Learn DAA : From B K Sharma

TCS-503: Design and Analysis of Algorithms

Advanced Design and Analysis Techniques: Greedy Algorithms

Learn DAA: From B K Sharma

Advanced Algorithm Design Techniques: Optimization Techniques

- Dynamic Programming
- Greedy Algorithms
- Backtracking

Advanced Algorithm Analysis Technique

- Amortized Analysis

Learn DAA: From B K Sharma

Advanced Algorithm Design Techniques: Optimization Techniques

Dynamic Programming

Optimal Substructure Property + Overlapping Substructure Property

Programming means "**Tabular Method**", not computer programming.

Works for more problems.

In between, not as fast as greedy.

Greedy Algorithms

Greedy Choice Property + Optimal Substructure Property

Can be applied to few problems.

but gives fast algorithms.

Backtracking

Can be applied to almost all problems.

but gives very slow algorithms.

Why Greedy Algorithm?

No direct solution available.

Easy-to-implement solutions to complex, multi-step problems.

When Greedy Algorithm?

When the problem has:

Greedy Choice Property + Optimal Substructure Property

a.k.a.

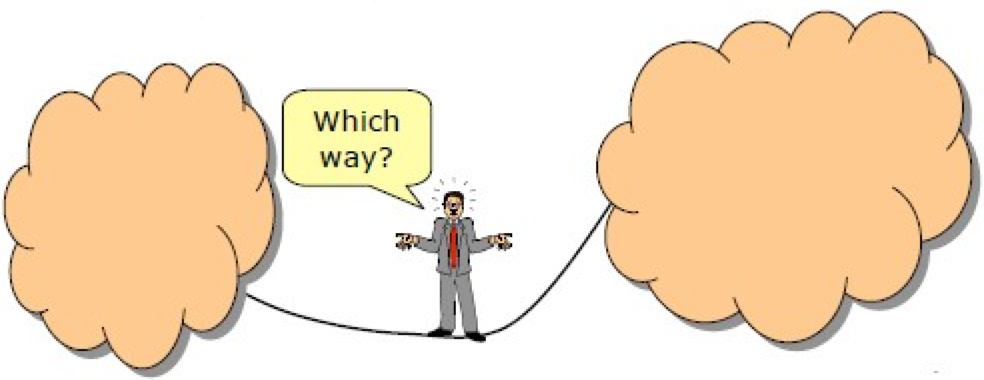
Elements of GA

A good clue that a greedy strategy will solve the problem.

Why Greedy Algorithm?

Hill-Climbing

You want to reach the summit, but can only see 10 yards ahead and behind (due to thick fog).



Greedy Choice Property

When we have a choice to make, make the one that looks best right now.

A locally greedy choice will lead to a globally optimal solution.

Make a locally optimal choice in hope of getting a globally optimal solution.

A globally optimal solution can be arrived at by making a locally optimal (greedy) choice.

Learn DAA: From B K Sharma

How Greedy Algorithm? What are the Steps of Greedy Algorithm?

1. Formulate the optimization problem in the form:

   we make a choice and we are left with one sub-problem to solve.

2. Show that the greedy choice can lead to an optimal solution:

   so that the greedy choice is always safe.

3. Demonstrate that an optimal solution to original problem =

   greedy choice + an optimal solution to the sub-problem

4. Make the greedy choice and solve top-down.

5. May have to preprocess input to put it into greedy order : e.g. Sorting activities by finish time.

Learn DAA: From B K Sharma

Problems to be solved using Techniques of Greedy Algorithm.

- Fractional Knapsack Problem
- Activity Selection Problem
- Huffman Code

Two Types of Knapsack Problem

- Binary(0/1) Knapsack Problem
  - Dynamic Programming Gives Optimal Solution
- Fractional Knapsack Problem
  - Techniques of Greedy Algorithm Gives Optimal Solution.

Learn DAA: From B K Sharma

Knapsack Problems

Binary (0/1) Knapsack Problem

A thief rubbing a store finds n items:

- the i^{th} item is worth v_i dollars and weights w_i pounds (v_i, w_i integers)

The thief can only carry W pounds in his knapsack

The thief can either not take an item or take whole item

Which items and how much should the thief take to maximize the value of his load?

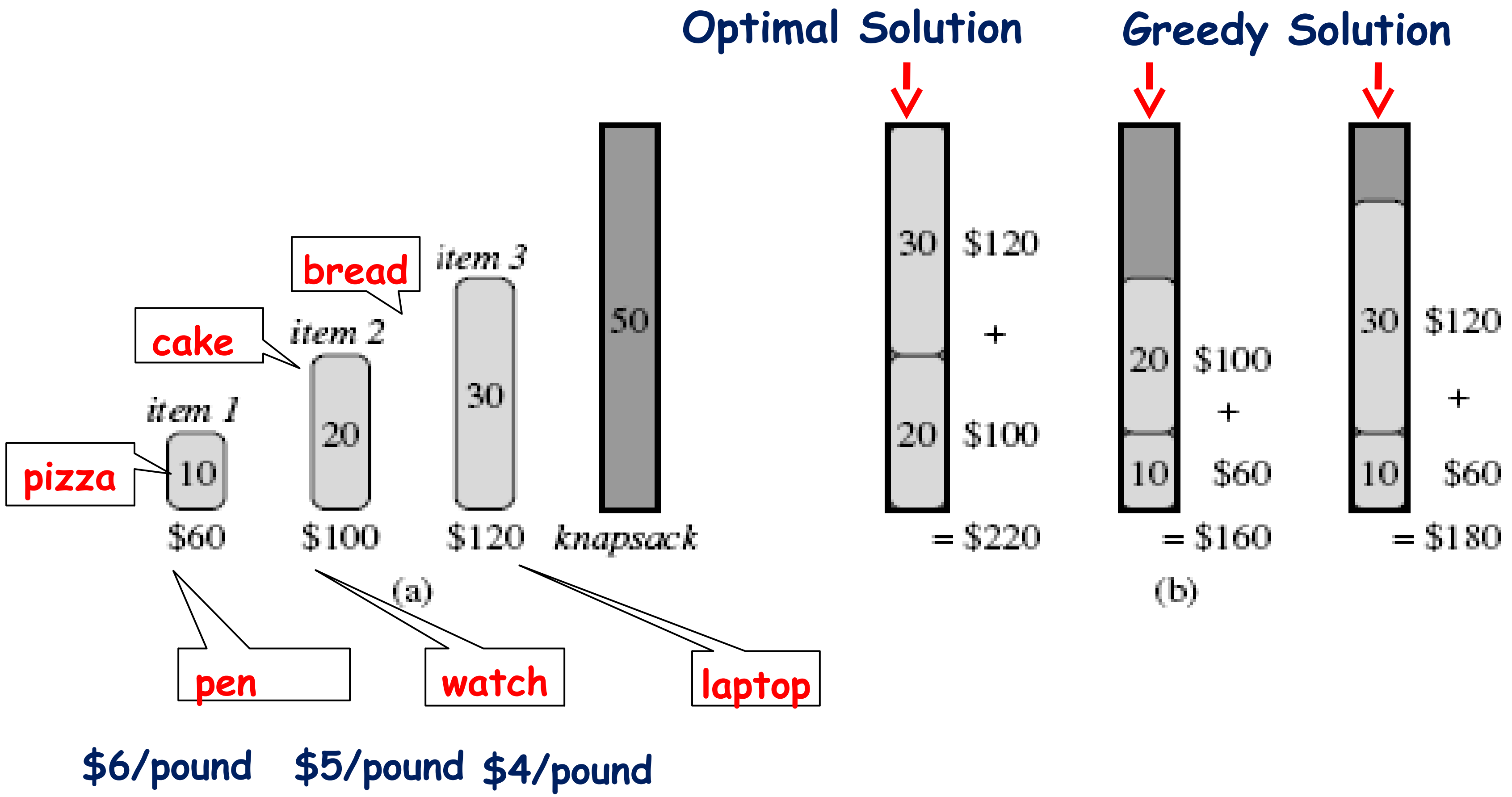
Fractional Knapsack Problem

Same as Binary (0/1) Knapsack Problem but:-

The thief can take fractions of items

Learn DAA: From B K Sharma

0-1 Knapsack - Greedy Strategy



- None of the solutions involving the greedy choice (item 1) leads to an optimal solution
  - The greedy choice property does not hold

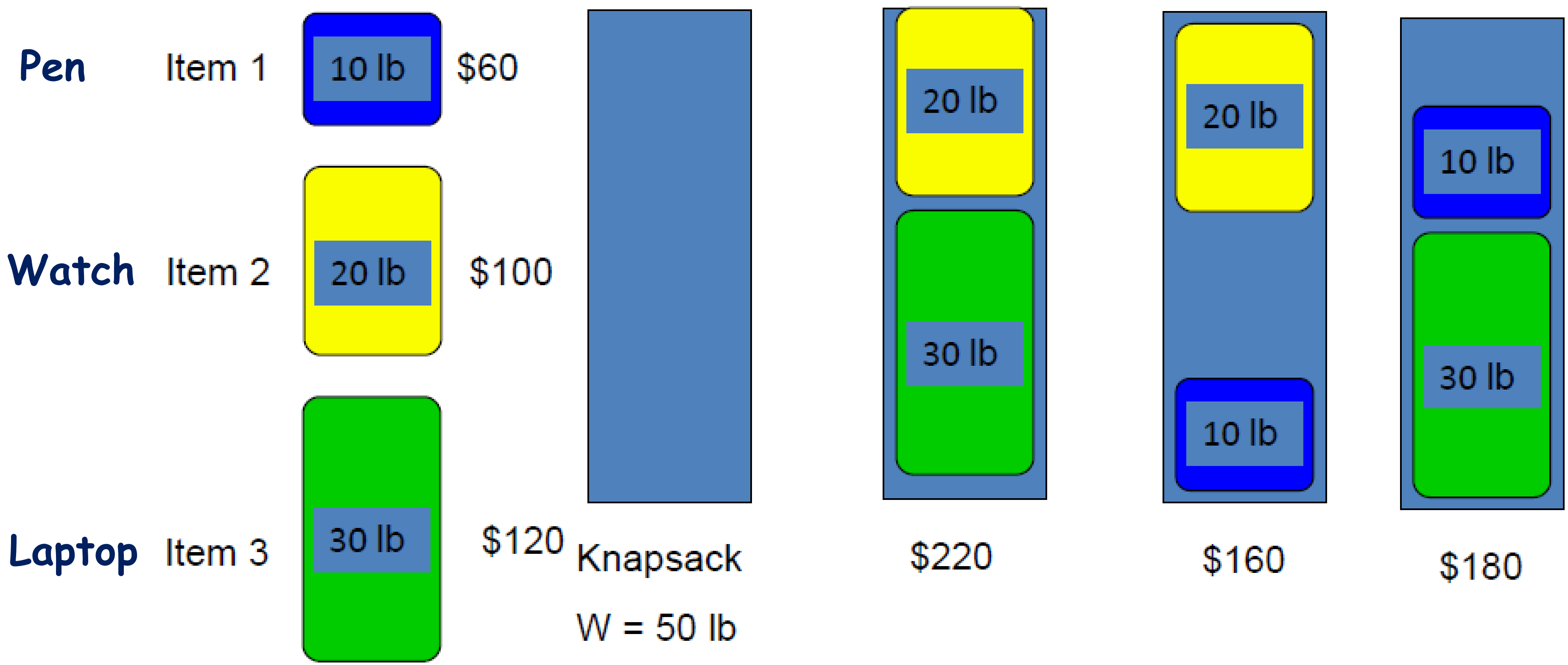
Learn DAA: From B K Sharma

Knapsack Problems

	Weight (w_i)	Cost(v_i)	v_i/w_i
Item 1	10 lb	\$60	\$6/lb
Item 2	20 lb	\$100	\$5/lb
Item 3	30 lb	\$120	\$4/lb

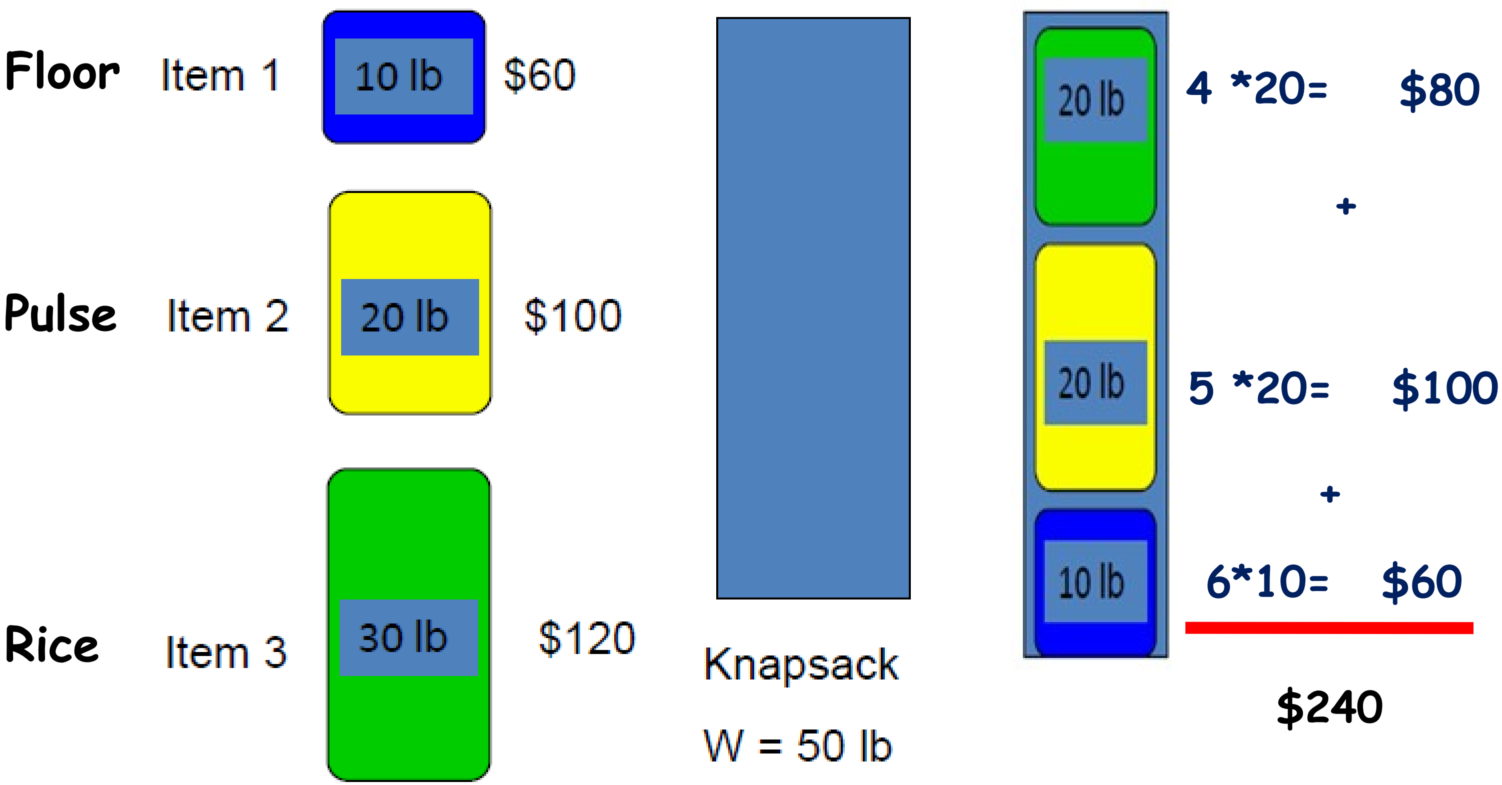
Learn DAA: From B K Sharma

Binary (0/1) Knapsack Problems



Learn DAA: From B K Sharma

Fractional Knapsack Problems



Question

What are the steps used in dynamic programming approach?

Discuss the (0/1) Knapsack Problem with respect to dynamic programming?

Is Greedy method equally applicable for the above problem?

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

. $x_i \leftarrow \min(1, w/w_i)$

. remove item i from list

. $w \leftarrow w - x_i w_i$

w – the amount of space remaining in the knapsack (w = W)

Learn DAA: From B K Sharma

Fractional Knapsack Problem

	Weight (w_i)	Cost(v_i)	v_i/w_i
Item 1	10 lb	$60	$6/lb
Item 2	20 lb	$100	$5/lb
Item 3	30 lb	$120	$4/lb

$w=W=50$

$w>0$ and item remains

Pick first item because $v_1/w_1=6$ which is maximum.

$x_1=\min(1, 50/10)=\min(1,5)=1$

$w = w - x_1w_1 = 50- 1*10=50-10=40$

$w>0$ and item remains

Pick second item since $v_2/w_2=5$

$x_2=\min(1,40/20)=\min(1,2)=1$

$w=w-x_2w_2=40-1*20=40-20=20$

$w>0$ and item remains

Pick the third item

$x_3=\min(1,20/30)=\min(1, 0.66)=0.66$

$w=w-x_3w_3=20-0.66*30=20-19.8=0.2$

$w>0$ and no item remaining

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. $w \leftarrow w - x_iw_i$

w – the amount of space remaining in the knapsack (w = W)

Optimal Load=10+20+19.8 =49.8 lbs

Optimal Cost=10* 6 +20 * 5 +19.8*4 =$239.2

Steps Toward Our Greedy Solution

1. Determine the optimal substructure of the problem
2. Develop a recursive solution
3. Prove that one of the optimal choices is the greedy choice
4. Show that all but one of the sub-problem resulted by making the greedy choice are empty.
   - For example if greedy choice is x_i then first take that item and we skip all other items that are not compatible with the x_i
5. Develop a recursive algorithm that implements the greedy strategy.
6. Convert the recursive algorithm to an iterative one