# Descrierea Soluțiilor
# Concursul Național "InfoPro", Runda 3
# Grupa A

## 1 Problema HalfTree

*Propunator: Bogdan Sitaru, University of Oxford*

Observăm că, dacă o muchie este parte din lanțul afectat, ea mereu va schimba costul total al arborelui independent de celelalte muchii afectate. Fie $\delta(m)$ cu cât schimbă costul muchia $m$ dacă aceasta este afectată.

Care este valoarea lui $\delta(m)$? Să presupunem că, în urma eliminării muchiei $m$, se formează doi subarbori $A$ și $B$. Totodată, fie $c$ costul muchiei $m$. Atunci, observăm că distanța între oricare nod din $A$ și oricare nod din $B$ este modificată cu $c/2$. Reiese următoarea formulă:

$$\delta(m) = \underbrace{|A|}_{\text{Un nod din } A} \times \underbrace{|B|}_{\text{Un nod din } B} \times \underbrace{\frac{c}{2}}_{\text{Valoarea modificării}}$$

Acum observăm că soluția optimă este aceea pentru care sumă $\delta(m)$-urilor pentru $m$-urile din lanțul selectat este maximă – căci aceasta corespunde unui cost final minim.

**Soluție în $O(N^3)$.** Selectăm două noduri $x$ și $y$ drept capete ale lanțului și calculăm valoarea acestuia.

**Soluție în $O(N^2)$.** Selectăm un nod $x$ și calculăm printr-un DFS valoarea tututor lanțurilor care îl au pe $x$ drept unul dintre capete.

**Soluție în $O(N)$ pentru costuri pozitive.** Pentru aceste subtask-uri, se poate calcula diametrul arborelui având $\delta(m)$ drept costuri pe muchii.

**Soluție în $O(N)$.** Pentru a obține punctaj maxim, vom înrădăcina arborele într-un nod (de exemplu nodul 1). Se va calcula $dp_{nod}$ ca valoarea maximă a unui lanț "în jos" care începe în nodul $nod$ . Recurența este $dp_{nod} = max(0, dp_{nxt} + \delta(nod, nxt))$ pentru orice $nxt$ fiu al lui $nod$. Pentru soluție, vom considera pentru fiecare nod lanțul cu valoarea maximă care îl are pe nodul $nod$ drept cel mai puțin adânc, acesta fiind format din suma a 2 lanțuri "în jos" începute în "nod" (cele cu valoarea cea mai mare).

## 2 Problema Adunare

*Propunator: Tamio-Vesa Nakajima, Oxford University*

Problema ne dă trei rânduri de cifre, pe care se aplică operația de schimbare a unei cifre. După fiecare operație, și înainte de toate operațiile, se vrea să afișăm numărul de moduri de a șterge coloane astfel încât numerele formate de cifrele din primele două rânduri, adunate, să fie egale cu numărul format de cifrele din al treila rând.

**Subtaskul 1.** Aici se vrea rezolvarea problemei in $O(N2^N)$. Efectiv se enumeră toate modurile de a șterge submulțimi de coloane (de care sunt $2^N$), și se verifică dacă sunt valide (în $O(N)$). Cum nu sunt operații de schimbare, nu trebuie calculat decât odată această valoare.

**Subtaskul 2.** Aici se vrea rezolvarea problemei in $O(N)$ per operație de schimbare. Este suficient să recalculăm, de fiecare dată, rezultatul, fară a ne refolosi de informațiile din starea precedentă a cifrelor, după fiecare operație. Astfel, voi descrie modul în care se poate calcula odată răspunsul, in $O(N)$, iar acest algoritm se aplică de $Q + 1$ ori.

Pentru a calcula răspunsul in $O(N)$, vom observa că coloanele pot fi de 5 feluri, în funcție de generarea și utilizarea restului. În enumerarea de mai jos, când spun că o coloană generează rest, înseamnă că dă rest mai departe (în stânga), și când spun că necesită rest, înseamnă ca trebuie să primeasca rest (din dreapta) pentru a fi validă.

1. O coloană care nu necesită și nu generează rest, de exemplu $\begin{matrix}1\\2\\3\end{matrix}$.
2. O coloană care necesită dar nu generează rest, de exemplu $\begin{matrix}0\\2\\3\end{matrix}$.
3. O coloană care nu necesită dar generează rest, de exemplu $\begin{matrix}9\\4\\3\end{matrix}$.
4. O coloană care necesită și generează rest, de exemplu $\begin{matrix}9\\3\\3\end{matrix}$.
5. O coloană invalidă indiferent de ce rest primește, de exemplu $\begin{matrix}0\\0\\3\end{matrix}$.

Coloana de tipul 5 o voi nota cu $\times$; în rest, dacă o coloană necesită restul $x$ și generează restul $y$, o voi nota cu $y \leftarrow x$ (sensul fiind că coloana pornește cu restul $x$ si ajunge la restul $y$). Observația importantă este că o secvență de coloane este corectă dacă și numai dacă coloanele, sub notația asta, arată ca un drum valid de la 0 la 0 în graful complet cu nodurile 0 și 1; de exemplu, coloane de tipurile $0 \leftarrow 1, 1 \leftarrow 1, 1 \leftarrow 0$ ar fi valide. De ce? Pentru că această condiție este echivalentă cu

a zice că coloanele se potrivesc două câte două, și nu necesita inițial, sau generează la final, rest. Evident coloanele $\times$ nu au voie să apară.

Astfel, problema devine: se da un șir de $\times$ și $x \leftarrow y$. Câte subșiruri din acest șir formează drumuri de la 0 la 0 în graful complet cu nodurile 0 și 1? Putem acum formula o soluție simplă cu programare dinamica: vom calcula un șir $d[i][j]$, cu semnificația "câte subșiruri ale primelor $i$ caractere formează drumuri valide de la 0 la $j$ în graful complet cu două noduri?" La final, soluția se găsește în $d[N][0]$.

**Subtaskul 3.** Aici se vrea complexitatea $O(N \log N + Q \log N)$. Putem rezolva cu tehnica arborilor de intervale. Într-un arbore de intervale, fiecare nod reprezintă o subsecvență din șirul pe care îl reprezentăm. Astfel, în nodul ce reprezintă subsecvența $s$, vom ține valorile $f(s, i, j)$ pentru $i, j \in \{0, 1\}$, unde $f(s, i, j)$ reprezintă numărul de subsecvențe ale lui $s$ care sunt drumuri valide de la $j$ la $i$ în graful complet cu nodurile 0, 1. Arborele de intervale poate fi ținut pentru că putem calcula aceste valori ușor când $s$ are lungimea 1:

$$f(\times, i, j) = 0$$
$$f(x \leftarrow y, i, j) = \begin{cases} 1, & x = i, y = j \\ 0, & \text{altfel} \end{cases}$$

Iar, dacă cunoaștem valorile $f(s, i, j)$, $f(t, i, j)$, putem afla aceste valori și pentru concatenarea lui $s$ și $t$:

$$f(st, i, j) = \underbrace{f(s, i, j)}_{\text{Subșiruri din } s} + \underbrace{f(t, i, j)}_{\text{Subșiruri din } t} + \underbrace{f(s, i, 0)f(t, 0, j) + f(s, i, 1)f(t, 1, j)}_{\text{Subșiruri din ambele}}$$

Arborele de intervale poate apoi să mențină aceste valori în complexitatea cerută. Observăm că dacă $s$ este șirul întreg de coloane, atunci $f(s, 0, 0)$ este soluția.

# 3 Problema Cosmar

*Propunator: Bogdan Ciobanu, Hudson River Trading*

Un prim pas pentru a rezolva problema este sa se gaseasca o structura cu care sa putem separa toate numerele toate au acelasi rest atunci cand sunt impartite cu o putere a lui 2. Aceasta conditie se poate reformula, daca am considera numerele in baza 2 ca niste siruri de caractere care incep de la cel mai nesemnificativ bit, atunci aceste numere ar avea prefix comun. O structura care ne permite natural o astfel de asociatie este un trie. Apoi trebuie inteles ce inseamna in contextul structurii atunci cand se schimba un bit. In cadrul unei operatii, alegerea unui rest inseamna alegerea unei cai din radacina arborelui pana intr-un nod, iar apoi schimbarea bitilor inseamna ca se schimba ordinea fiilor, sau mai bine zis, ordinea fiilor este de nedistins. Asadar, in problema initiala ne intereseaza echivalenta structurala intre cele doua. Aceasta echivalenta este un izomorfism si se poate rezolva, spre exemplu, prin a genera un reprezentant care ignora detaliile structurii care nu sunt relevante in cadrul echivalentei. O codificare pe care o putem genera in cazul problemei noastre este o paranteza a arborelui, pentru ca exista o bijectie intre ele. Pentru ca nu exista o ordonare a fiilor, vom alege o reprezentare canonica in care vom ordona fii dupa sirurile pe care le genereaza subarborele lor, in ordine lexicografica. Trebuie acordata atentie in plus pentru ca pot exista duplicate. Pentru a calcula codificarea, nu este nevoie sa se construiasca explicit structura de trie. O astfel de solutie ar lua 80 de puncte. Pentru ultimul subtask, se poate construi in mod implicit reprezentarea cu un algoritm in stil divide et impera, in care, pe fiecare nivel partitionam sirul in doua parti, in functie de valoarea pe care o are bitul. Ca sa intelegeti mai bine constructia implicita, va puteti gandi cum ati rezolva problema clasica de pereche de xor maxim, care in mod normal s-ar face cu trie, dar fara a construi structura.

**Echipa.** Setul de probleme pentru această rundă a fost pregatit de:

- prof. Adrian Panaete, Colegiul "A. T. Laurian", Botosani
- prof. Zoltan Szabó, Liceul Tehnologic "Petru Maior" Reghin / ISJ Mureș Tg. Mureș
- Andrei Constantinescu, student University of Oxford, Balliol College
- Bogdan Ciobanu, software engineer, Hudson River Trading
- George Chichirim, student University of Oxford, Keble College
- Tamio-Vesa Nakajima, student University of Oxford, University College
- Bogdan Sitaru, student University of Oxford, Hertford College