COMP3600/COMP6400 - Algorithms

Design Methodologies: Review and Applications

August 19, 2016

1 Divide-and-Conquer

Linear-Time Selection: The running time of the Linear-Selection Algorithm in the textbook is linear regardless of n elements being divided into groups of size 5 (see lecture 7) or groups of size 17 (see tutorial 1). Is the algorithm still linear for groups of size 9? Justify your answer.

2 Dynamic Programming

Edit distance: When a spell checker encounters a possible misspelling, it looks in its dictionary for other words that are close by. What is the appropriate notion of "closeness" in this case?

A natural measure of *the distance* between two strings is the extent to which they can be aligned, or matched up. The distance between two strings is the cost of their best possible alignment, assuming that the score is calculated as follows:

- 1 for a match,
- -1 for a mismatch,
- -2 for a gap.

For example, $X = SNOWY$ and $Y = SUNNY$ can be aligned like this:

```
S  -  N  O  W  Y
S  U  N  N  -  Y
----------------
1 -2  1 -1  -2 1
----------------
Cost: 1+(-2)+1+(-1)+(-2)+1=-2
```

In the above example, another alignment is as follows:

```
-  S  N  O  W  -  Y
S  U  N  -  -  N  Y
-------------------------------------
Cost: -8
```

Describe an $O(mn)$-time algorithm to solve this problem with an objective of maximising the alignment cost, assuming that $m = |X|$ and $n = |Y|$.

3 Greedy Algorithms

3.1 Fractional Knapsack Problem

Consider the diagram. A thief must fill his knapsack with the most valuable load as possible. The item can be split into parts as given in diagram (c). The greedy strategy works in this case, which in turn allows the problem to be solved in $O(n \log n)$ time. Devise a greedy algorithm for it.

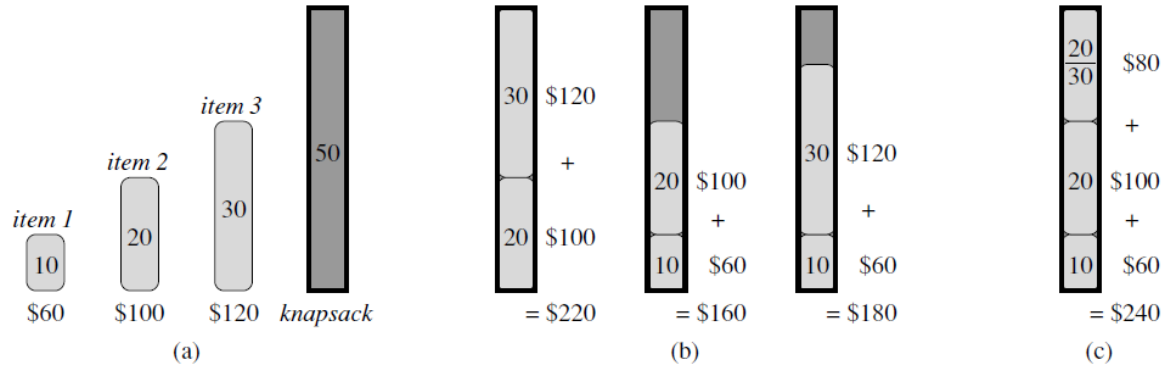

Figure 16.2 The greedy strategy does not work for the 0-1 knapsack problem. **(a)** The thief must select a subset of the three items shown whose weight must not exceed 50 pounds. **(b)** The optimal subset includes items 2 and 3. Any solution with item 1 is suboptimal, even though item 1 has the greatest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

4 Some Harder Problems

4.1 Revisiting Linear-Time Selection

Suppose that an algorithm uses only comparisons to find the ith smallest element in a set of n elements. Show that it can also find the $i-1$ smaller elements and the $n-i$ larger elements without any additional comparisons.

4.2 Coin exchanging problem

Consider the Australian coin denominations with $2c$ and $1c$ coins only. Now, we want to make change for n cents with *the fewest number of coins*.

- Describe a greedy algorithm to solve this.
- Prove that the algorithm is optimal.
- Suppose we only have $20c$, $5c$ and $2c$ coins. Does your algorithm still work for changing $78c$? If not, is there another greedy algorithm that will work? Justify your answer.

4.3 Stock market investment problem

15-10 (Textbook, 3rd edition, p.410-411)

4.4 Scheduling to maximize the profit

Suppose you have one machine and a set of n jobs $J_1, J_2, \ldots, J_n$ to process on that machine. Each job J_i has a processing time t_i, a profit p_i and a deadline d_i. The machine can process only one job at a time, and job J_i must run uninterruptedly for t_i consecutive time units. If job J_i is completed by its deadline d_i, you receive a profit p_i, but if it is completed after its deadline, you receive a profit of 0. Give an algorithm to find a schedule that deliverers the maximum profit, assuming that all processing times are integers between 1 and n. What is the running time of your algorithm?