EECS 3101 M: Design and Analysis of Algorithms

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/3101M`
Also on Moodle

The Knapsack Problem

- Given different items (w_i, v_i), $i = 1, \ldots, n$, take as much of each as required so that:
  - The total weight capacity W of the knapsack is not exceeded
  - The payoff V from the items is maximized
- Two versions:
  - Continuous – can take real-valued amounts of each item
  - Discrete or 0/1 – each item must be taken or not taken (no fractional quantities)
- A simple greedy algorithm works for the continuous version (Ch 16)
  Algorithm: Take as much as possible of the most valuable item and continue until the capacity is filled

0/1 Knapsack: the Greedy Algorithm Fails

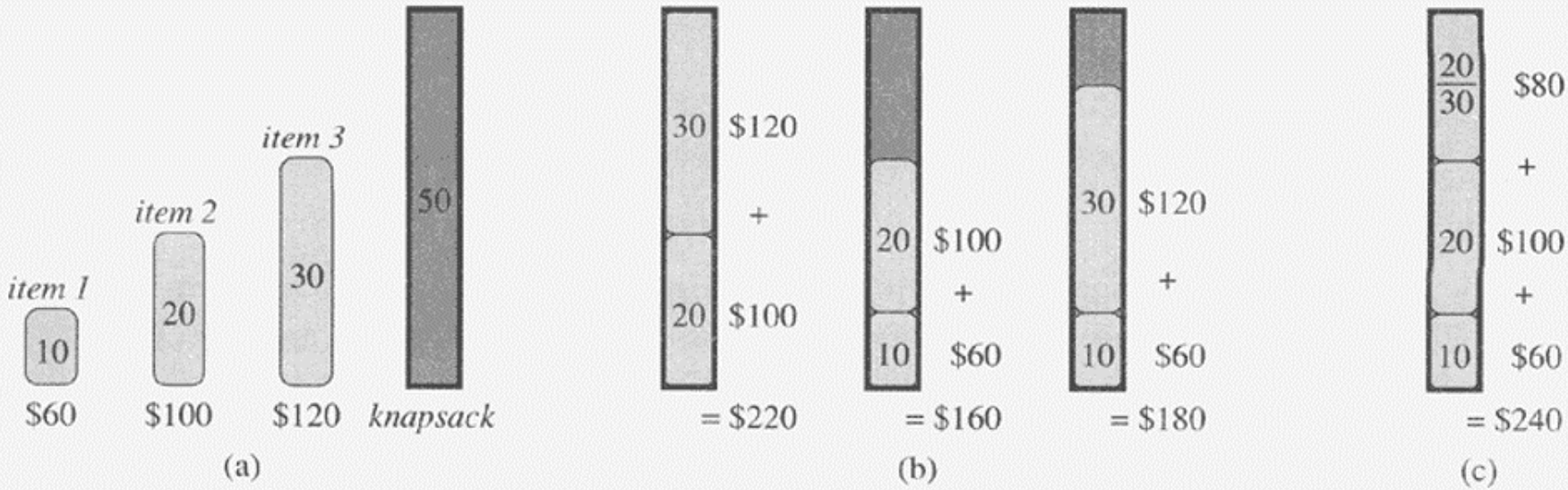

Figure 16.2 The greedy strategy does not work for the 0-1 knapsack problem. **(a)** The thief must select a subset of the three items shown whose weight must not exceed 50 pounds. **(b)** The optimal subset includes items 2 and 3. Any solution with item 1 is suboptimal, even though item 1 has the greatest value per pound. **(c)** For the fractional knapsack problem, taking the items in order of greatest value per pound yields an optimal solution.

0/1 Knapsack: Optimal Substructure and Recurrence

- Optimal substructure: Suppose we know that item n is selected. Then the solution of the subproblem for capacity $W - w_n$ must be optimal
- Subproblem: $c[i, w] =$ max value of knapsack of capacity w using items 1 through i
- Recurrence:
  $c[i, w] = 0$ if $i = 0$ or $w = 0$
  $c[i, w] = c[i - 1, w]$ if $w_i > w, i > 0$
  $c[i, w] = \max[v_i + c[i - 1, w - w_i], c[i - 1, w]]$ if $i > 0, w \geq w_i$

0/1 Knapsack: Details

- Which array element has the final solution? $c[n, W]$
- Which array elements can be initialized directly? $c[i, w]$ for $i = 0$ or $w = 0$
- What order should the table be filled?
- Complexity?
  Is this a polynomial time algorithm?
- How do you get the actual solution?

More Dynamic Programming Problems

- Longest increasing subsequence
- Coin changing
- Snowboarding problem
- More problems in homework, tutorials

Longest Increasing Subsequence

To apply dynamic programming, we have to:

- Given an array of distinct integers, to find the longest increasing subsequence.
- Subproblems?
- Recurrence?
- Alternative Solution: Use LCS!

Coin changing

- Given an amount and a set of denominations, to make change with the fewest number of coins.
- Subproblems?
- Recurrence?

More Difficult Problem

- The snow boarding problem : Find the longest path on a grid. One can slide down from one point to a connected other one if and only if the height decreases. One point is connected to another if it's at left, at right, above or below it.
- Example:

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

- What order to fill the table?

Back to a Grid Problem

Counting number of paths in a grid with blocked intersections

- Not an optimization problem

- Similar strategy to previous problems