UNIVERSITY INSTITUTE OF COMPUTING

MASTER OF COMPUTER APPLICATIONS

DESIGN AND ANALYSIS OF ALGORITHMS

24CAT-611

UNIT-2

DISCOVER . LEARN . EMPOWER

Outline

- **Greedy Method Technique:** Knapsack Problem

Knapsack Problem

Given a set of items, each with a weight and a value, determine a subset of items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

The **Knapsack Problem** is an combinatorial optimization problem. It appears as a subproblem in many, more complex mathematical models of real-world problems. One general approach to difficult problems is to identify the most restrictive constraint, ignore the others, solve a knapsack problem, and somehow adjust the solution to satisfy the ignored constraints.

Knapsack Problem (contd.)

In **Fractional Knapsack**, we can break items for maximizing the total value of knapsack. This problem in which we can break an item is also called the fractional knapsack problem.

In this case, items can be broken into smaller pieces, hence the thief can select fractions of items.

According to the problem statement,

- There are **n** items in the store
- Weight of **i^{th}** item $w_i > 0$
- Profit for **i^{th}** item $p_i > 0$ and
- Capacity of the Knapsack is **W**

Knapsack Problem (contd.)

In the Knapsack problem, items can be broken into smaller pieces. So, the thief may take only a fraction $\boldsymbol{x_i}$ of **i**$^{\mathbf{th}}$ item.

$$0 \leqslant x_i \leqslant 1$$

The **i**$^{\mathbf{th}}$ item contributes the weight $x_i.w_i$ to the total weight in the knapsack and profit $x_i.p_i$ to the total profit.

Hence, the objective of this algorithm is to

$$maximize \sum_{n=1}^{n}(x_i . pi)$$

It is clear that an optimal solution must fill the knapsack exactly, otherwise we could add a fraction of one of the remaining items and increase the overall profit.

Knapsack Problem (contd.)

Thus, an optimal solution can be obtained by

$$\sum_{n=1}^{n}(x_i . wi) = W$$

In this context, first we need to sort those items according to the value of

$$\frac{p_i}{w_i} \text{ , so that } \frac{p_i+1}{w_i+1} \leq \frac{p_i}{w_i} \text{ .}$$

Here, ***x*** is an array to store the fraction of items.

Algorithm-Knapsack Problem

Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)

```
for i = 1 to n
do
x[i] = 0
weight = 0
for i = 1 to n
if weight + w[i] ≤ W then
x[i] = 1
weight = weight + w[i]
else
x[i] = (W - weight) / w[i]
weight = W
break
return x
```

Analysis-Knapsack Problem

If the provided items are already sorted into a decreasing order of $\frac{p_i}{w_i}$,

then the while loop takes a time in ***O(n)***;
Therefore, the total time including the sort is in ***O(n log n)***.

Example-Knapsack Problem

Let us consider that the capacity of the knapsack ***W* = 60** and the list of provided items are shown in the following table −

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
Ratio $\left(\frac{p_i}{w_i}\right)$	7	10	6	5

As the provided items are not sorted based on $\mathbf{\frac{p_i}{w_i}}$

Example-Knapsack Problem

After sorting, the items are as shown in the following table.

Item	B	A	C	D
Profit	100	280	120	120
Weight	10	40	20	24
Ratio $(\frac{p_i}{w_i})$	10	7	6	5

Example-Knapsack Problem

After sorting all the items according to $\frac{p_i}{w_i}$.

First all of ***B*** is chosen as weight of ***B*** is less than the capacity of the knapsack.
Next, item ***A*** is chosen, as the available capacity of the knapsack is greater than the weight of ***A***.
Now, ***C*** is chosen as the next item.
However, the whole item cannot be chosen as the remaining capacity of the knapsack is less than the weight of ***C***.
Hence, fraction of ***C*** (i.e. (60 − 50)/20) is chosen.

Example-Knapsack Problem

Now, the capacity of the Knapsack is equal to the selected items. Hence, no more item can be selected.

The total weight of the selected items is **10 + 40 + 20 * (10/20) = 60**

And the total profit is **100 + 280 + 120 * (10/20) = 380 + 60 = 440**

This is the optimal solution.

We cannot gain more profit selecting any different combination of items.

Applications of Knapsack Problem

In many cases of resource allocation along with some constraint, the problem can be derived in a similar way of Knapsack problem.

Following is a set of example.

- Finding the least wasteful way to cut raw materials
- portfolio optimization
- Cutting stock problems

References

1) https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_fractional_knapsack.htm/

2) **Data Structures and Algorithms made easy** By *Narasimha Karumanchi.*

3) The Algorithm Design Manual, 2nd Edition by Steven S Skiena

4) **Fundamentals of Computer Algorithms - Horowitz and Sahani**

THANK YOU