Soluţie Intervale

Soluţie O(N^2) – 20-30 de puncte

Pentru fiecare indice i (1 <= i <= N) numărăm câte intervale compacte au ultimul element in i. Parcurgem permutarea cu un j în urmă şi reţinem pe parcurs minimul (min) şi maximul (max) din intervalul [j, i]. Dacă max-min = i-j, atunci secvenţa este interval compact.

Soluţie O(N*sqrt(N)) – 60 de puncte

Vom încerca să optimizăm soluţia descrisă mai sus. Pentru aceasta, considerăm în pasul curent ne aflăm la indicele i. Definim următoarele notaţii:

- Max[j] = maximul pe intervalul [j, i]
- Min[j] = minimul pe intervalul [j, i]
- Exp[j] = j + Max[j] – Min[j] = poziţia unde ar trebui să se termine secvenţa ce începe pe j ca să fie interval compact (luând în calcul doar elementele din permutare aflate între poziţiile i şi j)

Se observă că în vectorii Min şi Max apar „platouri” continue având aceeaşi valoare. Treptele pot fi identificate folosind câte o stivă. Astfel, nu este necesară memorarea explicită a vectorilor Min şi Max, iar operaţiile asupra lor (şi implicit asupra lui Exp) vor fi la nivel de interval. La trecerea de la pasul i-1 la pasul i, elementul P[i] poate mări Max sau micşora Min pentru un număr de poziţii j. Dacă pentru o poziţie j, Max[j] este înlocuit cu P[i] atunci Exp[j] este incrementat cu P[i] – Max[j] (similar în cazul micşorării lui Min, Exp[j] este incrementat cu Min[j] – P[i]). Aşadar, la un pas trebuie incrementate anumite intervale din Exp cu o valoare fixă x (datorită platourilor de valori egale din Min şi Max). În total au loc O(N) astfel de actualizări, deoarece un interval este actualizat atunci când un element este scos din una dintre cele două stive.

Pentru a rezolva problema, trebuie găsită o structură de date care să permită realizarea următoarelor două operaţii în timp cât mai bun:

- update(x, y, z) → se incrementează valorile din intervalul [x, y] cu z
- query(x) → care este numărul de apariţii al lui x? (la fiecare pas se va apela query(i) pe vectorul Exp)

Aceste operaţii pot fi realizate în complexitate O(sqrt(N)), dacă se procedează asemănător ca la problema arbore (http://infoarena.ro/problema/arbore, soluţie: http://infoarena.ro/preoni-2006/finala/solutii), cu diferenţa că nu va trebui verificat dacă o valoare există în şir, ci va trebui numărat de câte ori apare aceasta. Pentru a realiza acest lucru, pe fiecare bucată de sqrt(N) valori din vectorul Exp se va construi un hashtable. Memoria folosită per total este O(N) (se vor reţine sqrt(N) hash-uri având sqrt(N) valori).

Soluţie O(N*log(N)) – 100 de puncte

Soluţia de 100 de puncte nu se bazează pe abordarea folosită anterior. Se observă că numărul de intervale compacte este N + numărul de intervale compacte ce conţin cel puţin două elemente. Pentru a le număra pe acestea din urmă, observăm că se împart în două categorii: cele care au elementul minim mai spre stânga de cel maxim şi cele care au elementul maxim mai spre stânga de cel minim. Vom prezenta modul în care se numără intervalele compacte facând parte din prima categorie (pentru cealaltă categorie se procedează analog).

Iniţial, vom precalcula pentru fiecare element din permutare cea mai apropiată valoare mai mare şi, respectiv, mai mică decât cea curentă, atât spre stânga, cât şi spre dreapta.

Se parcurge permutarea de la stânga la dreapta. Pe parcurs se va calcula (din nou) "stiva maximelor" (stiva utilă pentru a determina la fiecare pas cel mai apropiat element din stânga mai mare decât elementul curent). Fie i indicele curent. Vom număra câte intervale compacte au ca minim pe P[i]. Se observă că aceste intervale pot avea valoarea maximă doar un element din stivă. De asemenea, maximul trebuie să se afle spre stânga de cel mai din dreapta element din stânga poziţiei i mai mic ca P[i]. În caz contrar, P[i] nu va fi minimul intervalului ce conţine poziţia i şi maximul respectiv. Cu aceste observaţii se poate construi o soluţie brute-force ce parcurge stiva şi intersectează intervalul pe care elementul din stivă este maxim (se cunoaşte din preprocesarea iniţială) şi intervalul pe care P[i] este minim. Dacă intervalul are Max – P[i] + 1 elemente, el este compact.

Această soluţie poate fi optimizată la complexitatea timp dorită, împărţind problema în 3 subprobleme după cazurile care reies din intersecţia intervalului de maxim cu cel de minim (intervalul de maxim este complet inclus in cel de minim, capătul din dreapta al intervalului de maxim este în afara celui de minim, dar cel din stânga este in interiorul acestuia şi cazul în care intervalul de maxim cuprinde intervalul de minim). Cel de-al 4-lea caz se poate fi considerat subcaz al unuia din cazurile menţionate anterior. A 3-a subproblemă nu este dificilă, deoarece soluţia poate fi cel mult 1 şi un test de existenţă este suficient.

Pentru a rezolva prima problemă, este necesară precalcularea minimelor aşteptate pentru fiecare element din „stiva de maxime". Explicaţia este următoarea: intervalul rezultat în urma intersecţiei este determinat de zona pe care elementul din stivă este maxim (fie acest interval [j, k]) şi atunci minimul aşteptat este max-k+j (formula reiese din max-min = k-j). Se observă că dintre toate aceste valori trebuie considerate doar cele care sunt incluse în intervalul [1, N]. Valorile minimelor aşteptate ar putea fi astfel stocate într-un vector de frecvenţă, dar rămâne problema numărării doar acelora care corespund unor intervale incluse în intervalul de minim şi unor elementele aflate în stivă. Pentru a rezolva această problemă, se vor folosi N stive (în locul vectorului de frecvenţă), iar răspunsul va fi căutat binar în stiva corespunzătoare lui P[i]. Stivele vor fi actualizate simultan cu „stiva de maxime". Subproblema 2 se rezolvă asemănător, ţinând cont că partea din dreapta a intervalului rămâne fixă.

Autorii soluţiilor: Cosmin Gheorghe, Paul-Dan Băltescu