CS F364: Design & Analysis of Algorithm

06 Greedy Algorithm Knapsack Problem

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Jan 29, 2021 ONLINE (Campus @ BITS-Pilani Jan-May 2021)

`http://ktiwari.in/algo`

Greedy Algorithm

- Strategy to solve constrained optimization problem
- Make sequence of choices [with] What looks best at the moment
- Incremental thus Efficient
- A greedy algorithm makes a **locally optimal choice** in the hope that the choice will lead to a **globally optimal** solution
- **Caution:** does NOT always yields optimal solutions
- Determine the problem has optimal substructure

Key ingredients

1. Greedy-choice property: we can assemble a globally optimal solution by making locally optimal (greedy) choices.
2. Optimal substructure: if an optimal solution to the problem contains within it optimal solutions to subproblems.

Activity Selection Problem

Activity Selection Problem:

- Consider a set $S = \{1, 2, 3, ..., n\}$ of n activities that can happen one activity at a time. Activity i takes place during interval $[s_i, f_i)$.
- Activity i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap
- Select maximum size set of mutually comparable activities.

Consider following set of activity

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

- $\{3, 9, 11\}$ is a compatible activity
- $\{1, 4, 8, 11\}$ is larger compatible activity. In fact it is the largest
- Another largest compatible activity is $\{2, 4, 9, 11\}$

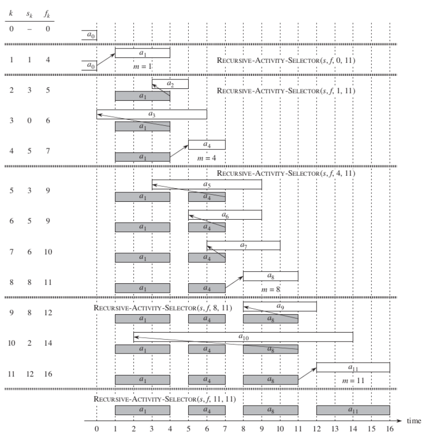
Activity Selection Problem

Assume that activities are in increasing order of their finishing time. If not, then sort it in $O(n \lg n)$ time.

Algorithm 1: Greedy-Activity-Selection(s, f)

```
n = length(s)
A = { 1 }
j = 1
for i = 2 to n do
    if s_i ≥ f_j then
        A = A ∪ { i }
        j = i
return A
```


Activity Selection Problem



Knapsack Problem

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

Fractional knapsack problem: He can take fraction of an items as well

0-1 knapsack problem: He can either take complete items or leave

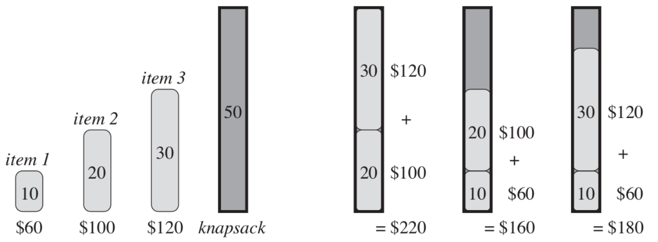
- Fractional knapsack problem can be solved with greedy but
- 0-1 knapsack problem needs Dynamic Programming

$$V(i, w) = max(V(i-1, w), V(i-1, w-w[i]) + P[i])$$

0-1 knapsack problem needs DP

Let knapsack can have 50kg

3 items of wt 10, 20, 30 of price Rs 60, 100 and 120 respectively



Thank You!

Thank you very much for your attention! (Reference[1])

Queries ?

[1] [1] Book - *Introduction to Algorithm*, By THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN