Fundamentele programarii

Curs 10

Tehnica - Programare Dinamica

www.cs.ubbcluj.ro/~avescan/fp-2009

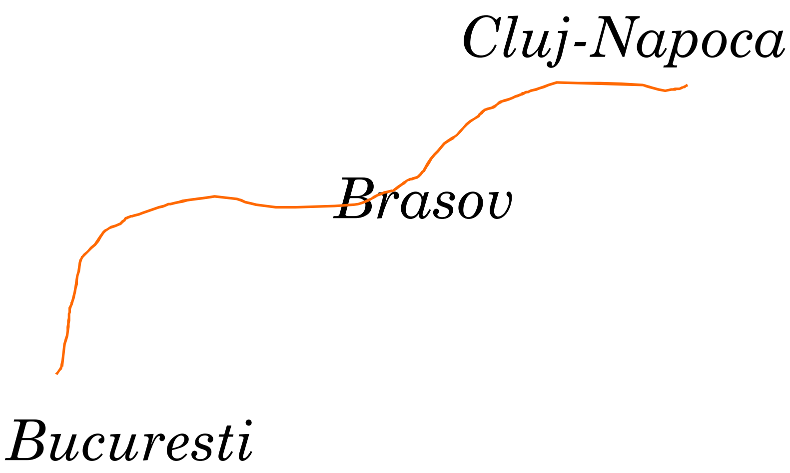
9.3 Tehnica programare dinamica

- furnizeaza intotdeauna solutia optima;
- poate fi aplicata doar unor clase de probleme (care indeplinesc anumite conditii);
- Metoda de proiectare – bottom-up
  - Se rezolva intai subcazurile care pot fi solutionate imediat;
  - Se combina aceste solutii, obtinand astfel solutii pentru cazurile mai mari decat cea initiala, pana cand se obtine solutia problemei initiale.
- timp de executie polinomial;
- Greedy – probleme in care **optimul general se obtine din optime locale (partiale)**;
- PD – probleme in care **optimul general implica optimele partiale.**

9.3 TEHNICA PROGRAMARE DINAMICA

Principiul optimalitatii

Optimul general implica optimele partiale.



Bucuresti – Brasov- Cluj-Napoca

- Drumul cel mai scurt

➔

- ➔ Bucuresti – Brasov
- ➔ Brasov-Cluj-Napoca
- drumul cel mai scurt

Observatie: Optimul partial nu implica optimul general.

- Drumul dBC de la Bucuresti la Craiova – cel mai scurt;
- Drumul dCC de la Craiova la Cluj-Napoca – cel mai scurt;
- NU → dBC+dCC -
  - NU este cel mai scurt de la Bucuresti la Cluj-Napoca

9.3.1 FORMALIZARE

- Solutia problemei este rezultatul unui sir de decizii D_1, D_2, …, D_n si se verifica principiul optimalitatii.
- S_0 --D_1-- S_1 --D_2-- … --D_n -- S_n
- Daca D_1, D_2, …, D_n este un sir de decizii care conduce sistemul in mod optim din S_0 in S_n, atunci una din urmatoarele conditii trebuie indeplinita:
  - D_k, …, D_n - sir de decizii ce conduce optim sistemul din starea S_{k-1} in starea S_n, $\forall$ 1<=k<=n.
    - Metoda se aplica *Inainte* D_k depinde de D_{k+1}, …, D_n.
  - D_1, …, D_k - sir de decizii ce conduce optim sistemul din starea S_0 in starea S_k, $\forall$ 1<=k<=n.
    - Metoda se aplica *Inapoi* D_k depinde de D_1, …, D_{k-1}.
  - D_{k+1}, …, D_n, D_1, …, D_k - sir de decizii ce conduce optim sistemul din starea S_{k-1} in starea S_n si din starea S_0 in starea S_k, $\forall$1<=k<=n.
    - Metoda *mixta*.

9.3.2 PROGRAMARE DINAMICA - ETAPE

- Etapele unui algoritm care utilizeaza PD sunt:
  - (1) verificare principiului optimalitatii;
    - Inainte;
    - Inapoi;
    - Mixt.
  - (2) scrierea unor relatii de recurenta care exprima mpodul de obtinere al optimului general din optime partiale;
  - (3) valoarea solutiei optime se determina in maniera bottom-up, plecand de la cazurile minimale pentru care valoarea solutiei se cunoaste.

9.3.3 PD – METODA INAINTE

- **Problema.** Se da un sir **s** de **ns** elemente. Se cere sa se tipareasca cel mai lung subsir crescator al acestuia. Ordinea de selectie a elementelor.
- Exemplu:
  - ns=5

	1	2	3	4	5
s	2	1	9	6	12

- Pentru fiecare s_i se calculeaza lungimea celui mai lung subsir crescator care se poate forma cu el;
- La final, se selecteaza elementul cu care se poate forma cel mai lung subsir crescator.

	1	2	3	4	5
s	**2**	1	**9**	6	**12**
L	3	3	2	2	1

9.3.3 PD – METODA INAINTE

- (1) verificare principiului optimalitatii
  - Pp. s_{i1}, s_{i2}, …, s_{ik},…s_{ip} – cel mai lung subsir cu elementul s_{ik}
  - ?➔ s_{ik},…s_{ip} – cel mai lung subsir incepand cu s_{ik} ?
    - prin reducere la absurd pp ca nu este;
    - exista alt subsir de lungime mai mare incapand cu s_{ik}.
      - Contradictie cu presupunerea initiala (s_{i1}, s_{i2}, …, s_{ik},…s_{ip} - cel mai lung).
  - Metoda inainte - D_k depinde de D_{k+1}, …, D_p.
- (2) relatiile de recurenta (L_k-lungimea celui mai lug subsir care incepe cu elementul de la pozitia k)
  - $L_{ns}=1$
  - $L_k=\max\{1+L_i \mid s_i>s_k, i=k+1,ns, k=ns-1,1\}$

9.3.3 PD – METODA INAINTE

- (3) obtinerea solutiei optime
  - Se calculeaza maximul din L; cel mai lung subsir crescator va avea lungimea egala cu acest maxim.
  - Afisarea valorilor subsirului
    - Se cauta max din L (fie pozitia p) si se afiseaza s[p].
    - Se cauta si se afiseaza primul element >=s[p] si care are lungimea mai mica cu 1 decat lungimea maxima max; se actualizeaza max←max-1;
    - Se parcurg toate elementele sirului.

	1	2	3	4	5
s	**2**	1	**9**	6	**12**
L	3	3	2	2	1

- max=3, p=1
  - afis s[1]=2, max=2;
  - s[3]=9>s[p=1]=2 si L[3]=2=max→afis s[3]=9; max=1;
  - s[5]=12>s[p=3]=9 si L[5]=1=max→afis s[5]=12; max=1;

9.3.4 PD – METODA INAPOI

- **Problema.** Se da un sir **s** de **ns** elemente. Se se determine numerele care formeaza cea mai mare suma divizibila cu ns (fiecare numar participa o singura data la suma).
- ns=5;

	1	2	3	4	5
s	2	3	4	9	3

- La primul pas se considera primul numar ca o suma de 1 element;
- La fiecare pas se determina sumele maxime care impartite la **ns** dau restul {0,1,…,ns}➔sm_{i+1} contine suma care prin impartirea lui **ns** are rest 0.

9.3.4 PD – METODA INAPOI

	1	2	3	4	5
s	2	3	4	9	3

Rest	0	1	2	3	4
sm	15	21	17	18	19
Numerele care compun suma sm =nrSm	2 3	2	2	2	3
	4 9	3	3	4	4
	9 3	4	9	9	9
		9	3	3	3
		3			

9.3.4 PD – METODA INAPOI

- (1) verificare principiului optimalitatii
  - Prin inductie matematica
    - Pentru primul numar – evident –exista o singura suma;
    - Fie sm_1, sm_2,…sm_i – primele i sume generate, sume presupuse optime.
    - Modul de constructie pentru sm_{i+1} demonstreaza ca sm_{i+1} va contine sume optime calculate cu primele i+1 numere din s.
  - ➔constructia sm_{k+1} depinde de sm_1, sm_2,…sm_{k-1}.
  - PD - metoda inapoi
- (2) relatiile de recurenta
  - $sm_1 = s_1$. daca s_1 mod ns =k

    = 0, altfel
  - $sm_i(k) = s_i$. daca s_i mod ns =k

    $= sm_{i-1}(k)$

    $= sm_{i-1}(p) + s_i$, daca $((sm_{i-1}(p) + s_i)$ mod sn)=k

    pt. i=2,ns si pt. k=0,n-1

9.3.4 PD – METODA INAPOI

- (3) obtinerea solutiei optime
  - Suma maxima divizibila cu ns = cea care are restul 0;
  - Valorile numerelor care formeaza suma sunt numerele din **nrSm.**

9.3.5 PD – METODA MIXTA

- **Problema.** Inmultirea optima a unui sir de matrice.
- Exemplu A(n,p)* B(p,m)➔C(n,m) – n*m inmultiri
- $A_1(10,1)*A_2(1,10)*A_3(10,1)*A_4(1,10)$
  - $B(10,10)*A_3(10,1)*A_4(1,10)$
  - $C(10,1) *A_4(1,10)$
  - D(10,10)
  - 100*+100*+100*=300*
- $A_1(10,1)*A_2(1,10)*A_3(10,1)*A_4(1,10)$
  - $A_1(10,1)*B(1,1)* A_4(1,10)$
  - $A_1(10,1)*C(1,10)$
  - D(10,10)
  - 10*+10*+100*=120*

9.3.5 PD – METODA MIXTA

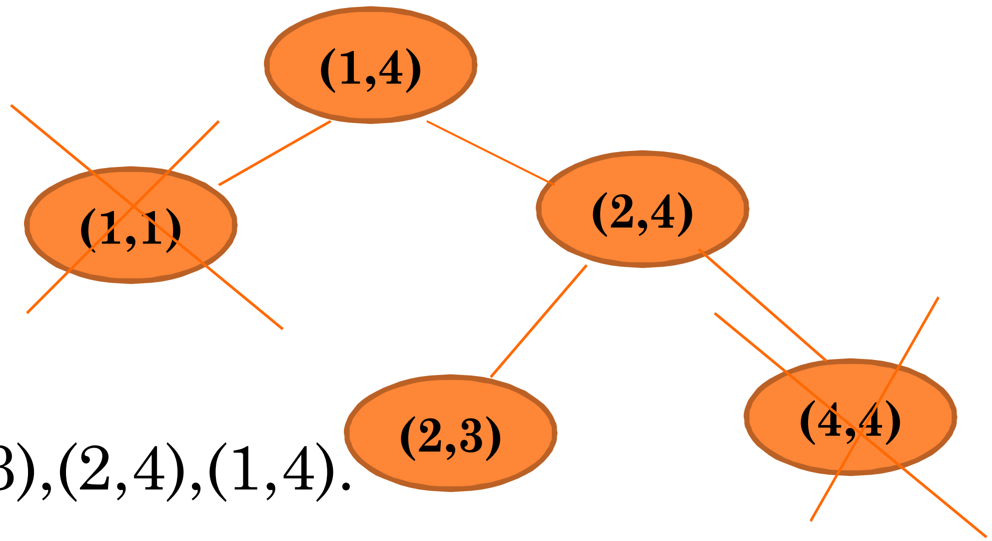
- (1) verificare principiului optimalitatii
  - Pp. s-a gasit o modalitate optima de efectuare a produsului- la final se inmultesc 2 matrice;
    - Prima: $A_1xA_2X\ldots xA_i$
    - A doua: $A_{i+1}xA_2X\ldots xA_n$
  - Cele doua produse au fost calculare cu numar minim de inmultiri ?
  - pp. prin reducere la absurd ca NU➔atunci exista o alta inmultire optima finala (prin imbunatatirea produsului initial dintre cele 2 matrice) = contradictie cu ipoteza (presupunerea initiala).
  - PD - metoda mixata: D_i depinde de deciziile $D_1, \ldots, D_{i-1}$ si de deciziile $D_{i+1}, \ldots, D_n$
- (2) relatiile de recurenta (DIM- vector cu dimensiunile matricelor)
  - a) A(i,i)=0
  - b) A(i,i+1)=DIM(i)*DIM(i+1)*DIM(i+2)
  - c) A(i,j)=min{A(i,k)+A(k+1,j)+DIM(i)*DIM(k+1)*DIM(j+1)}
  - i<=k<j

9.3.5 PD – METODA MIXTA

- (3) obtinerea solutiei optime
  - Pentru A(i,j) se retine in A(j,i) valoarea lui k pentru care se realizeaza minimul
  - Se creeaza un arbore binar astfel:
    - Radacina= (1,n);
    - Varful eticheta cu (i,j),i<j ➔ descendenti (i,k), (k+1,j)
  - Parcurgerea in postordine (SDR) – modul de calcula al produsului de matrice;
    - (i,j) ←➔paranteza inainte de A_i, parantza dupa $A_{j.}$

9.3.5 PD – METODA MIXTA

- Exemplu: $A_1(10,1)*A_2(1,10)*A_3(10,1)*A_4(1,10)$.
- Matricea A= 0 100 20 120
- 1 0 10 20
- 1 2 0 100
- 1 3 3 0
- Arborele



- SDR: (2,3),(2,4),(1,4).