Descriere problema Permutare, Lot 2020, Baraj 1

Tamio-Vesa Nakajima

February 2024

(Aceasta este o descriere neoficială a soluției acestei probleme, realizată ulterior autor. Nu reprezintă neapărat ideile comisiei științifice a acelui concurs.)

Cerință. Considerăm o permutare $p_1, \ldots, p_n$. Definim valoarea permutării $v(p)$ în felul următor: care este mărimea minimă a setului $S \subseteq [n-1]$* necesar astfel încât, dacă avem voie să interschimbăm elementul i cu elementul $i+1$ din p pentru oricare $i \in S$, atunci putem sorta permutarea S. Permutarea se schimbă de mai multe ori, comisia interschimbând elemente (potențial ne-adiacente) din p. După fiecare schimbare (și inițial) se vrea să calculăm valoarea permutării.

Idei. În primul rând vrem să înțelegem mai bine valoarea $v(p)$. Să considerăm un element p_i. Observăm că elementul p_i va trebui să plece la poziția p_i — deci e necesar neapărat să introducem în setul S toate elementele de la i la p_i. Se poate observa mai departe că aceste poziții sunt și suficiente pentru a putea sorta mereu șirul.

Astfel, putem să reformulăm problema în felul următor: să se găsească mărimea reuniunii tuturor intervalelor cu capetele în i și în p_i, la fiecare moment de tip. Observăm că după o interschimbare, se modifică exact două intervale; echivalent putem considera că se elimină două intervale, și se adaugă două intervale. Așadar am redus problema la o subproblemă, pe care o vom trata acum.

Subproblema critică. Considerăm o mulțime de intervale I. Fiecare interval e de forma $\{a, \ldots, b\} \subseteq [n]$. Avem voie să inserăm sau să eliminăm intervale după cum dorim. Vrem să știm care este mărimea mulțimii $\bigcup_{S \subseteq I} S$. Cum putem aborda această subproblemă?

Mai întâi să considerăm șirul $a[1], \ldots, a[n]$, unde $a[i] = \#\{S \in I \mid i \in S\}$, adică $a[i]$ este numărul de intervale din I în care apare i. Observăm mai întâi că $a[i] \geq 0$; mai mult, soluția este egală cu numărul de elemente din $a[i]$ care nu sunt egale cu 0.

*Definim $[n] = \{1, \ldots, n\}$.

La fel de bine, putem căuta numărul de elemente din $a[i]$ care *sunt* egale cu 0 (scădem la final acest număr din n); mai mult, putem doar să calculăm minimul șirului $a[i]$ și de câte ori apare el (căci dacă minimul este o atunci știm de câte ori apare, și dacă minimul este > 0 atunci 0 nu apare deloc; iar cum $a[i] \geq 0$ minimul nu poate fi < 0). Așadar, este suficient să găsim o structură de date ce poate găsi minimul lui a, și de câte ori apare.

Să considerăm acum cum afectează o inserare sau o eliminare a unui interval din I șirul a. Nu e greu de văzut că o inserare corespunde cu o incrementare a unei subsecvențe continue din a, și o eliminare cu o decrementare. Așadar, este suficient să găsim o structura de date ce poate simula incrementări/decrementări de subsecvențe din șirul a, și ce poate găsi eficient minimul șirului a și de câte ori apare minimul. Aceste operații se pot face cu un arbore de intervale cu propagare lazy.[†]

[†]Recomand această implementare: `https://codeforces.com/blog/entry/18051`.