 11,
6, 1, 7, 11,
7, 1, 8, 11,
8, 1, 9, 11,
9, 1, 11,
10,
11, 1,

Recursive Solution

```
RECURSIVE-ACTIVITY-SELECTOR(s, f, k, n)
m = k + 1
while m ≤ n and s[m] < f[k]        // find the first activity in S_k to finish
    m = m + 1
if m ≤ n
    return {a_m} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
else return Ø
```

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	9	9	10	11	12	14	16

Activity Selection Problem (Cont.)

GREEDY-ACTIVITY-SELECTOR(s, f)

```
n = s.length
A = {a_1}
k = 1
for m = 2 to n
    if s[m] ≥ f[k]
        A = A ∪ {a_m}
        k = m
return A
```

How about the run time complexity of the algorithm?

Elements of the greedy strategy

1. Determine the optimal substructure of the problem.

2. Show that if we make the greedy choice, then only one subproblem remains.

3. Prove that it is always safe to make the greedy choice. Develop a recursive algorithm that implements the greedy strategy.

4. Convert the recursive algorithm to an iterative algorithm.

Greedy-choice Property

- We can assemble a globally optimal solution by making locally optimal (greedy) choices.
- In dynamic programming, we make a choice at each step, but the choice usually depends on the solutions to subproblems.
- We solve dynamic problem using bottom-up method wherever greedy algorithm usually solve using top-down method

Greedy vs. Dynamic Programming

- **Dynamic programming**
  - Make a choice at each step.
  - Choice depends on knowing optimal solutions to subproblems. Solve subproblems first.
  - Solve bottom-up.

- **Greedy**
  - Make a choice at each step.
  - Make the choice before solving the subproblems.
  - Solve top-down.

Knapsack Problem

0-1 Knapsack Problem:

A thief robbing a store finds n items. The i^{th} item is worth i dollars and weighs w_i pounds, where i and w_i are integers. The thief wants to take as valuable a load as possible, but he can carry at most W pounds in his knapsack, for some integer W. Which items should he take?

Fractional Knapsack Problem:

In the fractional knapsack problem, the setup is the same, but the thief can take fractions of items, rather than having to make a binary (0-1) choice for each item.

Knapsack Problem (Cont.)