DESCRIEREA SOLUȚIILOR
CONCURSUL URMAȘII LUI MOISIL
CLASELE XI-XII

Problema 1: Perioade

Propusă de: stud. Ștefan-Cosmin Dăscălescu, Universitatea din București

Mai întâi, se poate observă că numărul de poziții care pot fi actualizate în query-uri este unul relativ mic, valorile cu cel mult k biți putând fi precalculate folosind un algoritm de tip backtracking. O altă abordare posibilă pentru precalculări este de a ține într-un *set* toate valorile posibile ce apar în update-uri și query-uri și să se folosească o căutare binară pentru a afla poziția unde facem actualizările în structura de date.

Soluții parțiale. În funcție de valorile date în query-uri, se pot folosi abordări de tipul brute-force pentru actualizarea pozițiilor, care țin cont fie de faptul că pozițiile sunt mici, fie de faptul că numărul de valori actualizate este foarte mic ($k = 2$).

De asemenea, pentru subtaksul 3, putem folosi un arbore de intervale sau un arbore indexat binar pentru fiecare literă pentru a procesa actualizările și operațiile.

Soluția de 100 de puncte. Pentru obținerea punctajului maxim, vom reține doar valorile din pozițiile ce se pot actualiza într-o structură de date ce ne permite update-uri și query-uri rapide, iar pentru a afla răspunsul corespunzător unui query, vom afla numărul de litere de acel fel ce ar fi existat în mod normal în acel interval, număr la care vom adăuga diferența dintre numărul de caractere ce ar trebui să fie pe pozițiile cu cel mult k biți de 1 și numărul de caractere ce există de fapt pe acele poziții, răspuns ce se calculează cu ajutorul structurii de date menționate anterior.

Problema 2: WeightDif

Propusă de: stud. Ovidiu Rața, Universitatea Alexandru Ioan Cuza din Iași

Cea mai eficientă soluție care nu folosește tehnica necesară pentru a obține 100 de puncte presupune găsirea arborelui parțial de cost minim selectând doar anumite muchii. Se sortează muchiile crescător după cost și pentru fiecare muchie se formează un graf cu n noduri și doar muchia respectivă, adăugând muchia imediat următoare în ordine crescătoare până graful devine conex. Pentru fiecare graf, răspunsul este diferența minimă dintre costul celei mai din stânga muchii selectate și al celei mai din dreapta muchii selectate (cea pentru care care graful a devenit conex). Pentru a verifica dacă graful este conex se foloseste o structură de păduri de mulțimi disjuncte (Disjoint Set Union - DSU).

Pentru a obține 100 de puncte trebuie optimizată soluția anterioară. Atunci când se schimbă muchia de start în loc de formarea unui nou graf se face rollback la muchie in DSU. Se foloseste căutarea binară pe diferența maximă dintre costurile muchiilor pentru a vedea la ce moment de timp se face rollback la fiecare muchie și se aplică algoritmul de Dynamic Connectivity.

Complexitatea este $\mathcal{O}(M \cdot \lg(N) \cdot \lg(MAXCOST))$