bcast - soluţie

Problema cere determinarea unui arbore de broadcast în care momentul de timp maxim la care un calculator primeşte informaţiile este minim. Vom calcula valorile **Tmin[i,S]**, ca fiind timpul minim al unui arbore de broadcast ce are ca nod rădăcină nodul i şi din arbore fac parte toate nodurile din submulţimea S (S este reprezentat ca un număr pe N biţi, în care biţii de 1 reprezintă nodurile ce fac parte din submulţime ; nodul i face neapărat parte din submulţime). Pentru calculul lui Tmin[i,S] avem următoarele relaţii:

- `Tmin[i,submulţimea formată doar din nodul i]=0`
- Tmin[i,S] = minim dupa ((S_1 inclus in S) si (j nod din S_1)) din {Durata_transmisie(i,j)+maxim{Tmin[j,S_1],Tmin[i,S-S_1]}}

Răspunsul îl vom avea în Tmin[1,submulţimea formată din toate nodurile]. Complexitatea algoritmului este $O(3^N * N^2)$.

emax - soluţie

1. Solutia $O(N^3)$

Se pastreaza doua matrice `MIN[i][j]` cu `MAX[i][j]` cu valoarea minima, respectiv maxima pentru o expresie care foloseste numerele A_i, A_{i+1} … A_{j-1}, A_j din sir. Este nevoie de numere mari pentru a calcula aceste doua matrice.

2. Solutia $O(N^2)$

Fie $V_i=|A_i|$. Se observa ca adancimea parantezarii nu ajunge niciodata la nivelul 2, astfel o solutie optima este mereu de forma $(V_1+V_2+...+V_{i1})*(V_{i1+1}+...+V_{i2})*...*(V_{ik+1}...+V_N)$ $(0<i1<i2...<ik<N)$.
Astfel, o solutie de complexitate patratica se poate obtine calculcand `MAX[i]=` valoarea maxima pentru o expresie care foloseste primii `i` termeni din sir.
$MAX[i] = max(MAX[j]*(S_i-S_j))$, $j < i$, unde $S_i=V_1+V_2+...+V_i$.
Pentru a evita lucrul cu numere mari vom considera ca `MAX[i]` reprezinta logaritmul valorii maxime care se obtine cu primii `i` termeni. Relatia de recurenta devine:
$MAX[i] = max(MAX[j] + lg(S_i-S_j))$, $j < i$
Pentru a calcula rezultatul final modulo `666013` se pastreaza si un vector de predecesori.

3. Solutia $O(N)$

Se observa ca daca toate valorile V_i ar fi > 1 solutia optima ar fi sa se inmulteasca toate numerele. Este evident ca valorile $V_i=0$ nu afecteaza rezultatul, astfel ca se pot ignora de la inceput. In schimb, valorile $V_i=1$ pot afecta rezultatul putand fi folosite in paranteze la adunare. Se mai observa ca numarul termenilor dintr-o paranteza este foarte mic, deoarece in general este mai avantajos sa se inmulteasca numerele decat sa se adune. Astfel, putem aplica aceeasi dinamica de la solutia $O(N^2)$, dar care foloseste doar un numar constant de termeni intr-o paranteza (maxim `5` in solutia oficiala, desi `3` este de ajuns). Este necesar sa se efectueze dinamica cu numar constant de termeni abia dupa ce au fost eliminate elementele nule pentru a functiona.

perm - soluţie

Orice permutare poate fi descompusă în produs de cicluri $p=c_1c_2...c_k$
Să notăm cu lg_i=lungimea ciclului `i`.
Gradul permutarii este egal cu $cmmmc(lg_1, lg_2, ..., lg_k)$.

Prin urmare trebuie să determinăm o partiţie a lui `n` cu proprietatea ca cmmmc al elementelor partiţiei este maxim.
Odată determinată această partiţie (cu alte cuvinte determinate lungimile ciclurilor permutării), pentru a obţine prima permutare de grad maxim în ordine lexicografică vom construi permutarea astfel:
1. Considerăm că $lg_1<=lg_2<=...<=lg_k$

2. Pentru ca permutarea să fie minimă din punct de vedere lexicografic, ciclul 1 va conţine elementele 1, 2, ..., lg_1, pe care le vom plasa în permutare în ordinea:
2 3... lg_1 1
Dacă lg_1=1, atunci p[1]=1.
Ciclul al doilea va conţine elementele lg_1+1, ..., lg_1+lg_2, pe care le plasăm în permutare în ordinea:
lg_1+2, lg_1+3, ..., lg_1+lg_2, lg_1+1.
Dacă lg_2=1, atunci p[2]=2.
etc.

Rămâne să analizăm cum determinăm o partiţie a lui n cu proprietatea că cmmmc este maxim.
Să amintim câteva relaţii:
cmmmc(a,b)=a*b/cmmmdc(a,b).

Să considerăm o partiţie a lui n: $lg_1+lg_2+\ldots lg_k=n$, unde $lg_i \geqslant 1$.
Să considerăm descompunerea în factori primi a lui cmmmc:
$cmmmc(lg_1, lg_2, \ldots, lg_k)=p_1^{a1} p_2^{a2}\ldots p_m^{am}$.
cmmmc este maxim dacă:
$cmmdc(lg_i, lg_j)=1$, pentru orice $i \neq j$
Prin urmare $lg_i=p_i^{ai}$.

Vom numi partiţie a lui n de tip **P** o partiţie de forma
$n=1+1+\ldots 1+p_1^{a1} + p_2^{a2}+\ldots+p_m^{am}$
O partiţie de tip P se numeşte optimală dacă produsul $p_1^{a1} p_2^{a2}\ldots p_m^{am}$ este maxim (valoarea acestui produs fiind costul partiţiei).

Să considerăm de asemenea şirul numerelor prime $\leqslant$ n, memorate în ordine crescătoare în vectorul Prim.
Notăm lg numărul de numere prime $\leqslant$n.
Notăm
C(x,k) = costul maxim al unei partiţii a lui x de tip P, în care factorii primi sunt $\leqslant$ Prim[k].

$C(x,0)=1$
$C(x,k)=\max\{ C(x,k-1)$
$\quad Prim[k]^i C(x-Prim[k]^i, k-1)$, unde i astfel încât $1<Prim[k]^i \leqslant n$ $\}$

Soluţia o vom obţine în C(n,lg).
Pentru reconstituirea soluţiei, vom reţine în Sol[n][k]=0, daca C(n,k)=C(n, k-1) sau i, dacă $C(n,k)= Prim[k]^i C(n-Prim[k]^i, k-1)$.

Observăm că nu este necesară memorarea întregii matrice C, este suficient să reţinem doar coloana curentă şi coloana următoare.

```
for (i=0; i<=n; i++) C[i][0]=1;
for  (k=1; k<=lg; k++)
      {
        for (i=1; i<=n; i++)
             {C[i][k] = C[i][k-1]; sol[i][k]=0;
              for (t=Prim[k]; t<=i; t*=Prim[k])
                    if (C[i][k]<t*C[i-t][k-1])
                         {sol[i][k]=t;
                          C[i][k]=t*C[i-t][k-1];}
         }
```