

Algoritmer og datastrukturer: Opgave om dynamisk programmering*

Besvares enkeltvis og afleveres senest fredag den 29. maj

Martin Zachariasen, DIKU

18. maj 2009

Consider the problem of neatly printing a paragraph on a printer. We are given n words of length l_1, l_2, \dots, l_n , measured in characters. The paragraph should be printed neatly on a number of lines that hold a maximum of M characters each. Our criterion of “neatness” is as follows. If a given line contains words i to j , where $i \leq j$, and we leave exactly one space between words, the number of extra spaces s at the end of the line is

$$s = M - (j - i) - \sum_{k=i}^j l_k$$

If $s = 0$, then there are no spaces at the end of the line, but if $s > 0$, we would like to “penalize” these extra spaces. This is done by applying a cost function $c(i, j)$, which measures the cost of having a line that contains words i to j , where $i \leq j$. We assume that $c(i, j)$ can be computed in $O(j - i + 1)$ time, and that $c(i, j) = \infty$ if words i to j do not fit on a single line. A typical choice for function $c(i, j)$ — where words i to j do fit on a single line — is

$$c(i, j) = \left(M - (j - i) - \sum_{k=i}^j l_k \right)^3$$

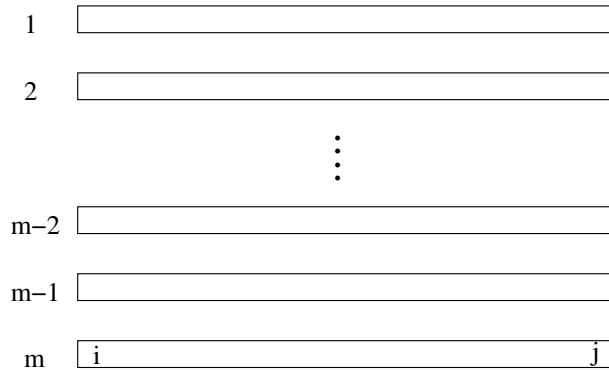
where extra spaces are penalized quite heavily. (One extra space costs 1, two extra spaces cost 8 etc.). However, our printing algorithm should work for any choice of $c(i, j)$, so let us assume that $c(i, j)$ is unknown for now.

Our goal is to print the n words in such a way that the total cost over all lines printed, *except* the last one, is *minimized*. The problem can be solved by dynamic programming; in exercises 1–4 we follow the four steps suggested in CLRS chapter 15.

*Baseret på opgave 15-2 i CLRS.

Exercise 1

Characterize the structure of an optimal solution. What is the optimal substructure for the problem? Hint: Assume that words i to j , $i \leq j$, are printed on line m in an optimal solution to the problem:



What can you say about the total cost of the lines $1, \dots, m - 1$?

Exercise 2

Recursively define the value of an optimal solution. Write down a recursive formula for the value of an optimal solution. Hint: Express the value using a function $f[j]$, which is the minimum cost of an optimal solution for the first j words with the assumption that word j is printed last on a line.

Exercise 3

Compute the value of an optimal solution in a bottom-up fashion. Describe your dynamic programming algorithm in detail. What is the running time of the algorithm? What is the running time if we assume that M is a constant?

Exercise 4

Construct an optimal solution from computed information. What information is needed to construct an optimal solution (and not only its value)?

Exercise 5

Run your algorithm on the example where $M = 10$, $n = 8$ and $l_1 = 4$, $l_2 = 5$, $l_3 = 7$, $l_4 = 2$, $l_5 = 4$, $l_6 = 3$, $l_7 = 8$ and $l_8 = 2$.