Algoritmer & Datastrukturer
Aflevering 5

Anders Bjerg Pedersen, Hold 2

15. marts 2007

Excercise 1

Antag som i opgavetekstens hint, at ordene i til j, $i \leq j$ er printet i linien m i en optimal løsning til vores problem. Der må så gælde, at ordene $1, \ldots, i-1$ også må stå i optimal orden i vores løsning. Hvorfor? Argumentet er det sædvanlige "copy-paste-argument: vores optimale løsning må have placeret ordene $1, \ldots, i-1$ på den bedste måde, så $c[1, i-1]$ er minimeret. Fandtes der en bedre måde, kunne vi bruge denne i stedet, men så ville vores løsning jo ikke være optimal. Samme argument gælder for ordene $j+1, \ldots, n$, der ligeledes må være placeret optimalt, hvis ordene fra i til j i linie m er det. Altså har vores problem egenskaben med de optimale substrukturer, og derfor kan den optimale løsning til et problem findes ud fra løsningerne til de mindre delproblemer.

Den totale omkostning ved de første $m-1$ ($c[1, m-1]$) linier må altså være minimal i vores optimale løsning og i alle tilfælde mindre end eller lig den totale omkostning ved de første m linier ($c[1, m]$). Omkostningsfunktionen er altså monotont voksende efter antal linier.

Excercise 2

Hvis alle ordene passer på én linie, er algoritmen selvfølgelig triviel. Hvis ikke, er vores ide at forsøge os med at fylde den første linie, så dennes omkostning minimeres, og herefter så at fylde resten af ordene ind i linier på en optimal måde. Vi indfører her det n-dimensionale array $f[i]$, der er defineret til at indeholde omkostningerne ved at placere ordene $i, \ldots, n$ (bemærk: ikke helt samme fremgangsmåde som i hintet!).

Vi definerer en ny funktion, $\textsc{Max}(i)$ til at være det største j, for hvilket $j - i + \sum_{k=i}^{j} l_k < M$, altså det største j, for hvilket ordene fra i til j kan være på en linie af længde M (eller igen igen med andre ord: $\textsc{Max}(i)$ er det sidste ord, der er plads til på den linie, som indledes af ordet i). Her gælder der selvfølgelig, at hvis $\textsc{Max}(i) = n$, er $f[i] = 0$ (så kan alle de resterende ord være på én linie). Den rekursive definition af $f[i]$ kommer da til at se ud som følger:

$$f[i] = \begin{cases} 0, & \textsc{Max}(i) = n \\ \min_{j=i,\ldots,\textsc{Max}(i)} \left\{ M - j + i - \sum_{k=i}^{j} l_k + f[j+1] \right\}, & 1 \leq \textsc{Max}(i) < n \end{cases}$$

I dette udtryk kan man så erstatte de første 4 led med ens cost-funktion (f.eks. sætte et "i tredje"rundt om dem som foreslået i opgaven).

Excercise 3

Vores "bottom-up"metode bliver lidt omvendt, idet vi starter bagfra. De minimale omkostninger for at placere alle ord er da givet som $f[1]$. Ud fra det i'te ord skal vi finde de mindste omkostninger ved enten kun at placere det i'te ord i linien ($\textsc{Max}(i) = i = j$) eller placere op til og med det j'te ord ($\textsc{Max}(i) = j$) og så tillægge omkostningerne ved at placere resten af ordene ($f[j+1]$). Vi bemærker, at der i hver linie højest kan forekomme $M/2$ ord (i hvilket tilfælde ordene i linien alle vil være netop én karakter lange med ét mellemrum mellem hvert ord), og dermed kan beregningerne i den rekursive algoritme udføres i tiden $O((M/2)n) = O(Mn)$. Hvis M var en konstant, ville den udgå af køretidsberegningen, og algoritmen ville køre i $O(n)$ tid.

Excercise 4

Hmm...ja det er jo nok noget snedigt noget med at bruge sit f-array og ordlængderne til noget smart, men det må lige blive en anden gang...

Excercise 5

Vi får følgende værdier i vores f-array:

$$f = [4, 9, 4, 12, 4, 9, 2, 0]$$

Vi er her startet bagfra med at beregne $f[8] = 0$, da $\textsc{Max}(i) = 8 = n$. Som et eksempel kan vi se på beregningen af $f[5]$, hvor vi får følgende udtryk, da $\textsc{Max}(5) = 6$:

$$f[5] = \min_{j=5,6} \left\{ 10 - j + 5 - \sum_{k=5}^{j} l_k + f[j+1] \right\} = \min\{15, 4\} = 4.$$

Det færdige resultat kommer så til at se nogenlunde således ud:

1	1	1	1	■	2	2	2	2	2
3	3	3	3	3	3	3	■	4	4
5	5	5	5	■	6	6	6	■	■
7	7	7	7	7	7	7	7	■	■
8	8	■	■	■	■	■	■	■	■