Algorithms: Greedy Method

Knapsack Problem

Dr. Md. Abul Kashem Mia, Professor, CSE Dept, BUET

Greedy Algorithms: Principles

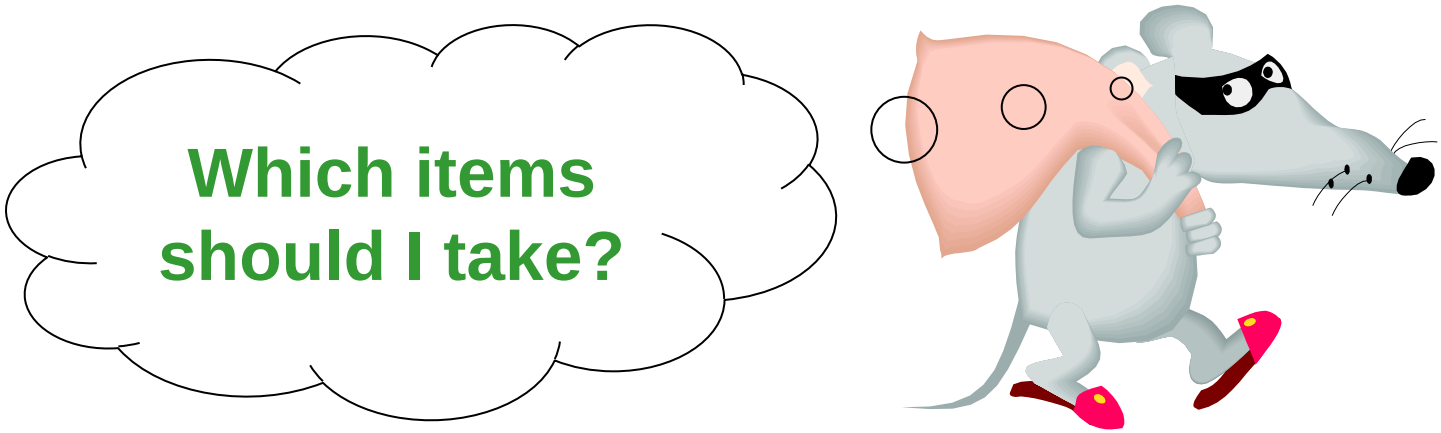
- A *greedy algorithm* always makes the choice that looks best at the moment.
- A greedy algorithm works in phases.

  At each phase:
  - You take the best you can get right now, without regard for future consequences.
  - You hope that by choosing a local optimum at each step, you will end up at a global optimum.
  - For some problems, it works.

The Knapsack Problem

The famous *Knapsack Problem*:

A thief breaks into a museum. Fabulous paintings, sculptures, and jewels are everywhere. The thief has a good eye for the value of these objects, and knows that each will fetch hundreds or thousands of dollars on the clandestine art collector's market. But, the thief has only brought a single knapsack to the scene of the robbery, and can take away only what he can carry. What items should the thief take to maximize the haul?

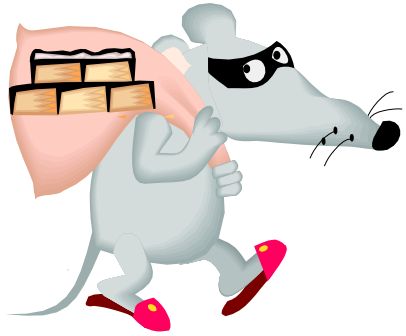


The Knapsack Problem

There are two versions of the problem:

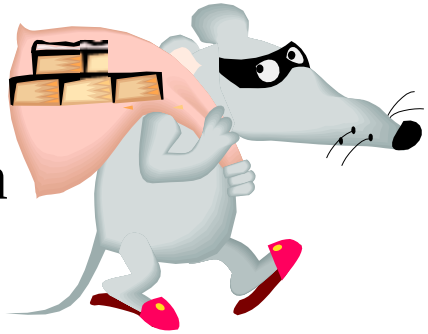
(1) "0-1 knapsack problem"

Items are indivisible: you either take an item or not. Solved with *dynamic programming*.



(2) "Fractional knapsack problem"

Items are divisible: you can take any fraction of an item. Solved with a *greedy algorithm*.

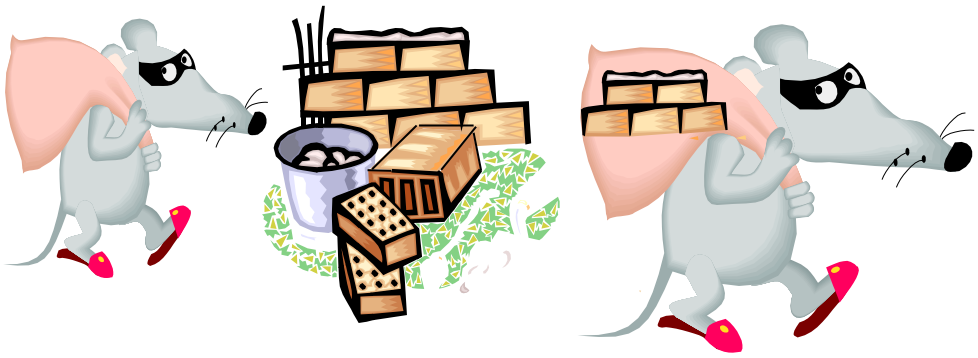


The Knapsack Problem

- More formally, the *0-1 knapsack problem*:
  - The thief must choose among n items, where the ith item worth v_i dollars and weighs w_i pounds
  - Carrying at most W pounds, maximize value
    - Note: assume v_i, w_i, and W are all integers
    - "0-1" because each item must be taken or left in entirety
- A variation, the *fractional knapsack problem*:
  - Thief can take fractions of items
- Think of items in 0-1 problem as gold ingots, in fractional problem as buckets of gold dust.

Optimal Substructure Property

- Both problems exhibit the optimal substructure property.
- To show this for both the problems, consider the most valuable load weighing at most W pounds
  - Q: If we remove item j from the load, what do we know about the remaining load?
  - A: The remaining load must be the most valuable load weighing at most $W - w_j$ that the thief could take from the n-1 original items excluding item j.



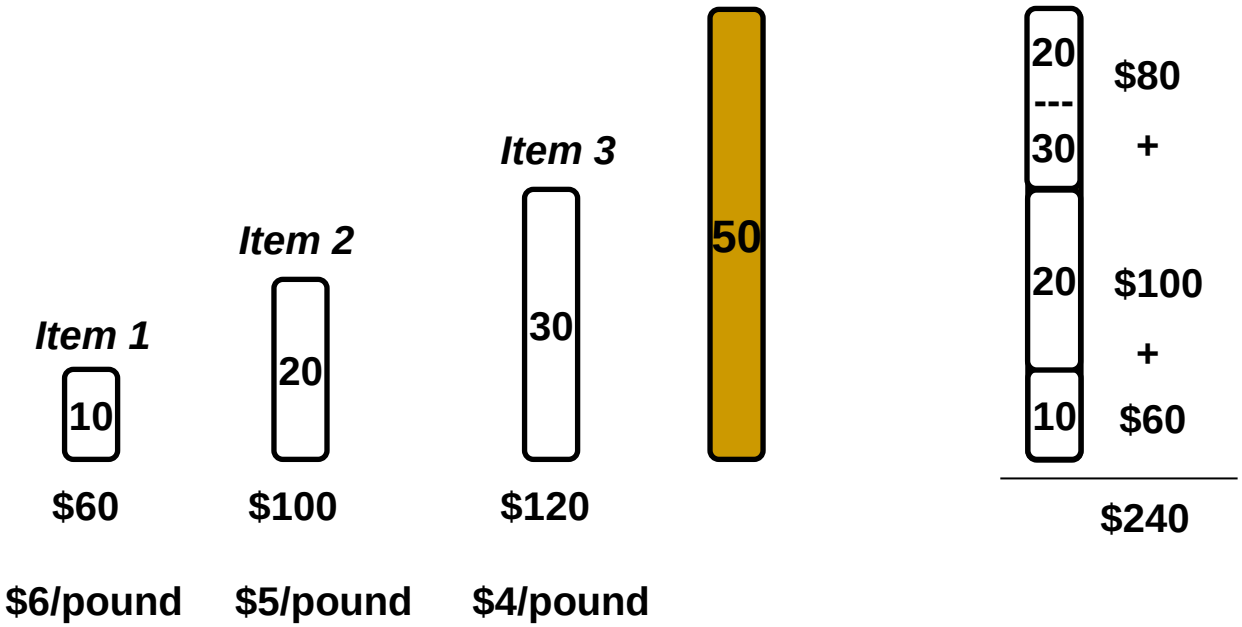
Fractional Knapsack Problem

- Knapsack capacity: W
- There are n items: the *i*-th item has value v_i and weight w_i
- Goal:
  - find x_i such that for all $0 \leq x_i \leq 1, \quad i = 1, 2, .., n$

    $\sum w_i x_i \leq W$ and

    $\sum x_i v_i$ is maximum

Fractional Knapsack - Example



Fractional Knapsack Problem

Greedy strategy:

- Pick the item with the maximum value per pound v_i/w_i
- If the supply of that element is exhausted and the thief can carry more: take as much as possible from the item with the next greatest value per pound
- It is good to order items based on their value per pound

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n}$$

Fractional Knapsack Problem

Alg.: Fractional-Knapsack (W, $v[n]$, $w[n]$)

1. while $w > 0$ and as long as there are items remaining
2. pick item with maximum v_i/w_i
3. $x_i \leftarrow \min(1, w/w_i)$
4. remove item i from list
5. $w \leftarrow w - x_i w_i$

- w – the amount of space remaining in the knapsack (initially $w = W$)
- Running time: $\Theta(n)$ if items already ordered; else $\Theta(n \log n)$

0-1 Knapsack problem

- Thief has a knapsack with maximum capacity *W*, and a set *S* consisting of *n* items
- Each item *i* has some weight w_i and benefit value v_i (all w_i , v_i and *W* are integer values)
- Problem: How to pack the knapsack to achieve maximum total value of packed items?
- Goal:

find x_i such that for all $x_i \in \{0, 1\}$, $i = 1, 2, .., n$

$\sum w_i x_i \leq W$ and

$\sum x_i v_i$ is maximum

0-1 Knapsack - Greedy Strategy Fails