Xp – descrierea soluţiei

Autor: Asist. univ. dr. ing. Mugurel Ionuţ Andreica – Universitatea Politehnica din Bucureşti

<u>Soluţia 1</u>
Complexitate de timp: O(N^2)
Memorie: O(1) sau O(N)
Punctaj: 30 puncte

Fiecare valoare `Prod[i]` este calculată independent (în timp `O(N)`).

<u>Soluţia 2</u>
Complexitate de timp: O(N)
Memorie: O(N)
Punctaj: 50 puncte

În cadrul acestei soluţii se vor genera toate valorile `val[i]` şi se vor memora într-un vector. Apoi vom calcula valorile **`PR[i]=(val[i]·…·val[N]) mod Q`**, în timp `O(N)` per total. Avem **`PR[N]=val[N]`** şi **`PR[1≤i≤N-1]=(val[i]·PR[i+1]) mod Q`**.
Apoi vom parcurge valorile `val[i]` de la `1` la `N`, menţinând pe parcurs produsul primelor `i-1` elemente. Când ajungem la poziţia `i`, fie `PL` acest produs. Atunci vom avea **`Prod[i]=(PL·PR[i+1]) mod Q`** (unde considerăm `PR[N+1]=1`).
Astfel, putem calcula toate valorile `Prod[i]` în timp `O(N)` per total.

<u>Soluţia 3</u>
Complexitate de timp: O(N·log(N))
Memorie: O(log(N))
Punctaj: 100 puncte

Această soluţie foloseşte tehnica de programare denumită ***Divide et Impera***. Se va scrie o funcţie recursivă **`divide_et_impera(i, j, A[i], B[i], outProd)`**, în care *`outProd`* va reprezenta produsul elementelor din afara intervalului de poziţii `[i,j]` (bineînţeles, *modulo Q*). Vom efectua primul apel *`divide_et_impera(1, N, A[1], B[1], 1)`*.
În cadrul funcţiei se vor efectua următoarele procesări. Dacă `i=j` atunci avem `Prod[i]=outProd`. Altfel, fie `mij=(i+j)/2`. Vom calcula produsul elementelor `val` de pe poziţiile `i, …, mij` (pornind de la `A[i]` şi `B[i]`, care sunt parametrii ai funcţiei). Fie acest produs `PL` şi fie `A[mij+1]` şi `B[mij+1]` valorile ce pot fi calculate imediat din `A[mij]` şi `B[mij]` (`A[mij]` şi `B[mij]` sunt obţinute la sfârşitul traversării tuturor elementelor de pe poziţiile `i, …, mij`). Apelăm apoi `divide_et_impera(mij+1, j, A[mij+1], B[mij+1], (outProd·PL) mod Q)`.
Apoi, pornind de la `A[mij+1]` şi `B[mij+1]`, vom calcula `PR`=produsul tuturor elementelor `val` de pe poziţiile `mij+1, …, j` (modulo `Q`).
La final apelăm `divide_et_impera(i, mij, A[i], B[i], (outProd·PR) mod Q)`.
Complexitatea de timp a soluţiei este `O(N·log(N))`, iar memoria folosită este de ordinul `O(log(N))` (deoarece dimensiunea stivei de apeluri recursive ajunge la `log(N)`).

<u>Soluţia 4</u>
Complexitate de timp: O(N)
Memorie: O(sqrt(N))
Punctaj: 100 puncte

Această soluţie se bazează pe soluţia `2`, însă memorează vectorul `PR` doar din `K` în `K` poziţii. Când ajungem la o poziţie `i`, avem produsul `PL` al elementelor dinaintea lui `i`, valoarea `PR[h]` a următoarei poziţii `h>i` memorate din vectorul `PR` (`h` este la distanţă de cel mult `K` de `i`), precum şi un alt vector cu valorile elementelor de pe poziţiile de la `i` până la `h`. Procesând valorile cu atenţie, putem obţine o complexitate liniară (`O(N)`), iar memoria folosită este de ordinul `K+N/K`. Valoarea minimă a memoriei utilizate se obţine pentru `K=sqrt(N)`.