Homework #4

Deadline 11:59pm Saturday 7 December 2013

*Please present a **printed** report with all your answers, explanations, and sample plots. Submit also a soft copy of the source code and binaries used to generate these results. Please note that copying of any results or source code will result in ZERO credit for the whole homework.*

Make sure to read chapters 15 and 16 of the **CLRS** textbook before attempting the questions.

Problem 1

Prove that the Dynamic Programming solution developed in the lectures for the **0-1 Knapsack Problem** exhibits the *optimal substructure* property.

Recall that the optimal value for a knapsack of weight w considering the first k items is:

$$B[k,w]=\begin{cases} B[k-1,w] & \text{if } w_k>w \\ \max\left(B[k-1,w],B[k-1,w-w_k]+b_k\right) & \text{otherwise}\end{cases}$$

In particular, you need to prove that if $B[k, w]$ is optimal solution for the first k items for weight w, then $B[k-1, w]$ is also the optimal solution for the first $k-1$ items and weight w in the first case, … etc.

Problem 2

Consider the 0-1 Knapsack Problem:

1. For n = 5, 10, 50, 100, 500, 1000, 10000 generate n items with random weights as integers from 1 to 10. Assume that $W = 100$.
2. Implement the dynamic programming algorithm (above) using the *bottom-up* approach in C++, and run it on each of the randomly generated problems in (1).
3. Implement in C++ the greedy algorithm (where at each step the item with the highest value per weight is chosen), and run on each of the problems from (1). Plot the values obtained from the solution of (2) versus values obtained from (3).

Problem 3

You have an input text consisting of a sequence of n words of lengths $l_1, l_2, \dots, l_n$, where the length of a word is the number of characters it contains. Your printer can only print with its built-in Courier 10-point fixed-width font set that allows a maximum of M characters per line. (Assume that $l_i \le M$ for all $i = 1 \dots n$). When printing words i and $i+1$ on the same line, one space character (blank) must be printed between the two words. In addition, any remaining space at the end of the line is padded with blanks. Thus, if words i through j are printed on a line, the number of extra space characters at the end of the line (after word j) is

$$B(i,j)=M-j+i-\sum_{k=i}^{j} l_k$$

There are many ways to divide a paragraph into multiple lines. To produce nice-looking output, we want a division that fills each line as much as possible. A heuristic that has empirically shown itself to be effective is to charge a cost of the ***cube*** of the number of extra space characters at the end of each line. To avoid the unnecessary penalty for extra spaces on the last line, however, the cost of the last line is 0. In other words, the cost LineCost(i, j) for printing words i through j on a line is given by

$$LineCost(i,j)=\begin{cases}\infty & \text{if words } i \text{ through } j \text{ do not fit on a line}\\ 0 & \text{if } j=n \text{(i.e. last line)}\\ B(i,j)^3 & \text{otherwise}\end{cases}$$

The total cost for typesetting a paragraph is the sum of the costs of all lines in the paragraph. An optimal solution is a division of the n words into lines in such a way that the total cost is minimized.

1. Write down an expression for the objective to be minimized (the total cost of typesetting a paragraph).
2. Show that it exhibits optimal substructure i.e. the optimal solution of the problem contains optimal solutions to sub-problems.
3. Recursively define the value of an optimal solution in terms of solutions to sub-problems.
4. Describe a method to compute $B(i, j)$ in $O(1)$ time for any i and j after a pre-processing step. How much space is required and how much time is spent in the pre-processingstep?
5. Describe a dynamic programming algorithm to efficiently solve the problem. Analyze its space and time requirements.
6. Implement the algorithm using C++ and run it on the input samples in the attached data file using M = 40 and M = 72. Your algorithm should take M as a command line argument, read the input from `stdin`, and outputs the optimal solution (minimum cost) and the optimal placement of words on lines on the `stdout`. (Assume that spaces separate words i.e. any string of characters between two spaces is a word). See the example below. Include your code and output from the samples in your report.
7. If the LineCost above is modified to use the number of blanks instead of the cube of the number of blanks, show that the problem can be solved by a greedy algorithm. Describe the greedy choice, and argue that any optimal solution can be converted to the greedy solution. [Hint: what is the property of the greedy solution, and how can you convert any other solution to it such that the cost does not change?]

Example

The output for M=40 on Sample2.txt is below. Each line starts with the line number, followed by the number of characters that are occupied in that line e.g. 35 denotes that this line has 35 characters i.e. 5 blanks are inserted at the end.

```
COST = 2026
1:[35] The first practical mechanized type
2:[36] casting machine was invented in 1884
```

```
3:[37] by Ottmar Mergenthaler. His invention
4:[38] was called the "Linotype". It produced
5:[37] solid lines of text cast from rows of
6:[37] matrices. Each matrice was a block of
7:[36] metal -- usually brass -- into which
8:[34] an impression of a letter had been
9:[39] engraved or stamped. The line-composing
10:[32] operation was done by means of a
11:[35] keyboard similar to a typewriter. A
12:[37] later development in line composition
13:[32] was the "Teletypewriter". It was
14:[36] invented in 1913. This machine could
15:[37] be attached directly to a Linotype or
16:[39] similar machines to control composition
17:[39] by means of a perforated tape. The tape
18:[40] was punched on a separate keyboard unit.
19:[36] A tape-reader translated the punched
20:[39] code into electrical signals that could
21:[38] be sent by wire to tape-punching units
22:[40] in many cities simultaneously. The first
23:[35] major news event to make use of the
24:[31] Teletypewriter was World War I.
```

Instructions

Please submit a soft copy of the solutions together with source code and binaries in one zip file. The file should be named as **CMP461.HW##.First.Last.zip** where HW## is the homework number e.g. HW01, First and Last are your first and last name. So, if your name is Mohamed Aly, and this is homework #1, the file should be named **CMP461.HW01.Mohamed.Aly.zip.** Failing to follow these instructions will cost you points.

Acknowledgment: Some problems are adapted from Erik Demaine.