Metoda programării dinamice (II)

Obiective:
1. Exersarea metodei programării dinamice prin implementarea a două probleme specifice:
2. Problema subşirului crescător maximal (LAS – Longest Ascending Subsequence)
3. Problema celui mai lung subşir comun (LCS – Longest Common Subsequence)

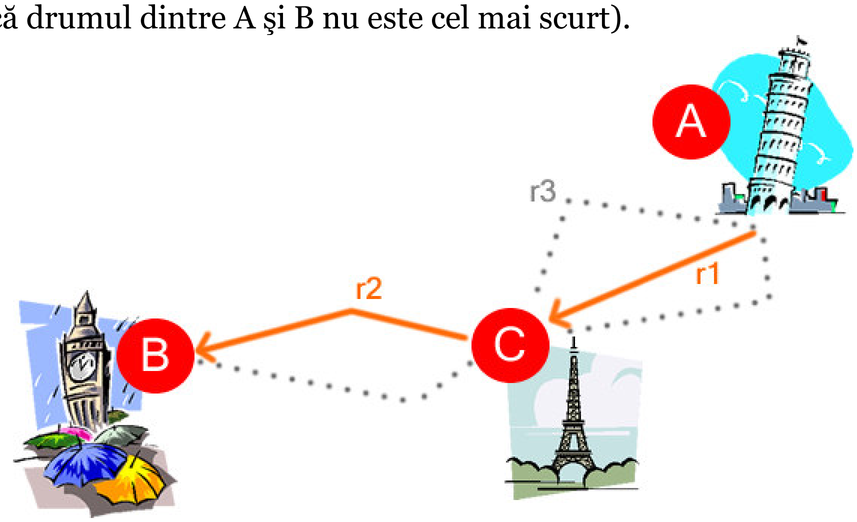
1. Metoda programării dinamice – Prezentare generală

Programarea dinamică este o tehnică ce se aplică pentru rezolvarea problemelor pentru care se cere o soluție optimă. Termenul de programare dinamică a fost introdus de către Richard Bellman[1,2] în anul 1953. Metoda conduce la obținerea unui **timp de calcul polinomial**.

Programarea dinamică are la bază **principiul optimalității**: problema poate fi descompusă în subprobleme asemănătoare de dimensiuni mai mici iar soluțiile (optime) ale acestor subprobleme contribuie la obținerea soluției optime pentru problema dată. Cu alte cuvinte, **optimul general implică optimul parțial**.

Exemplu

O ilustrare a acestui principiu este dată de următorul exemplu: dacă drumul cel mai scurt între două oraşe A şi B trece prin oraşul intermediar C pe ruta r1 (A-C) r2 (C-B) atunci şi ruta r1 corespunde celui mai scurt drum care uneşte oraşele A şi C (dintre toate rutele posibile) precum şi porțiunea de drum r2 (C-B) corespunde celui mai scurt drum care uneşte oraşele C şi B. În caz contrar (presupunem ca ar exista o altă rută r3(A-C) mai scurtă atunci aceasta ar însemna că drumul dintre A şi B nu este cel mai scurt).



[1] http://en.wikipedia.org/wiki/Richard_Bellman
[2] http://en.wikipedia.org/wiki/Dynamic_programming

Rezolvarea unei probleme folosind această metodă presupune:

a) divizarea problemei în subprobleme de dimensiuni mai mici
b) rezolvarea tuturor subproblemelor de manieră optimă folosind etapele a, b, c
c) reunirea soluțiilor subproblemelor pentru a obține soluția optimă a problemei inițiale

2. Subşirul crescător maximal (LAS – Longest Ascending Subsequence)

Problemă

Fie un şir de n numere întregi. Să se găsească cel mai lung subşir crescător al acestuia. În cazul în care există mai multe subşiruri de lungime maximă, va fi afişat unul dintre ele la alegere.

Intrare	**Ieşire**
n = 11 7, 1, 5, 2, 8, 2, 4, 1, 8, 7, 10	6 1, 2, 2, 4, 8, 10

Cel mai lung subşir crescător are lungimea 6:
7, **1**, 5, **2**, 8, **2**, **4**, 1, **8**, 7, **10**
Obs.: soluția nu este unică, putem avea de exemplu: 7, **1**, 5, **2**, 8, **2**, **4**, 1, 8, **7**, **10**

Problema se rezolvă calculând pentru fiecare element i al vectorului de numere întregi (a) lungimea celui mai lung subşir crescător care poate fi format începând cu elementul i (pe care o reținem în vectorul lung[i]). În final, vom afişa valoarea maximă din vectorul lung, ceea ce corespunde lungimii celui mai mare subşir crescător.

Algoritmul pleacă de la următoarea idee: presupunem cunoscute valorile lung[j] pentru toți indecşii j după i, j=i+1 ... n-1. Vom construi cel mai lung subşir crescător care începe cu elementul a[i], alipindu-l celui mai lung subşir crescător format cu elemente aflate în şir după poziția i (j=i+1 ... n-1) cu condiția de a respecta proprietatea de subşir crescător a noului subşir format, şi anume: adăugăm a[i] ca primul element la subşirul care începe cu a[j] numai dacă a[i] <= a[j]. Vom alege acel j pentru care lung[j] este maxim. În acest caz, lung[i] va deveni lung[j] + 1. Dacă nu există nici un element a[j] mai mare decât a[i], atunci elementul lung[i] = 1 (a[i] este un terminator de subşir, neexistând nici un element mai mare decât el poziționat după indexul i).

Rezumând, vom avea:

$$lung[i] = \begin{cases} 1 & daca\ a[i] > a[j]\ \forall j = \overline{i+1, n-1} \\ lung[k]+1 & unde\ k = \arg\max_j \left\{ lung[j],\ j = \overline{i+1, n-1} \right\} \end{cases}$$

plecând de la condiția inițială lung[n-1] = 1 (cel mai lung subşir crescător pentru şirul format din ultimul element are lungimea 1). În final, lungimea celui mai lung subşir crescător va fi dată de $\max\left\{lung[i], i = \overline{0, n-1}\right\}$.

Exemplu

i		0	1	2	3	4	5	6	7	8	9	10
a		7	1	5	2	8	2	4	1	8	7	10
lung	i=10											1
	i=9										2	1
	i=8									2	2	1
	i=7								3	2	2	1
	i=6							3	3	2	2	1
	i=5						4	3	3	2	2	1
	i=4					3	4	3	3	2	2	1
	i=3				5	3	4	3	3	2	2	1
	i=2			4	5	3	4	3	3	2	2	1
	i=1		6	4	5	3	4	3	3	2	2	1
	i=0	4	**6**	4	5	3	4	3	3	2	2	1

Algoritmul de construcţie a şirului de lungimi parţiale lung, este prezentat în continuare în limbajul pseudocod.

Pseudo cod

```
lung[n-1] ← 1
pentru i = n-2, 0 execută
     max ← 0
     pentru j = i+1, n-1 execută
          dacă a[i] <= a[j] atunci
               dacă max < lung[j] atunci
                    max ← lung[j]
               sf.dacă
          sf.dacă
     sf.pentru
     lung[i] ← max + 1
sf.pentru
```

Pentru a identifica cel mai lung subşir, vom căuta maximul (şi poziţia acestuia) din şirul lung.

Pseudo cod

```
max ← lung[0]
poz ← 0
pentru i = 1, n-1 execută
     dacă max < lung[i] atunci
          max ← lung[i]
          poz ← i
     sf.dacă
sf.pentru

scrie „Cel mai lung subşir crescător are lungimea” + max
scrie „Iar subşirul este:”

scrie a[poz]
pentru i = poz+1, n-1 execută
```

```
    dacă lung[i] == max - 1 şi a[i] >= a[poz] atunci
         scrie a[i]
         poz ← i
         max ← max - 1
    sf.dacă
sf.pentru
```

Discuţie

Algoritmul are complexitatea $O(n^2)$ întrucât pentru construcţia şirului de lungimi este necesară parcurgerea dublă a şirului a (cu cei doi indecşi i mergând de la n-2 la 0 şi j mergând pentru fiecare i de la i+1 la n-1).

Pentru aflarea valorii maxime din şirul lung se realizează o căutare de complexitate $O(n)$. Deasemenea, subşirul este tipărit cu o complexitate de $O(n)$.

Complexitatea finală a soluţiei este de $O(n^2)$.

3. Cel mai lung subşir comun (LCS – Longest Common Subsequence)

Problemă

Fie două şiruri a şi b de n respectiv m numere întregi. Să se găsească cel mai lung subşir comun şirurilor a şi b.

Intrare	Ieşire
n = 11 7, 6, 5, 2, 8, 2, 8, 1, 4, 7, 10 m = 7 6, 9, 2, 1, 8, 9, 7	4 6, 2, 8, 7

Cel mai lung subşir comun are lungimea 4:
7, **6**, 5, **2**, **8**, 2, 8, 1, 4, **7**, 10
6, 9, **2**, 1, **8**, 9, **7**

Problema se rezolvă prin divizare în probleme de dimensiuni mai mici. Anume, vom nota cu lung[i,j] lungimea celui mai lung subşir comun şirurilor alcătuite din primele i elemente ale lui a şi primele j elemente ale lui b, i=0,n-1 şi j=0,m-1. Cu această notaţie suntem interesaţi evident de valoarea lung[n-1, m-1].

Problemele de dimensiuni mici pot fi rezolvate imediat:

$$lung[0,0] = \begin{cases} 1 & daca\ a[0] = b[0] \\ 0 & altfel \end{cases},$$

$$lung[0,j] = \begin{cases} 1 & daca\ a[0] = b[j] \\ lung[0,j-1] & altfel \end{cases} \quad \text{şi}$$

$$lung[i,0] = \begin{cases} 1 & daca\ a[i] = b[0] \\ lung[i-1,0] & altfel \end{cases}$$

Pentru a calcula lung[i, j] vom reduce problema la dimensiunile i-1 respectiv j-1 pentru care dispunem deja de lung[i-1, j], lung[i-1, j-1] şi lung[i, j-1]. Şi anume, pentru a ajunge la subşirul comun primelor i elemente din a şi primelor j elemente din b, avem 3 posibilităţi de înaintare:

1) la primele i-1 elemente din a mai adăugăm a[i] lăsând j neschimbat, ceea ce face ca lungimea celui mai lung subşir comun să fie lung[i-1, j]
2) la primele j-1 elementedin b mai adăugăm b[j] lăsând i neschimbat, ceea ce face ca lungimea celui mai lung subşir comun să fie lung[i, j-1]
3) la primele i-1 elemente din a mai adăugăm a[i] iar la primele j-1 elementedin b mai adăugăm b[j], caz în care:
   a. dacă a[i] = b[j] atunci lungimea celui mai lung subşir creşte cu 1 faţă de lung[i-1, j-1]
   b. în caz contrar, lungimea celui mai lung subşir este tot lung[i-1, j-1] (adăugarea la fiecare şir a câte un element a[i] respectiv b[j] diferite nu schimbă lungimea totală a celui mai lung subşir comun)

Exemplu

a/b	7	6	5	2	8	2	8	1	4	7	10
6	0	1	1	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1	1
2	0	1	[illegible]	[illegible]	2	2	2	2	2	2	2
1	0	1	1	2	2	2	2	3	3	3	3
8	0	1	1	2	3	3	3	3	3	3	3
9	0	1	1	2	3	3	3	3	3	3	3
7	1	1	1	2	3	3	3	3	3	4	4

Afişarea subşirului comun se realizează parcurgând matricea în sens invers, de la poziţia n-1, m-1 la 0,0:

a/b	7	6	5	2	8	2	8	1	4	7	10
6	**0**	**1**	**1**	**1**	**1**	1	1	1	1	1	1
9	0	1	1	1	**1**	1	1	1	1	1	1
2	0	1	1	2	2	**2**	**2**	2	2	2	2
1	0	1	1	2	2	2	2	**3**	**3**	[illegible]	[illegible]
8	0	1	1	2	3	3	3	3	**3**	3	3
9	0	1	1	2	3	3	3	3	**3**	3	3
7	1	1	1	2	3	3	3	3	3	**4**	**4**

Algoritmul de construcţie a şirului de lungimi parţiale lung, este prezentat în continuare.

Pseudo cod

```
lung[0,0] ← (a[0] == b[0]) ? 1 : 0

pentru i = 1, n - 1 execută
     lung[i, 0] ← (a[i] == b[0]) ? 1 : lung[i-1, 0]
sf.pentru
```

```
pentru j = 1, m - 1 execută
     lung[0, j] ← (a[0] == b[j]) ? 1 : lung[0, j-1]
sf.pentru

pentru i = 1, n - 1 execută
     pentru j = 1, m - 1 execută
          lung[i, j] = max{
                  (a[i] == b[j]) ? lung[i-1, j-1] + 1 : 0,
                   lung[i-1, j],
                   lung[i, j-1]}
     sf.pentru
sf.pentru
```

Lungimea celui mai lung subşir comun celor două şiruri de numere a şi b de dimensiuni n şi m se găseşte la poziţia lung[n-1, m-1]. Afişarea subşirului constă în parcurgerea matricei în sens invers, plecând de la n-1, m-1.

Pseudo cod

```
scrie „Subşirul comun maxim are lungimea" + lung[n-1,m-1]
scrie „Iar subşirul este:"

i ← n-1
j ← m-1
cât timp i>=0 şi j>=0 execută
     dacă a[i] == b[j] atunci
          scrie a[i]
          i ← i-1
          j ← j-1
     altfel
     dacă i-1>=0 şi lung[i, j] == lung[i-1, j] atunci
          i ← i-1
     altfel
          j ← j-1
     sf.dacă
sf.cât timp
```

Discuţie

Algoritmul are complexitatea $O(n*m)$ întrucât pentru construcţia matricei de lungimi este necesară parcurgerea celor două şirului a şi b.

Pentru aflarea subşirului comun este necesară parcurgerea matricei de lungimi parţiale cu o complexitate de $O(m+n)$ întrucât sunt necesare n+m operaţii de decrementare a lui i sau j pentru a ajunge de la poziţia n-1,m-1 la 0,0.

Complexitatea finală a soluţiei este de $O(n*m)$.

4. Tema de laborator

Implementaţi problemele LAS (cel mai lung subşir crescător dintr-un şir) şi LCS (cel mai lung subşir comun pentru două şiruri de numere întregi).

Pentru problema LAS, formatul fişierului de intrare este următorul:

1) pe prima linie este dat numărul n de elemente din şir
2) pe a doua linie sunt date elementele şirului separate printr-un spaţiu.

Veţi afişa în fişierul de ieşire lungimea celui mai lung subşir crescător iar pe linia următoare elementele subşirului, separate prin spaţiu.

Intrare (in.txt)	Ieşire (out.txt)
11 7 1 5 2 8 2 4 1 8 7 10	6 1 2 2 4 8 10

Pentru problema LCS, formatul fişierului de intrare este următorul:

1) pe prima linie este dat numărul n de elemente din primul şir, a
2) pe a doua linie sunt date elementele şirului a separate printr-un spaţiu
3) a 3-a linie conţine numărul m de elemente din cel de-al doilea şir, b
4) pe a 4-a linie sunt date elementele şirului b separate printr-un spaţiu

Veţi afişa în fişierul de ieşire lungimea celui mai lung subşir comun iar pe linia următoare elementele subşirului, separate prin spaţiu.

Intrare	Ieşire
11 7 6 5 2 8 2 8 1 4 7 10 7 6 9 2 1 8 9 7	4 6 2 8 7

5. Probleme propuse

Cel mai lung subşir descrescător impar

Problema #1

Fie un şir de n numere întregi. Să se găsească cel mai lung subşir descrescător al acestuia alcătuit numai din elemente impare. În cazul în care există mai multe subşiruri de lungime maximă, va fi afişat unul dintre ele la alegere.

Intrare	Ieşire
n = 11 7, 1, 5, 2, 8, 2, 4, 1, 8, 7, 10	3 7, 5, 1

Cel mai lung subşir descrescător impar are lungimea 3:
7, 1, **5**, 2, 8, 2, 4, **1**, 8, 7, 10

Jocul fazan

Problema #2

Fie un text alcătuit dintr-un număr de n cuvinte. Să se găsească cea mai lungă secvenţă de cuvinte cu proprietatea că fiecare cuvânt din secvenţă începe cu ultimile 2 litere ale cuvântului anterior.

Intrare	Ieşire
n = 6 astazi mare zilele repede cuier leopard	3 astazi zilele leopard

Cea mai lungă subsecvenţă este:
astazi mare **zilele** repede cuier **leopard**

Verificare vocabular

Problema #3

Fie un dicţionar alcătuit dintr-un număr de n cuvinte şi fie m cuvinte introduse (posibil eronat) de către un utilizator într-un editor text. Realizaţi operaţie de verificare a vocabularului, corectând cuvintele introduse de utilizator pe baza celor din dicţionar.

Intrare	Ieşire
n = 5 mare univers banca universitate stilou m = 3 univesr banca bnca	univers banca banca

Common Subsequence – F (ACM 2003)

Problema #4

Problema F dată la concursul studenţesc internaţional de programare ACM SouthEastern Europe Region, Bucureşti 2003, textul disponibil la adresa:

http://www.acm.ro/2003/probleme/F.pdf

Longest Palindrom

Problema #5

Fie un şir de n numere întregi. Să se găsească cel mai lung subşir al acestuia care să fie palindrom (un subşir palindrom afişat în ordine inversă este identic cu subşirul original). În cazul în care există mai multe subşiruri de lungime maximă, va fi afişat unul dintre ele la alegere.

Intrare	Ieşire
n = 9 6, 1, 5, 3, 7, 8, 3, 1, 9	5 1, 3, 7, 3, 1

Cel mai lung subşir palindrom are lungimea 5:
6, **1**, 5, **3**, **7**, 8, **3**, **1**, 9