Greedy Algorithms

Optimization Problems

- Maximization
- Minimization

Key Concepts

- To solve an optimization problem, begin by drawing a picture and introducing variables.
- Find an equation relating the variables.
- Find a function of one variable to describe the quantity that is to be minimized or maximized.

Greedy Algorithms

- **Greedy algorithms** aim to make the optimal choice at that given moment. (gives a sub-optimal solution)
- Each step it chooses the optimal choice, without knowing the future. It attempts to find the globally optimal way to solve the entire problem using this method.

Finding the Path With Maximum Reward **(Example)**

Suppose we have a robot that is placed at cell `(0, 0)` of an `m * n` grid.

The robot has to navigate the grid and reach its goal position, while collecting a reward from each cell it passes through.

The aim of navigation is to follow a path that maximizes the reward through the grid.

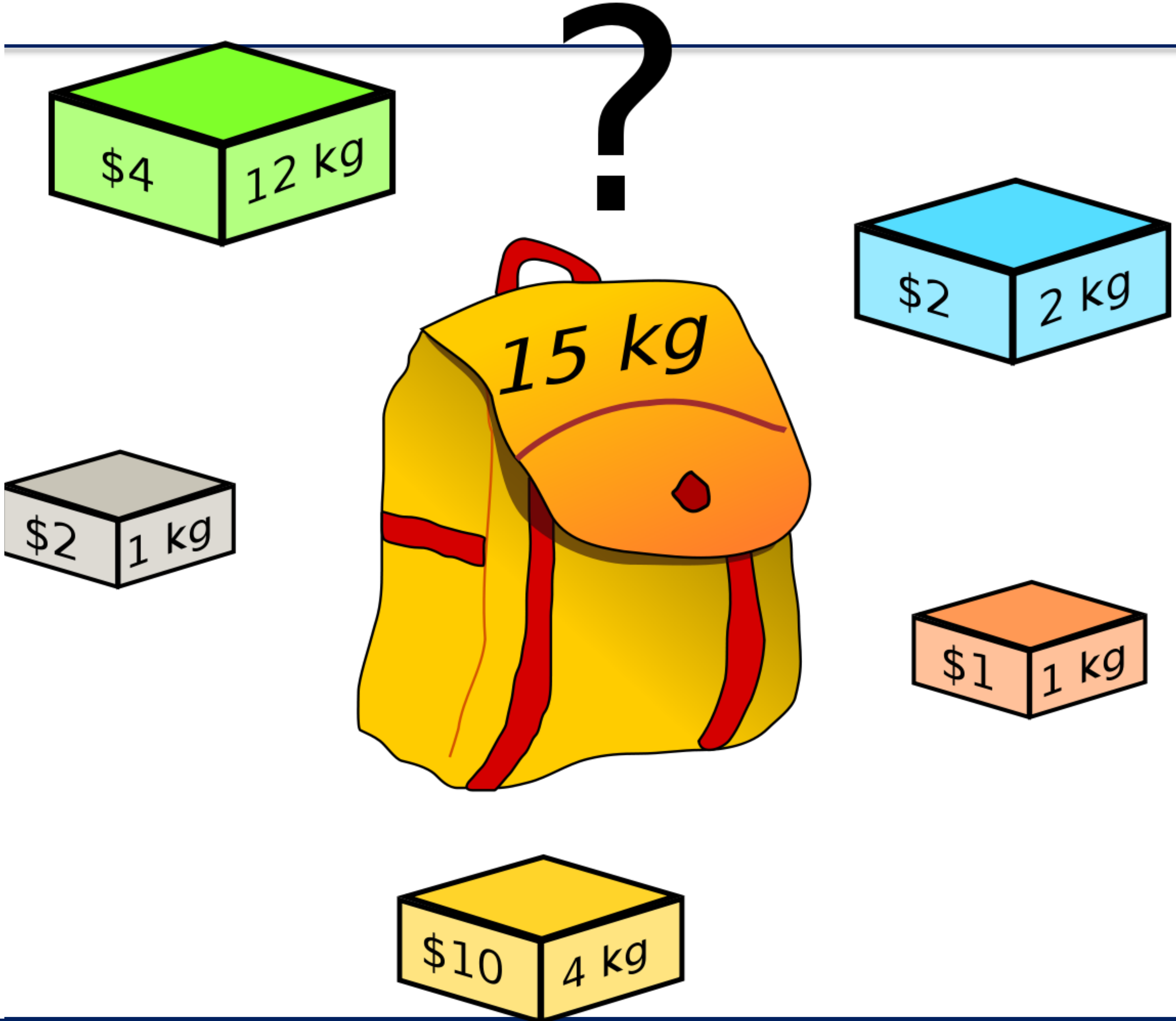
The only legal moves allowed are an "up" move and a "right" move.

Knapsack problem

The problem states:

Which items should be placed into the knapsack such that: the value or profit obtained by putting the items into the knapsack is maximum.

“under our constrain (the Knapsack size)”



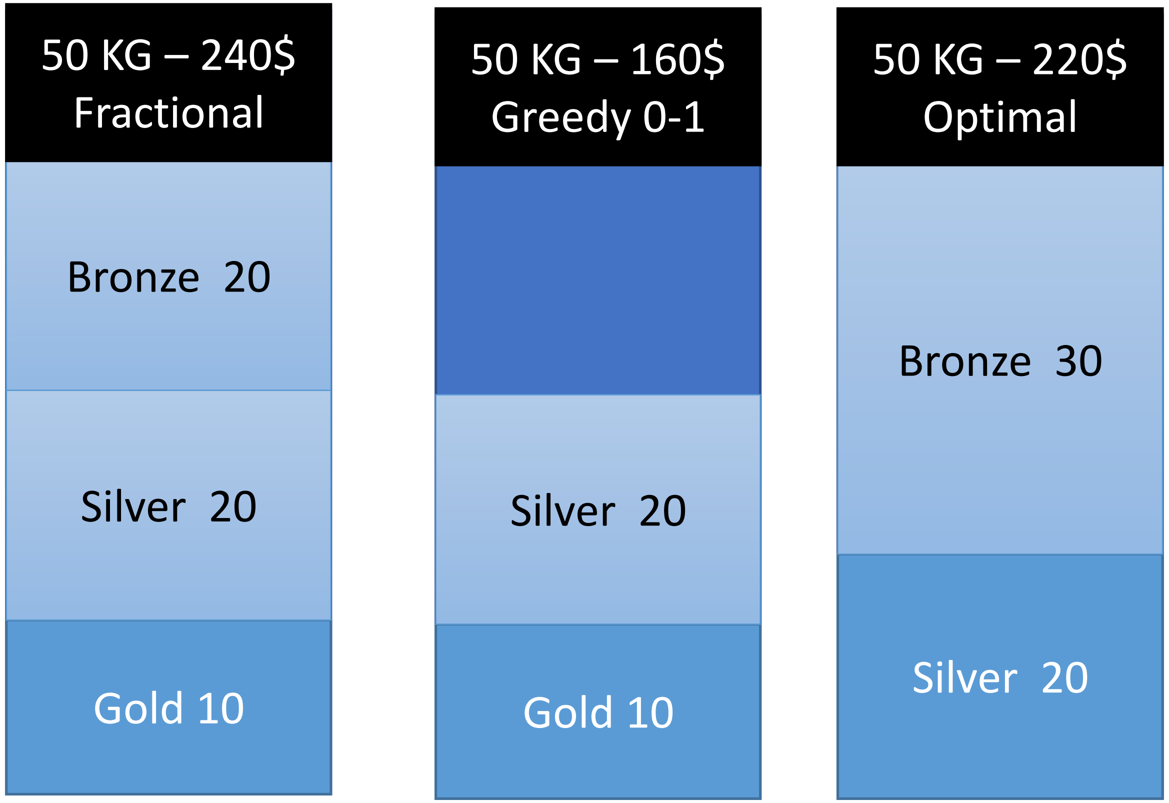
Knapsack problem

- Objective function: value in the knapsack

  - Factional
    - You can take any fraction of any item.
  - 0-1
    - You either take the whole item or leave it.

Knapsack problem - Example

Item	Weight KG	Value$	Value$/KG
• Gold	10	60	6$
• Silver	20	100	5$
• Bronze	30	120	4$



50 KG – 240$
Fractional

Bronze 20

Silver 20

Gold 10

50 KG – 160$
Greedy 0-1

Silver 20

Gold 10

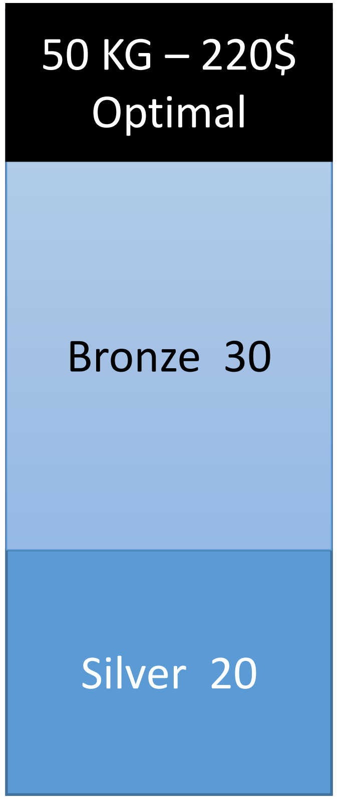
50 KG – 220$
Optimal

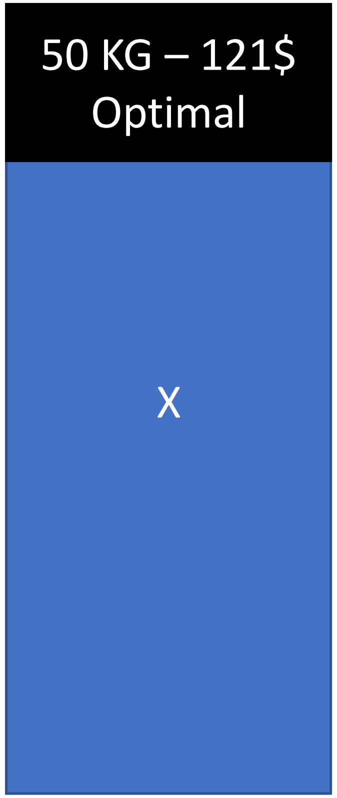
Bronze 30

Silver 20

Knapsack problem - Example

Item	Weight KG	Value$
Gold	10	60
Silver	20	100
Bronze	30	120
X	**50**	**121**





Another greedy Strategy. The item with the highest value

There is no known greedy algorithm solves 0-1 Knapsack problem

Greedy fails because no future insight and no back track

0/1 Knapsack Problem

- In 0/1 Knapsack Problem,
- As the name suggests, items are indivisible here.
- We can not take the fraction of any item.
- We have to either take an item completely or leave it completely.
- It is solved using dynamic programming approach.

0/1 Knapsack Problem

- **Time Complexity**

- Each entry of the table requires constant time θ(1) for its computation.
- It takes θ(nw) time to fill (n+1)(w+1) table entries.
- It takes θ(n) time for tracing the solution since tracing process traces the n rows.
- Thus, overall θ(nw) time is taken to solve 0/1 knapsack problem using dynamic programming.

Fractional Knapsack Problem

- In Fractional Knapsack Problem,
- As the name suggests, items are divisible here.
- We can even put the fraction of any item into the knapsack if taking the complete item is not possible.
- It is solved using Greedy Method.

Fractional Knapsack Problem

- **Time Complexity**

- The main time taking step is the sorting of all items in decreasing order of their value / weight ratio.
- If the items are already arranged in the required order, then while loop takes O(n) time.
- The average time complexity of **Quick Sort** is O(nlogn).
- Therefore, total time taken including the sort is O(nlogn).