CS 2009
Design and Analysis of Algorithms

Waheed Ahmed

Farrukh Salim Shaikh

Optimization Problems

- For most optimization problems you want to find, not just ***a*** solution, but the ***best*** solution.
- A ***greedy algorithm*** sometimes works well for optimization problems. It 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.

THE GREEDY PARADIGM

Commit to choices one-at-a-time,

never look back,

and hope for the best.

Greedy Approach

- Like dynamic programming, used to solve optimization problems.
- Problems exhibit optimal substructure (like DP).
- Problems also exhibit the **greedy-choice** property.
    - When we have a choice to make, make the one that looks best *right now*.
    - Make a **locally optimal choice** in hope of getting a **globally optimal solution**

Greedy Approach

- Does **greedy-choice** property lead us to optimize solution?
- 

- **Greedy algorithms** mostly (but not **always**) fail to find the globally **optimal solution** because they usually **do** not operate exhaustively on all the data.

Coin Change Problem

Another problem with a greedy solution!

Coin Change Problem

- Goal: Given currency denominations: 1, 5, 10, 25, 100, devise a method to pay amount to customer using fewest number of coins.

- A greedy algorithm to do this would be:
  At each step, take the largest possible bill or coin that does not overshoot.
- For US money, the greedy algorithm always gives the optimum solution

Coin Change Problem

- Goal: Given currency denominations: 1, 5, 10, 25, 100, devise a method to pay amount to customer using fewest number of coins.

Ex. 34¢.

Cashier's algorithm. At each iteration, add coin of the largest value that does not take us past the amount to be paid.

Ex. $2.89.

Coin Change Problem

CASHIERS-ALGORITHM $(x, c_1, c_2, \ldots, c_n)$

SORT n coin denominations so that $c_1 < c_2 < \ldots < c_n$

$S \leftarrow \phi$ ⟵ set of coins selected

WHILE $x > 0$

  $k \leftarrow$ largest coin denomination c_k such that $c_k \leq x$

  IF no such k, RETURN "no solution"

  ELSE

    $x \leftarrow x - c_k$

    $S \leftarrow S \cup \{ k \}$

RETURN S

Knapsack Problem : Greedy approach

- Greedy approach is also good for fractional knapsack problem but not for 0/1 knapsack.

- **Fractional knapsack** In which you can take fraction of item if you want

- **0/1 knapsack** In which you can only take complete item or leave it but you cannot take fraction of it

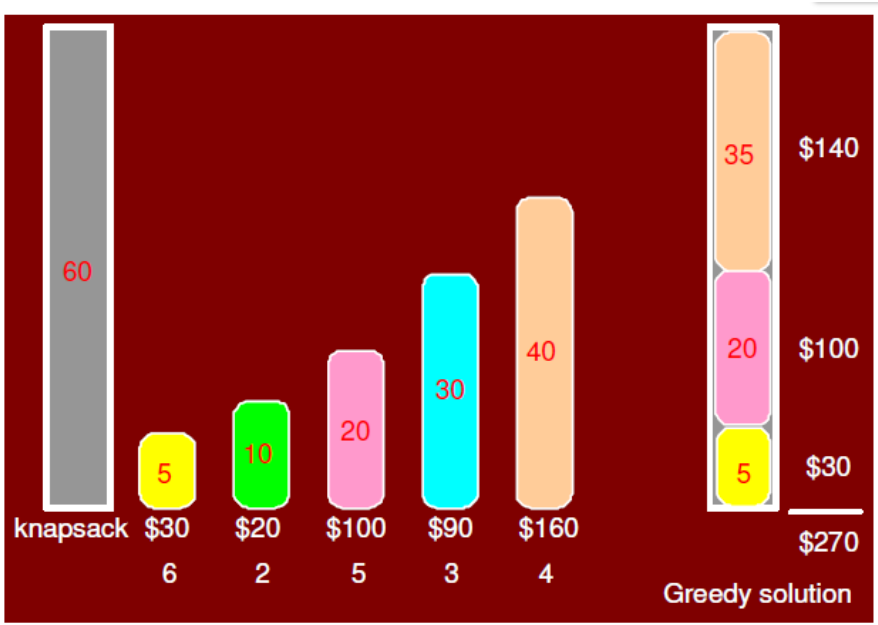
Knapsack Problem : Greedy approach

- Suppose “i” item has value “v(i)” and weight “w(i)”. Capacity of knapsack (bag) is W. Then greedy approach would be :
- Sort in decreasing order of value/weight i-e v(i)/w(i)
- Now start selecting items till you fill the bag.

- In fractional knapsack, you utilize complete capacity W because you can take fraction of items but in 0/1 knapsack, you may or may not.

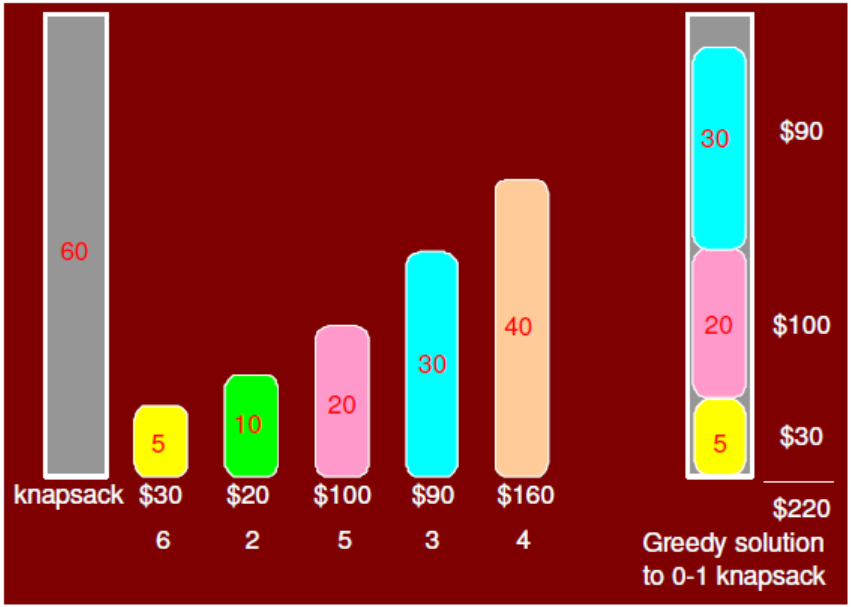
Knapsack Problem : Greedy approach

Greedy/optimal solution to fractional knapsack is :



Knapsack Problem : Greedy approach

- Greedy solution to 0/1 knapsack :



Knapsack Problem : Greedy approach

- Optimal/non-greedy solution to 0/1 knapsack :