Greedy Algorithm

Knapsack Problem

Optimization Problems

- Finding out the optimal solution from all feasible solutions.
- **Solution Space:** Set of all solution over the given n input.
- **Feasible Solution:** Solutions from the solution space which will satisfy the constraints (conditions).
- **Optimal Solution:** One of the feasible solution which optimizes our goal.
- Optimization problem can be either maximization problem or minimization problem.
- Techniques used to solve Optimization Problems:
  - Greedy Algorithms
  - Dynamic Algorithms
  - Branch and Bound

Greedy Algorithm

- The idea of a greedy algorithm is to make the choice that looks best at that moment.
- Most of the problems in greedy technique contain ‘n’ inputs (solution space) and our objective is finding a subset which will produce optimal solution.
- Does not cover all combinations (takes less time).
- Does not always yield optimal solution.

Greedy Algorithm: Example

Problem: Want to travel from Delhi to Mumbai in 18 hours with minimum cost.

Solution Space: on foot, Bike, Bus, Car, Train, Flight.

Feasible Solution: Train, Flight (18 hours)

Optimal Solution: Train (minimum cost)

Greedy Algorithm: Applications

1. Knapsack Problem (fractional)
2. Huffman Code
3. Activity Selection Problem
4. Minimum Spanning Trees :
   - Prim’s Algorithm
   - Kruskal’s Algorithm
5. Single Source Shortest Paths:
   - Dijkstra's Algorithm
   - Bellman Ford

Knapsack Problem (Fractional)

Problem: We are given 'n' objects and a knapsack or bag. Object *i* has a weight w_i and the knapsack has a capacity ***W***. If fraction x_i, $0 \leq x_i \leq 1$, is placed into the knapsack, then a profit of $x_i p_i$ is earned. The objective is to obtain a filling of the knapsack that maximizes the total profit earned. Since the knapsack capacity is **W**, we require the total weight of all chosen objects to be at most **W**.

Type of Problem: Optimization problem → maximization problem

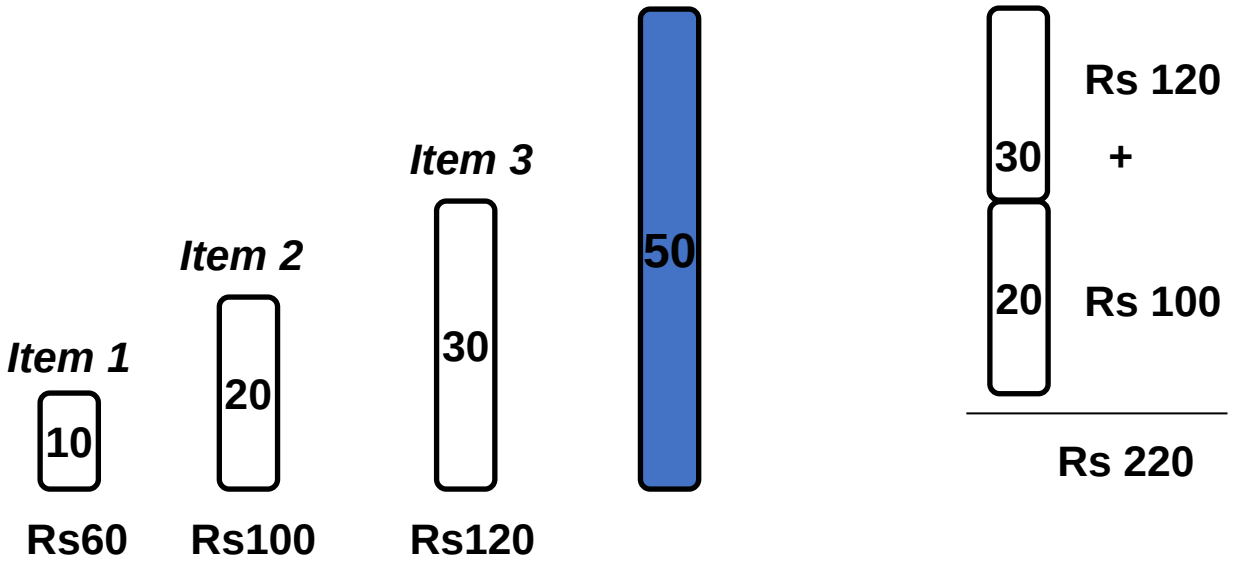
Formal Representation:

Problem: $\sum_{i=1}^{n} w_i > W$

Feasible Solution: $\sum_{i=1}^{n} x_i w_i \leq W$

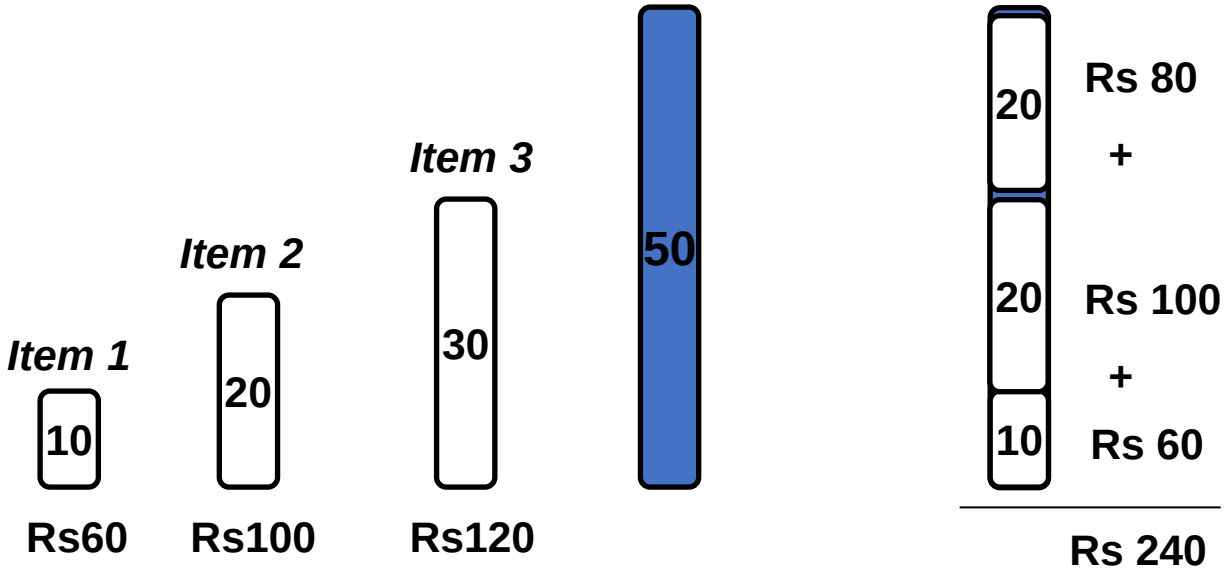
Optimal Solution: $\sum_{i=1}^{n} x_i p_i \quad (should\ be\ maximum)$

Fractional Knapsack – Example 1



Fractional Knapsack – Example 2

- *E.g.:*



Knapsack Problem: Algorithm

Step 1: Find $\frac{p_i}{w_i}$ for all the objects. →O(n)

Step 2: Sort the array based on $\frac{p_i}{w_i}$ in decreasing order. →O(nlogn)

Step 3: Take one by one object until knapsack size becomes zero. →O(n)

```
1.  for i =1 to n
2.    do x[i] =0
3.  weight = 0
4.  while weight < W
5.    do i = best remaining item
6.    if weight + w[i] ≤ W
7.      then x[i] = 1
8.      weight = weight + w[i]
9.    else
10.     x[i] = (w - weight) / w[i]
11.     weight = W
12. return x
```

Time complexity: O (nlogn)

n=3 M=20

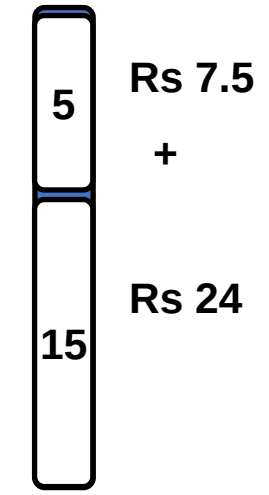
Object	Object1	Object2	Object3
Profit	25	24	15
Weight	18	15	10

Problem 1

```
1.  for i =1 to n
2.    do x[i] =0
3.  weight = 0
4.  while weight < W
5.    do i = best remaining item
6.    if weight + w[i] ≤ W
7.      then x[i] = 1
8.      weight = weight + w[i]
9.    else
10.     x[i] = (w - weight) / w[i]
11.     weight = W
12. return x
```

Number of objects n=3,
knapsack size W=20

Objects i	Object 1	Object 2	Object 3
Profits p[i]	25	24	15
Weights w[i]	18	15	10
	25/18=1.39	**24/15=1.6**	**15/10=1.5**



Rs 7.5

+

Rs 24

Rs 31.5

Problem 2

```
1.  for i =1 to n
2.    do x[i] =0
3.  weight = 0
4.  while weight < W
5.    do i = best remaining item
6.    if weight + w[i] ≤ W
7.      then x[i] = 1
8.      weight = weight + w[i]
9.    else
10.     x[i] = (w - weight) / w[i]
11.     weight = W
12. return x
```

Objects i	Object 1	Object 2	Object 3	Object 4	Object 5
Profits p[i]	25	55	45	20	35
Weights w[i]	5	15	12	7	6

Number of objects n=5,
Knapsack size W=3.5

Problem 3

```
1.  for i =1 to n
2.    do x[i] =0
3.  weight = 0
4.  while weight < W
5.    do i = best remaining item
6.    if weight + w[i] ≤ W
7.      then x[i] = 1
8.      weight = weight + w[i]
9.    else
10.     x[i] = (w - weight) / w[i]
11.     weight = W
12. return x
```

Objects i	Object 1	Object 2	Object 3	Object 4	Object 5	Object 6	Object 7
Profits p[i]	10	5	15	7	6	18	3
Weights w[i]	2	3	5	7	1	4	1

Number of objects n=7,
Knapsack size W=15.

n=5 M=15

Object	Object1	Object2	Object3	Object4	Object5
Profit	2	28	25	18	9
Weight	1	4	5	3	3

Thank You