

# Data Structures and Algorithms

## Solutions 9

Markus Innerebner, Romans Kasperovics  
dsa(a)inf.unibz.it

Free University of Bozen - Bolzano, Italy

Thu, April 30, 2009

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#define INFTY 1000000
#define MAX_WORDS 1000

void wwrap(char txt[], int m);

int main() {
    char text[] = "Alice_was_beginning_to_get_very_tired_of_sitting_by_"
        "her_sister_on_the_bank,_and_of_having_nothing_to_do.";
    wwrap(text,30);
    return 0;
}
```

# Solutions 07

## Task 1: Word-wrap function

```
/* assumes words in txt are separated with a single white space and txt is a  
proper null-terminated string */  
void wwrap(char txt[], int m) {  
    int words[MAX_WORDS]; // words[i] points to the last character of word i in txt  
    int F[MAX_WORDS]; // costs of wrapping j words  
    int P[MAX_WORDS]; // choices (last line break) for j words  
    int i,j;  
    // reading words and counting their lengths  
    int tlen = strlen(txt);  
    int wcount = 0; // word count in txt  
    for (i=0; i<=tlen; i++)  
        if (txt[i] == ' ' || txt[i] == '\0')  
            words[wcount++] = i-1;  
    // calculating f(j) in a bottom-up fashion  
    ...  
    // printing the paragraph  
    pprint (wcount-1,m,txt,words,wcount,P);  
}
```

# Solutions 07

## Task 1: Calculating $f(j)$ in a bottom-up fashion

```
// calculating  $f(j)$  in a bottom-up fashion
for ( $j = 0; j < wcount; j++$ ) { // calculating  $f[j]$  from  $0..wcount-1$ 
     $P[j] = -1$ ; // initial value – no breaks at all
     $F[j] = cost(0, j, m, words, wcount)$ ; // cost of no breaks at all
    if ( $F[j] \geq INFTY$ ) // if words  $0..j$  do not fit into one line
        for ( $i = 0; i < j; i++$ ) { // searching for minimum cost last break
            int  $t = F[i] + cost(i+1, j, m, words, wcount)$ ;
            if ( $t < F[j]$ ) {
                 $F[j] = t$ ;
                 $P[j] = i$ ;
            }
        }
    }
}
```

# Solutions 07

## Task 1: Results so far.

Alice<sup>0</sup> was<sup>1</sup> beginning<sup>2</sup> to<sup>3</sup> get<sup>4</sup> very<sup>5</sup> tired<sup>6</sup> of<sup>7</sup> sitting<sup>8</sup> by<sup>9</sup> her<sup>10</sup> sister<sup>11</sup>  
on<sup>12</sup> the<sup>13</sup> bank,<sup>14</sup> and<sup>15</sup> of<sup>16</sup> having<sup>17</sup> nothing<sup>18</sup> to<sup>19</sup> do.<sup>20</sup>

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
words[]	4	8	18	21	25	30	36	39	47	50	54	61	64	68	74	78	81	88	96	99	103
F[]	625	441	121	64	16	482	290	221	89	52	20	302	257	189	101	61	36	306	182	137	97
P[]	-1	-1	-1	-1	-1	2	2	2	3	4	4	7	7	8	9	9	10	12	14	14	15

Alice was beginning to get  
very tired of sitting by  
her sister on the bank, and  
of having nothing to do.

# Solutions 07

## Task 1: Printing the paragraph

```
void pprint (int j, int m, char txt[], int words[], int wcount, int breaks[]) {  
    if (j<0) return;  
    // printing words from 0 to breaks[j]  
    pprint(breaks[j],m,txt,words,wcount,breaks);  
    // printing words from breaks[j]+1 to j  
    int i = breaks[j]+1;  
    int ecount = extra(i,j,m,words,wcount);  
    int k = (i==0) ? 0 : words[i-1] + 2;  
    for (; k<=words[j]; k++) {  
        if (txt[k] == ' ') {  
            int es = ceil((double) ecount / (j-i));  
            printf("%*c", es + 1, ' ');  
            ecount -= es;  
            i++;  
        } else printf("%c",txt[k]);  
    }  
    printf("\n");  
}
```

# Solutions 07

## Task 1: Functions 'cost' and 'extra'

```
int extra(int i, int j, int m, int words[], int wcount) {  
    return m - ((i == 0) ? words[j]+1 : words[j]-words[i-1]-1);  
}
```

```
int cost(int i, int j, int m, int words[], int wcount) {  
    int ecount = extra(i,j,m,words,wcount);  
    if (ecount < 0) return INFTY;  
    else return ecount*ecount;  
}
```

# Solutions 07

## Task 2: Function 'extra2'

```
int extra2(int i, int j, int m, int words[], int wcount) {  
    int ecount = extra(i,j,m,words,wcount);  
    if (j == wcount-1 && (double)ecount / m > 0.1) return 0;  
    else return ecount;  
}
```

# Solutions 07

## Task 2: Results

Alice<sup>0</sup> was<sup>1</sup> beginning<sup>2</sup> to<sup>3</sup> get<sup>4</sup> very<sup>5</sup> tired<sup>6</sup> of<sup>7</sup> sitting<sup>8</sup> by<sup>9</sup> her<sup>10</sup> sister<sup>11</sup>  
on<sup>12</sup> the<sup>13</sup> bank,<sup>14</sup> and<sup>15</sup> of<sup>16</sup> having<sup>17</sup> nothing<sup>18</sup> to<sup>19</sup> do.<sup>20</sup>

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
words[]	4	8	18	21	25	30	36	39	47	50	54	61	64	68	74	78	81	88	96	99	103
F[]	625	441	121	64	16	482	290	221	89	52	20	302	257	189	101	61	36	306	182	137	36
P[]	-1	-1	-1	-1	-1	2	2	2	3	4	4	7	7	8	9	9	10	12	14	14	16

Alice was beginning to get  
very tired of sitting by her  
sister on the bank, and of  
having nothing to do.

$A_1$  is a  $4 \times 3$  matrix,  $A_2$  is a  $3 \times 6$  matrix,  $A_3$  is a  $6 \times 2$  matrix.

$((A_1 \times A_2) \times A_3)$  contains  $4 \cdot 3 \cdot 6 + 4 \cdot 6 \cdot 2 = 120$  multiplications.

$(A_1 \times (A_2 \times A_3))$  contains  $3 \cdot 6 \cdot 2 + 4 \cdot 3 \cdot 2 = 60$  multiplications.

# Solutions 07

## Task 3

```
/* assumes that M[][] is initialized with INFTY */
int MemoMM(int i, int j, int D[], int M[][N], int P[][N]) {
    if (i == j) return 0;
    else if (M[i][j] < INFTY) return M[i][j]; // element already calculated
    else {
        int k, q;
        for (k=i; k<j; k++) {
            q = MemoMM(i,k,D,M,P) + MemoMM(k+1,j,D,M,P) + D[i]*D[k+1]*D[j+1];
            if (q < M[i][j]) {
                M[i][j] = q;
                P[i][j] = k; // needed for printing (task 4)
            }
        }
        return M[i][j];
    }
}
```

```
void printParentheses(int i, int j, int P[][N]) {  
    if (i == j) printf("A%i",i+1);  
    else {  
        printf("(");  
        printParentheses(i,P[i][j],P);  
        printf("_x_");  
        printParentheses(P[i][j]+1,j,P);  
        printf(")");  
    }  
}
```

# Solutions 07

## Task 4: Main function

```
#include <stdio.h>
#define N 3 // number of matrices
#define INFTY 1000000

int main() {
    int D[N+1] = {4,3,6,2}; // dimensions of 3 matrices
    int M[N][N]; // auxiliary memory structure for costs
    int P[N][N]; // auxiliary memory structure for decisions
    int i,j;
    for (i=0; i<N; i++)
        for (j=0; j<N; j++)
            M[i][j] = INFTY;
    MemoMM(0,N-1,D,M,P);
    printParentheses(0,N-1,P);
    printf("\n");
}
```