**DESCRIEREA SOLUȚIILOR, OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, LOT SENIORI**

# Problema Minimize

*Propusă de: Tamio-Vesa Nakajima, Andrei-Costin Constantinescu, Lucian Bicsi*
*Mulțumim lui Ioan Popescu pentru pregătirea problemei*

Să ne imaginăm că modificăm o subsecvență continuă de elemente — care este modul optim de a seta acești biți? Cum fiecare bit e independent, putem să considerăm doar cazul unde numerele au un singur bit. Dacă bitul de înaintea primului bit schimbat și primul de după (sau 0, dacă acești biți nu există) sunt egali, atunci se pot seta toți biții să fie egali. Altfel, undeva în șirul de biți schimbați trebuie să fie o pereche de biți diferiți adiacenți; iar optim va fie exact un bit de acest fel. Dar, atunci costul va fi egal cu `xor`-ul numerelor din stânga respectiv dreapta numerelor schimbate! Astfel deducem următorul rezultat.

**Observation 1.** *Costul minim care poate fi realizat dacă ne hotărâm că schimbăm anumite valori din șir va fi egal cu costul șirului dacă numerele schimbate ar fi eliminate din șir. Astfel, costul minim realizabil dacă se modifică $k$ numere este gal cu costul minim realizabil dacă se elimină $k$ numere.*

Să presupunem că soluția optimă va elimina mai multe subsecvențe disjuncte de numere, printre care și o subsecvență de lungime $l$. Secvențele nu vor fi nici măcar adiacente. Fixând celelalte $k-l$ numere eliminate, să considerăm toate subsecvențele posibile de lungime $l$ care ar putea să fie alese. Dacă am putea, am vrea să eliminăm mereu subsecvența de lungime $l$ care îmbunătățește costul cât mai mult. Asta s-ar putea să nu fie posibil, pentru că subsecvențele trebuie să fie disjuncte și non-adiacente! Observăm că, pentru fiecare dintre cele $k-l$ elemente deja eliminate, ne este interzis să alegem $l+2$ secvențe de lungime $l$ (prima este cea care s-ar opri fix înainte de elementul eliminat, ultima este cea care începe fix după). Astfel, cu siguranță secvența de lungime $l$ cea mai bună pe care am putea s-o alegem este printre cele mai bune $(k-l)(l+2)+1$ secvențe de lungime $l$. Deducem deci:

**Observation 2.** *Pentru a construi o soluție optimă, trebuie doar să considerăm, pentru fiecare lungime $1 \leq l \leq k$, doar primele $(k-l)(l+2)+1$ secvențele de lungime $l$ care, ordonate după cât de mult îmbunătățesc costul dacă ar fi eliminate.*

Observăm că putem menține aceste secvențe ușor după fiecare modificare. O modificare va schimba doar $k^2$ secvențe, din punctul de vedere a cât de mult îmbunătătesc costul dacă ar fi eliminate. Putem itera efectiv prin acestea, menținând douâ heap-uri pentru fiecare lungime de secvență, sortate după cât de mult îmbunătățeste costul o subsecvență dacă este eliminată. Un heap va fi pentru cele mai bune $l+2$ secvențe, altul pentru restul. Astfel, o modificare poate fi realizată cu complexitatea $O(k^2 \log n)$, iar după modificare vom cunoaște efectiv toate secvențele care ne interesează.

Acum, cum putem răspunde eficient la interogări? Observăm că sunt, per total, $O(k^3)$ secvențe care ne interesează. Să considerăm dinamica $d$ unde $d_{ij}$ reprezintă optimul pentru primele $i$ elemente dacă sunt eliminate deja $j$ elemente. Dacă considerăm doar cele $O(k^3)$ secvențe care știm că sunt suficiente pentru soluția optimă, observăm că dinamica se schimbă doar la $O(k^3)$ $i$-uri, și trebuie găsită tot doar pentru $O(k^3)$ $i$-uri, mai exact capetele drepte, respectiv stângi ale intervalelor cu pricina. Așadar, putem calcula dinamica doar la aceste poziții, folosind doar intervalele găsite anterior. Cum recalcularea pentru un interval anume se poate face în $O(k)$, rezulta complexitatea $O(k^4)$ pentru fiecare interogare.