CS1510 Greedy Problems 9, 10 & 13

Rebecca Negley, Sean Myers

September 14, 2011

9. The input to this problem consists of an ordered list of n words. The lenght of the ith word is w_i, that is the ith word takes up w_i spaces. (For simplicity assume that there are no spaces between words.) The goal is to break this ordered list of words into lines, this is called a layout. Note that you can not reorder the words. The length of a line is the sum of the lengths of the words on that line. The ideal line length is L. No line may be longer than L, although it may be shorter. The penalty for having a line of length k is $L - K$. *The total penalty is the* **sum** *of the line penalties.*
The problem is to find a layout that minimizes the total penalty.
Prove or disprove that the following greedy algorithm correctly solves this problem.

    For i = 1 to n
        Place the ith word on the current line if it fits
            else place the ith word on a new line

Solution: Let's assume that this greedy algorithm, which we will name Most Crammed Line (MCL) is incorrect. To be incorrect, it means that there exists some input I for which MCL does not produce the optimal output. Suppose $OPT(I)$ is the optimal solution for input I that agrees with $MCL(I)$ for the greatest number of words. Let word w_k be the first word that $MCL(I)$ puts on a different line from $OPT(I)$. Suppose w_k is on line L_a in $MCL(I)$. Since $OPT(I)$ and $MCL(I)$ are identical until word w_k, all lines L_i such that $i < a$ are identical in $OPT(I)$ and $MCL(I)$. $OPT(I)$ cannot put w_k on line L_{a-1} because, if it could, $MCL(I)$ would have put w_k on line L_{a-1}. $OPT(I)$ cannot put w_k on line L_a because then $OPT(I)$ would not differ from $MCL(I)$ on the kth word. If $OPT(I)$ puts w_k on line L_i where $i > a + 1$, there will be no words on the $(a + 1)$st line, which would not be optimal. Hence, $OPT(I)$ must put w_k on the next line, L_{a+1}.

Now let's create $OPT'(I)$ and take word w_k from L_{a+1} and put it at the end of L_a. (We know there is room to do this because $MCL(I)$ puts it there.) Clearly, $OPT'(I)$ agrees with $MCL(I)$ for one more step. We need to make sure that $OPT'(I)$ is still optimal. In order to do that, we look at the two penalties of the lines that we have altered: $Penalty(OPT(I)_a)$, $Penalty(OPT(I)_{a+1})$, $Penalty(OPT'(I)_a)$, and $Penalty(OPT'(I)_{a+1})$. If $OPT'(I)$ is still optimal, then:

$$Penalty(OPT'(I)_a) + Penalty(OPT'(I)_{a+1}) \leq Penalty(OPT(I)_a) + Penalty(OPT(I)_{a+1}) \qquad (1)$$

Observe that

$$Penalty(OPT'(I)_a) = Penalty(OPT(I)_a) - w_k,$$

$$Penalty(OPT'(I)_{a+1}) = Penalty(OPT(I)_{a+1}) + w_k,$$

since the only change being made to each is removing w_k from line L_{a+1} (which increases the cost by w_k) and adding w_k to L_a, which reduces its cost by the same amount. Then, $Penalty(OPT'(I)_a) + Penalty(OPT'(I)_{a+1}) = Penalty(OPT(I)_a) - w_k + Penalty(OPT(I)_{a+1}) + w_k = Penalty(OPT(I)_a) + Penalty(OPT(I)_{a+1})$, which gives us (1).

Since $OPT'(I)$ is still optimal and it agrees with $MCL(I)$ for one more word than $OPT(I)$ agrees with $MCL(I)$, it violates our original assumtion that $OPT(I)$ agrees with $MCL(I)$ the maximum amount of words. Therefore, the MCL algorithm is optimal.

10. The input to this problem consists of an ordered list of n words. The lenght of the ith word is w_i, that is the ith word takes up w_i spaces. (For simplicity assume that there are no spaces between words.)
The goal is to break this ordered list of words into lines, this is called a layout. Note that you can not reorder the words. The length of a line is the sum of the lengths of the words on that line. The ideal line length is L. No line may be longer than L, although it may be shorter. The penalty for having a line of length k is $L - K$. *The total penalty is the* **maximum** *of the line penalties.*
The problem is to find a layout that minimizes the total penalty.
Prove or disprove that the following greedy algorithm correctly solves this problem.

```
For i = 1 to n
    Place the ith word on the current line if it fits
        else place the ith word on a new line
```

Solution: The MCL algorithm will not solve this problem. If we have words of length 5, 4 and 3 with the max $L = 10$, then MCL will produce a suboptimal result compared to the optimal solution. The optimal solution would be to place 5 on its own line, incurring a cost of 5, and then have 4 + 3 on it's own line having a cost of 3. The total cost would be 5. MCL would cram as much as it could onto one line before continuing. By doing so, it could fit 5 and 4 onto the first line, producing a cost of 1. The next line would just have 3, which would produce a cost of 7. The total would be 7, which is greater than 5. Hence, the MCL algorith does not always produce optimal output and does not solve the problem.

13. Consider the following problem.
INPUT: Positive integers $r_1, ..., r_n$ and $c_1, ..., c_n$
OUTPUT: An n by n matrix A with 0/1 entries such that for all i the sum of the ith row in A is r_i and the sum of the ith column in a is c_i, if such a matrix exists.
Think of the problem this way. You want to put pawns on an n by n chessboard so that the ith row has r_i pawns and the ith column has c_i pawns.
Consider the following greedy algorithm that constructs A row by row. Assume that the first i-1 rows have been constructed. Let a_j be the number of 1's in the jth column in the first $i-1$ rows. Now the r_i columns with the maximum $c_j - a_j$ are assigned 1's in row i, and the rest of the columns are assigned 0's. That is, the columns that still needs the most 1's are given 1's. Formally prove that this algorithm is correct.

Solution: Let's assume that this algorithm is incorrect. In the case of being incorrect, there must be some input I in which some row or column of the output matrix has a sum that differs from the designated integer for that row or column. For this input I, choose the optimal solution $OPT(I)$ that agrees with the solution given by the greedy algorithm for the greatest number of steps. Say k is the first step where $Greedy(I)$ chooses to place a 1 in a position where $OPT(I)$ contains a 0. (Note that, by its design, $Greedy$ places a 1 onto the matrix at each step - assuming everything is initially 0. Therefore, we do not need to consider a step where $Greedy(I)$ places a 0 where $OPT(I)$ contains a 1.) Say k is in row i and column q.

Choose a column p in row i where $OPT(I)$ contains a 1 and $Greedy(I)$ has not yet put a 1. This has to exist for row i in $OPT(I)$ to have the appropriate number of 1s, since $Greedy(I)$ has placed a 1 where $OPT(I)$ was 0. Construct $OPT'(I)$, which is initially identical to $OPT(I)$. In $OPT'(I)$, swap points (i,p) and (i,q), making it so (i,p) = 0, while (i,q) = 1. When this is done, there is now an imbalance in $OPT'(I)$ that needs to be corrected, since we have added an extra 1 to the sum of column q and removed 1 from the sum of p. In order to do this, we need to find some row further down (since opt and greedy are in accord in rows less than i), let's call it row j, in $OPT'(I)$ where column q contains a 1. Row j has to exist because if there is an optimal solution, column q needs have the correct number of 1s. Since $Greedy(I)$ agreed with $OPT(I)$ before row i and $Greedy(I)$ had not filled column q as of i, $OPT(I)$ had not either. Since $OPT(I)$ put a 0 at (i,q), it had to fill column q after row i.

In $OPT'(I)$, let $(j, q) = 0$ and $(j, p) = 1$. The sum of row j will not change since (j, q) and (j, p) increase and decrease by equal amounts. Then j has the exact opposite effect of row i, in which column q has one less 1 and column p has one more 1, balancing out the effects of row i. This means that $OPT'(I)$ is still an optimal solution and agrees with $Greedy(I)$ for one additional step, invalidating the statement that $OPT(I)$ agrees with $Greedy(I)$ for the greatest number of steps. This means that the greedy algorithm is indeed correct.