BITS, PILANI – K. K. BIRLA GOA CAMPUS

Design & Analysis of Algorithms

(CS F364)

Lecture No. 8

The 0-1 Knapsack Problem

Given: A set **S** of ***n*** items, with each item ***i*** having

w_i - a positive **"weight"**

v_i - a positive **"benefit"**

Goal:

Choose items with maximum total benefit but with weight at most ***W***.

And we are not allowed to take fractional amounts

In this case, we let ***T*** denote the set of items we take

Objective : **maximize** $\sum_{i \in T} v_i$

Constraint : $\sum_{i \in T} w_i \leq W$

Greedy approach

Possible greedy approach:

Approach1: Pick item with largest value first

Approach2: Pick item with least weight first

Approach3: Pick item with largest value per weight first

We know that none of the above approaches work

Exercise:

Prove by giving counterexamples.

Brute Force Approach

- **Brute Force**

The naive way to solve this problem is to go through all **2^n subsets** of the ***n*** items and pick the subset with a legal weight that **maximizes** the value of the knapsack.

Optimal Substructure

As we did before we are going to solve the problem in terms of sub-problems.

- Our first attempt might be to characterize a sub-problem as follows:

Let O_k be the optimal subset of elements from $\{I_0, I_1, \ldots, I_k\}$.

Observe

Optimal subset from the elements $\{I_0, I_1, \ldots, I_{k+1}\}$ may not correspond to the optimal subset of elements from $\{I_0, I_1, \ldots, I_k\}$

That means, the solution to the optimization problem for S_{k+1} might **NOT** contain the optimal solution from problem S_k.

where S_k: Set of items numbered ***1*** to ***k.***

Optimal Substructure

Example: Let **W=20**

Item	Weight	Value
I_0	3	10
I_1	8	4
I_2	9	9
I_3	8	11

- The best set of items from *$\{I_0, I_1, I_2\}$* is *$\{I_0, I_1, I_2\}$*
- BUT the best set of items from *$\{I_0, I_1, I_2, I_3\}$* is *$\{I_0, I_2, I_3\}$.*

Note

1. Optimal solution, *$\{I_0, I_2, I_3\}$ of S_3* for does NOT contain the optimal solution, *$\{I_0, I_1, I_2\}$* of *S_2*

2. *$\{I_0, I_2, I_3\}$* build's upon the solution, *$\{I_0, I_2\}$*, which is really the optimal subset of *$\{I_0, I_1, I_2\}$* with weight **12** or less.

(We incorporates this idea in our 2nd attempt)

Recursive Formulation

S_k: Set of items numbered ***1*** to ***k.***

Define **B[k,w]** to be the best selection from **S_k** with weight at most ***w***

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w],\ B[k-1,w-w_k]+b_k\} & \text{else} \end{cases}$$

Our goal is to find **B[n, W],** where n is the total number of items and W is the maximal weight the knapsack can carry.

This does have **Optimal Substructure property**.

Exercise – Prove Optimal Substructure property.

Optimal Substructure

- Consider a most valuable load ***L*** where $W_L \leq W$

Suppose we remove item ***j*** from this optimal load ***L***

The remaining load $L'_j = L - \{I_j\}$

must be a most valuable load weighing at most

$$W'_j = W - \{w_j\}$$

pounds that the thief can take from

$$S'_j = S - \{I_j\}$$

That is L'_j should be an optimal solution to the

0-1 Knapsack Problem(S'_j***,*** W'_j***)***

The 0-1 Knapsack Problem

Exercise

Overlapping Subproblems

Analysis

Pseudo- code

```
for w = 0 to W
        B[0,w] = 0
for i = 1 to n
        B[i,0] = 0
for i = 1 to n
   for w = 0 to W
        if w_i <= w          // item i can be part of the solution
            if b_i + B[i-1,w-w_i] > B[i-1,w]
                B[i,w] = b_i + B[i-1,w- w_i]
            else
                B[i,w] = B[i-1,w]
        else B[i,w] = B[i-1,w]          // w_i > w
```

The 0-1 Knapsack Problem

- How to find actual Knapsack Items?

if ***B[i,k] ≠ B[i−1,k]*** then

mark the ***ith*** item as in the knapsack

i = i−1, k= k- w_i

else

i = i−1 // Assume the ith item is not in the knapsack