Descrierea Soluțiilor
Olimpiada Societății pentru Excelență și Performanță în Informatică, Etapa Județeană
Clasele XI-XII

1 Problema Polihroniade

Propunător: Andrei Constantinescu, University of Oxford

Problema Polihroniade a fost gandita initial ca problema de dificultate relativ mai scazuta a setului, deoarece aceasta nu necesita vreo tehnica cunoscuta in prealabil. Desi ea poate parea greu de abordat la prima vedere, observam in cele ce urmeaza cum cateva observatii, relativ simple, fac problema mai usor de atacat.

1.1 Operatiile Comuta

Fie i, j, k, l numere intre 1 si N, atunci operatiile O_1 = `L i j` si O_2 = `C k l` comuta. Cu alte cuvinte, ne este indiferent daca asupra matricei A efectuam intai O_1 si apoi O_2 sau invers (adica O_2 si apoi O_1). In mod inductiv, deducem ca daca problema admite o solutie, atunci ea admite o solutie in care intai permutam liniile lui A, si apoi coloanele lui A. Mai mult, in mod informal putem spune ca liniile si coloanele sunt independente, fiecare fiind o problema de sine statatoare. Asadar, este suficient sa explicam cum efectuam operatiile pe linii, cele pe coloane fiind rezolvate analog. Totusi, rezolvarea este inca neclara - avem nevoie de alte observatii.

1.2 Metoda Mersului Invers

Sa consideram problema inversa: plecam de la o tabla de sah si vrem ca prin interschimbari succesive de linii si/sau coloane sa ajungem la matricea A. Cum operatiile din problema sunt reversibile, problema inversa nu este cu nimic mai grea sau mai usoara decat cea pusa in discutie. Intr-o tabla de sah exista fix 2 tipuri de linii: cele de forma `0101...01` si, respectiv, cele de forma `1010...10`. Mai mult, avem fix $\frac{N}{2}$ linii din fiecare tip. Ce este acum vital sa observam este ca operatiile pe linii si coloane nu schimba foarte mult aceasta configuratie. Mai exact, indiferent ce interschimbari am efectua, matricea rezultata va avea $\frac{N}{2}$ linii identice si alte $\frac{N}{2}$ linii identice care sunt fix complementul primelor $\frac{N}{2}$ (adica au `0` unde era `1` si vice-versa). Bineinteles, in ambele tipuri trebuie sa avem exact $\frac{N}{2}$ biti de `0` si $\frac{N}{2}$ biti de `1`, cum interschimbarile de orice fel nu schimba continutul liniilor, ci doar ordinea elementelor din ele. In mod clar, aceleasi observatii sunt valabile si pentru coloane.

Date fiind cele spuse, avem acum o conditie necesara pentru ca problema sa aiba solutie pentru o matrice A data: matricea A trebuie sa aiba exact 2 tipuri de linii: $\frac{N}{2}$ de un tip, si $\frac{N}{2}$ de celalalt tip, unde tipurile sunt complementare si contin exact $\frac{N}{2}$ biti de `1` (automat, si $\frac{N}{2}$ biti de `0`) fiecare. De asemenea, aceeasi conditie trebuie sa fie indeplinita si pentru coloane. Ce este posibil suprinzator acum este ca aceste conditii sunt si suficiente, adica, cu alte cuvinte, matricea A poate fi transformata intr-o tabla de sah daca **si numai daca** aceste conditii sunt adevarate pentru instanta noastra. Verificarea acestor conditii se poate face in timp $O(N^2)$ si aduce cu sine 40 de puncte pe problema (credem noi, aceasta este cea mai dificila parte a problemei, si este, prin urmare, rasplatita generos ca punctaj).

1.3 Transformarea Matricei A in Tabla de Sah cu Numar Minim de Operatii

Pentru restul probemei, facem presupunerea ca matricea A se poate transforma in tabla de sah, asa cum este specificat, de altfel, si in enuntul problemei. Cheia rezolvarii sta in urmatoarea observatie: atata timp cat aranjam ca prima linie a matricei A sa alterneze (`0101...01` sau `1010...10`) si ca prima coloana a matricei A sa alterneze, restul matricei va fi, in mod garantat, o tabla de sah (acest lucru rezulta din observatia de mai sus legata de tipul liniilor: avem $\frac{N}{2}$ linii de fiecare tip, si tipurile sunt complementare; similar pentru coloane). Mai sus am stabilit ca liniile si coloanele sunt independente, deci cele doua probleme se pot trata separat. De asemenea, ele se pot reformula mai simplu: dat fiind un sir binar de lungime N format din $\frac{N}{2}$ caractere `0` si $\frac{N}{2}$ caractere `1`, transformati-l intr-un sir alternant folosind un numar minim de operatii de interschimbare a doua caractere.

Exista doar doua siruri binare alternante de acest tip: `0101...01` si `1010...10`. Fara a restrange generalitatea, vom incerca sa il obtinem pe primul, dar pentru o solutie completa trebuiesc incercate ambele si intoarsa solutia care duce la un numar minim de operatii. In sirul `0101...01` observam ca toate caracterele `0` sunt pe pozitii pare si toate caracterele `1` sunt pe pozitii impare. In sirul nostru pe care dorim sa il aducem la aceasta forma exista trei tipuri de caractere:

1. Caractere care sunt la locul lor - adica `0` pe pozitii pare si `1` pe pozitii impare.
2. Caractere `0` care sunt pe o pozitie impara (deci trebuie obligatoriu sa faca parte din cel putin o interschimbare). Sa notam multimea pozitiilor de acest tip din sirul nostru cu S_0.
3. Caractere `1` care sunt pe o pozitie para (deci trebuie obligatoriu sa faca parte din cel putin o interschimbare). Sa notam multimea pozitiilor de acest tip din sirul nostru cu S_1.

Reiterand, pozitiile din $S_0 \cup S_1$ trebuie sa faca parte dintr-o interschimbare pentru a le aduce pe o pozitie cu paritatea corecta. Observam ca sunt exact atatea caractere `0` care nu sunt la locul lor cate caractere `1` nu sunt la locul lor (notam pentru aceasta ca numarul de caractere `0` si de caractere `1` este egal), deci $|S_0| = |S_1|$. Prin urmare, cele de mai sus ne spun ca avem nevoie de cel putin $|S_0|$ interschimbari pentru a rezolva problema. Mai mult, o solutie cu fix $|S_0|$ interschimbari se poate construi: interschimbam pe rand cate o pozitie din S_0 cu una din S_1 pana cand epuizam ambele multimi (fiecare interschimbare practic pune la locul lor doua caractere). Asadar, aceasta solutie este si optima din punctul de vedere al numarului de interschimbari. Bineinteles, trebuie sa aplicam acest algoritm si pentru coloane, dar acest lucru nu schimba conceptual rezolvarea. In total, solutia se construieste in timp $O(N)$ dupa citire. Cum este posibil acest lucru daca complexitatea de citire a matricei este $O(N^2)$? Ei bine, orice matrice care se poate transforma in tabla de sah este unic determinata de prima sa linie si prima sa coloana, deci s-ar putea spune ca insasi formatul de intrare este inhibitor pentru obtinerea unei complexitati timp mai bune (daca ni s-ar fi dat doar prima linie si prima coloana problema se putea rezolva in $O(N)$, lucru mentionat aici mai mult

ca o curiozitate matematica decat ca o observatie necesara pentru rezolvarea problemei). Folosind aceasta solutie obtinem restul de 60 de puncte pe problema.

2 Problema Bob

Propunător: Costin-Andrei Oncescu, University of Oxford

Problema Bob a fost pusa in concurs ca o problema de tehnica, care nu este foarte dificila pentru cunoscatorii de programare dinamica.

2.1 Calculul Numarului Maxim de Pachete

Prima cerinta, valorand 31 de puncte, se preteaza la un algoritm de tip Greedy relativ simplu: Se itereaza prin zile in ordine crescatoare, la fiecare pas retinand pentru fiecare tip de cafea daca l-am intalnit deja in pachetul curent sau nu. In momentul in care avem in pachetul curent toate tipurile de cafea, putem inchide pachetul si sa deschidem unul nou, reluand algoritmul de unde am ramas. Daca dupa terminarea celor N zile ramanem cu un pachet incomplet de cafea, atunci il vom alipi ultimului pachet inchis. Complexitatea totala este dominata de citire, si este, deci, $O(NK)$.

2.2 Calculul Numarului de Impartiri Maxime in Pachete

Aceasta cerinta este mai dificila, necesitand generalizarea ideii de mai sus. In primul rand, am dori sa calculam

e_i = cel mai mic j astfel incat intervalul de zile $i \dots j$ sa contina fiecare tip de cafea cel putin o data

Observati cum, de la un punct incolo, valoarea e_i este nedefinita, deoarce nici macar intervalul $i \dots N$ nu contine toate tipurile de cafea - in acest caz luam $e_i = N + 1$ prin conventie, lucru pe care il simulam, pentru simplitate, presupunand existenta unei zile fictive $N + 1$ in care se produc toate tipurile de cafea.

Vom calcula e_i in ordine descrescatoare a zilelor. Mai exact, pe masura ce iteram i, vom mentine inca un indice j (initial egal cu $N + 1$), care va scadea pe masura ce i scade ("tehnica celor 2 pointeri"). In particular, vom mentine tot timpul invariantul ca intervalul $i \dots j$ contine toate tipurile de cafea ce putin o data. De asemenea, vom retine un vector de frecventa f_k = in cate zile din intervalul $i \dots j$ se produce tipul de cafea k, pentru $1 \leq k \leq K$. Acum, cand ajungem la o noua valoare a lui i, actualizam f cu tipurile de cafea din ziua i, si apoi incercam sa scadem j cat mai mult; adica scadem j cat timp $i \dots j - 1$ contine toate tipurile de cafea (lucru pe care il verificam inspectand tipurile de cafea din ziua j si vazand daca eliminarea lor din f ar pastra toate valorile din f pozitive). Prin constructie, algoritmul prezentat ne asigura ca la finalul fiecarui pas avem $j = e_i$. Pe parcursul executiei atat i cat si j scad de $O(N)$ ori, deci complexitatea acestui past din rezolvare este $O(NK)$ si calculeaza corect vectorul e.

Mai apoi, ne propunem sa calculam

m_i = numarul maxim de pachete de cafea care se pot obtine considerand doar zilele $i \dots N + 1$

In mod intuitiv, $m_{N+1} = 1$, deoarce ziua $N+1$ constituie un pachet de sine statator. In spiritul algoritmului Greedy din prima parte a problemei, se remarca relativ usor ca $m_i = 1 + m_{e_i+1}$, oricare

ar fi $1 \leq i \leq N$. Putem, asadar, calcula vectorul m in timp $O(N)$, iterand i in ordine descrescatoare. Bineinteles, $m_1 - 1$ reprezinta chiar raspunsul la prima cerinta, calculat pe o alta cale (scazatorul se datoreaza pachetului fictiv $N + 1$), dar acum avem informatii suplimentare ce ne vor fi de folos.

In cele din urma, vrem sa calculam

$$d_i = \text{numarul de impartiri intr-un numar maxim de pachete de cafea (adica } m_i \text{ pachete) a zilelor } i \dots N + 1$$

deoarece raspunsul la a doua cerinta a problemei este chiar d_1. Pentru a continua rezolvarea, avem nevoie de observatia ca m_i **este monoton descrescator in** i.

Din definitie, avem ca $d_{N+1} = 1$. Pentru restul valorilor i, calculul lui d_i se va face in ordine descrescatoare de la N la 1. Fixam acum o valoare i si vedem in cele ce urmeaza cum putem calcula d_i in functie de $d_{i+1}, d_{i+2}, \dots$. Orice impartire in cat mai multe pachete valide a zilelor $i \dots N$ se compune dintr-un interval de forma $i \dots j$, unde obligatoriu $j \geq e_i$, si inca $m_i - 1$ alte intervale. Pentru a ne asigura ca impartirea obtinuta este optima, j trebuie ales de asa natura incat $m_{j+1} = m_i - 1$ si $j \leq N$ (intervalele de forma $i \dots N + 1$ sunt artificiale si nu trebuie luate in calcul). Deoarece m este monoton descrescator, valorile $e_i \leq j \leq N$ pentru care $m_{j+1} = m_i - 1$ formeaza un interval $[j_0, j_1]$, ale carui capete pot fi cautate binar in timp $O(\log N)$.[1] Date fiind acestea, putem calcula d_i prin expresia $d_i = \sum_{j=j_0}^{j_1} d_{j+1}$. O implementare naiva a acestei idei duce la o complexitate de $O(N^2)$, si este insuficienta.

Din fericire, ultimul pas este simplu: suma mentionata poate fi calculata in $O(1)$ daca in timp ce calculam valorile d_i mentinem si vectorul de sume partiale pe sufixe ale lui d (suma respectiva devine apoi o diferenta a doua sume partiale). Complexitatea calculului vectorului d este astfel $O(N \log N)$ datorita cautarii binare (si poate fi adusa la $O(N)$ folosind ideea mai rafinata din nota de subsol), deci complexitatea pentru toata problema este $O(NK + N \log N)$, suficienta pentru 100 de puncte.

[1] Pentru cei mai bravi dintre voi, ele pot fi calculate si in timp total $O(N)$, folosind paradigma celor 2 pointeri, dar acest lucru nu este necesar pentru punctaj maxim.

3 Problema Dreptunghi

Propunător: Szabó Zoltan, Liceul Tehnologic Petru Maior Reghin / ISJ Mureș Tg. Mureș

Problema **dreptunghi** are 4 cerințe cu diferite grade de dificultate. Însă rezolvarea principală se bazează pe faptul că observăm că definiția recursivității permite salvarea informațiilor într-o structură de arbore binar.

Arborele binar are ca rădăcină informația Hk sau Vk, apoi se definește subarborele stâng, și apoi subarborele drept.

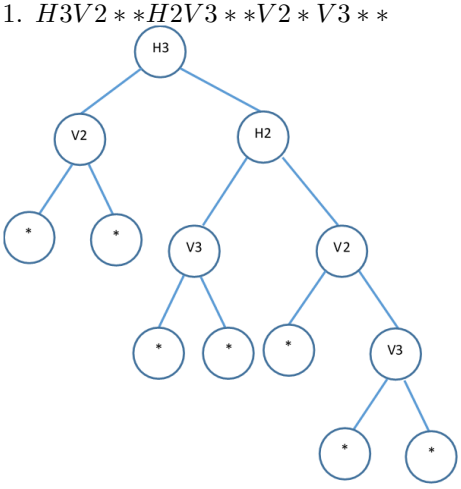
Informațiile din șirul de caractere se găsesc astfel în preodine, și putem să construim arborele.

1	1	2	2	2	2
1	1	2	2	2	2
1	1	2	2	2	2
3	3	3	4	4	4
3	3	3	4	4	4
5	5	6	6	6	7
5	5	6	6	6	7
5	5	6	6	6	7

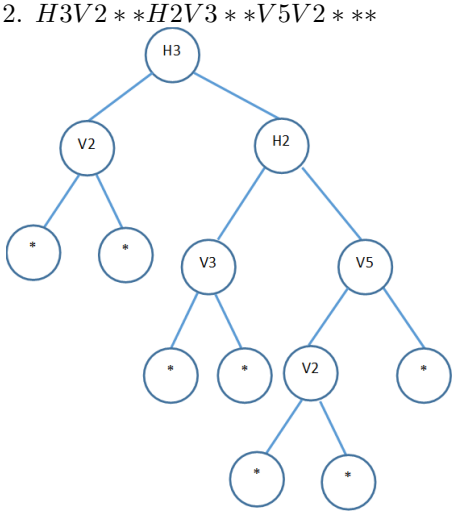
Pentru dreptunghiul de mai sus există 4 coduri, așa cum s-a prezentat în enunțul problemei. Fiecărui cod îi corespunde un arbore binar. În continuare vom desena fiecare arbore, iar apoi vom studia prorpietățile necesare pentru a rezolva cerințele problemei.

În continuare prezentăm arborii corespunzători celor 4 coduri de numerotare, luate în ordine lexicografică:

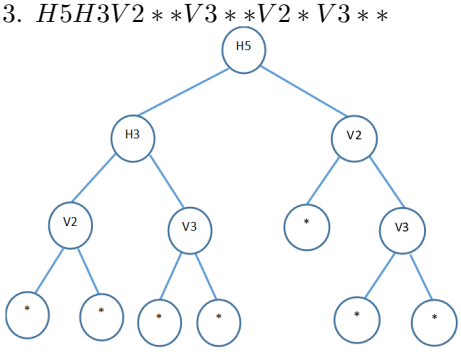
1. $H3V2**H2V3**V2*V3**$



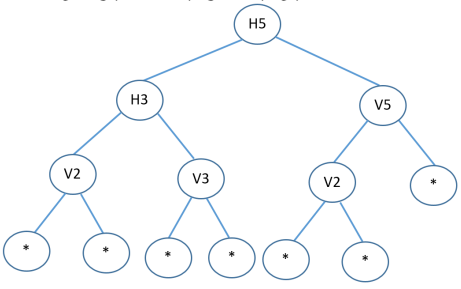
2. $H3V2**H2V3**V5V2***$



3. $H5H3V2**V3**V2*V3**$



4. $H5H3V2**V3**V5V2***$



3.1 Prima cerință (14 puncte)

$P = 1$ **(Numărul de subdiviziuni pe care codul C le generează)**

Răspunsul la această întrebare este numărul caracterelor '*' pe care le conține codul C.

Este cerința cea mai simplă a problemei și se poate rezolva prin simpla parcurgere a tuturor elementelor șirului de caractere citit și calcularea numărului de apariții al caracterului '*' în șir.

De asemenea din arborii desenați mai sus, observăm că numărarea unui dreptunghi se întâmplă când suntem la o frunză a arborelui, deci rezultatul cerut va fi numărul frunzelor arborelui binar pe care îl construim la citire. Acest rezultat se poate obține printr-o parcurgere în adâncime a arborelui.

3.2 A doua cerință (21 de puncte)

$P = 2$ **(Dimensiunile unui dreptunghi de arie minimă pentru care acest cod este valid)**

Studiind informațiile din arborii desenați mai sus, vom observa că:
Dacă nodul din rădăcina (sub)arborelui conține:

(a) o literă H și numărul k - asta înseamnă că

- dimensiunea orizontală a dreptunghiului trebuie să permită o linie la coordonata k, deci dreptunghiul va avea o dimensiune orizontală de cel puțin valoare k.
- toate liniile orizontale în subarborele stâng vor fi trasate în subdreptunghiul de sus de coordonate mai mici, deci **nu vor influența** dimensiunea orizontală a dreptunghiului actual.
- dimensiunea orizontală va fi afectată **cumulativ** de dimensiunea orizontală depistată în subarborele drept.
- dimensiunea verticală va fi maximul dintre dimensiunile verticale indicate de subarborele stang respectiv subarborele drept.

(b) o literă V și numărul k - asta înseamnă că

- dimensiunea verticală a dreptunghiului trebuie să permită o linie la coordonata k, deci dreptunghiul va avea o dimensiune verticală de cel puțin valoare k.
- toate liniile verticale în subarborele stâng vor fi trasate în subdreptunghiul din stânga de coordonate mai mici, deci **nu vor influența** dimensiunea verticală a dreptunghiului actual.
- dimensiunea verticală va fi afectată **cumulativ** de dimensiunea verticală depistată în subarborele drept.
- dimensiunea orizontală va fi maximul dintre dimensiunile orizontale indicate de subarborele stâng respectiv subarborele drept.

(c) un caracter '*' - asta înseamnă că

- este vorba despre cazul banal de dreptunghi minim 1×1.

3.3 A treia cerință (29 de puncte)

$P = 3$ **(Numărul de codificări distincte)**

Se știe că numărul arborilor binari cu N noduri este egal cu numărul lui Catalan de ordin N.

O parte conexă maximală a arborelui în care toate nodurile sunt etichetate identic cu litera H respectiv V, au proprietatea că permit rescrierea arborelui pentru o codificare echivalentă, prin schimbarea raportului Tată-Fiu al rădăcinii și a descendentului din stânga sau dreapta. Dacă partea conexă conține K noduri, atunci numărul arborilor pe care ii putem construi va fi $C(K)$, fiind numărul lui Catalan de ordin K. Rezultatul pentru problema noastră va fi produsul numerelor Catalan al numărului de noduri din *fiecare componentă conexă din arbore*, etichetate cu aceeași literă H sau V. Întrucât lungimea șirului de caractere ce reprezintă intrarea pentru problema noastră nu depășește 350 de carctere, rezultă cu nu suntem nevoiți să calculăm optim acest număr, chiar si o solutie calculată în $O(N^2)$ va intra satisfăcător în timpul de execuție.

Formula propusă este $C(n) = C(0) \cdot C(n-1) + C(1) \cdot C(n-2) + \ldots + C(n-1) \cdot C(0)$ pentru orice $n > 0$, și $C(0) = 1$.

Pentru exemplul nostru, observăm că oricare din subarbori are 6 noduri care formează 4 zone conexe:

- o zonă formată din 2 noduri etichetate cu H.
- o zonă formată din 2 noduri etichetate cu V.
- o zonă formată dintr-un nod etichetat cu V.
- o zonă formată dintr-un nod etichetat cu V.

Deci, rezultatul calculat va fi $C(2) \cdot C(2) \cdot C(1) \cdot C(1) = 2 \cdot 2 \cdot 1 \cdot 1 = 4$.

3.4 A patra cerință (36 de puncte)

$P = 4$ **(Primul cod în ordine lexicografică echivalent cu cel dat)**

Studiind arborii pe care i-am prezentat mai sus, observăm:

Arborele corespunzător primului cod în ordine lexicografică are proprietatea că toate nodurile arborelui care sunt etichetate cu V sau H au un nod fiu din stânga, iar eticheta acestui nod diferă de valoarea respectivă.

Sarcina noastră este ca să modificăm structura nodurilor astfel ca pentru niciun nod să nu existe etichetă identică de H sau V pentru nodul fiului din stânga.

Cu o parcurgere în adâncime vom depista toate aceste noduri, și vom rearanja nodurile din arbore, astfel încat descendentul din stânga să devină tată la vechiul tată, iar nodul tată se va transforma în fiu din dreapta.

Algoritmul va realiza în mod repetat modificarea arborelui, până când nu vor mai exista noduri cu această proprietate, apoi cu o parcurgere în preordine se va putea tipări răspunsul la cerința problemei, și anume primul cod în ordine lexicografică.

Echipa Setul de probleme pentru această rundă a fost pregătit de:

- prof. Adrian Panaete, Colegiul "A. T. Laurian", Botosani
- prof. Zoltan Szabó, Liceul Tehnologic "Petru Maior" Reghin / ISJ Mureș Tg. Mureș
- Adrian Emanuel Dicu, student Universitatea Politehnică din București
- Alexa Maria Tudose, student University of Oxford, University College
- Alexandra Maria Udristoiu, student Universitatea din București
- Andrei Constantinescu, student University of Oxford, Balliol College
- Bogdan Ciobanu, software engineer, Hudson River Trading
- Bogdan Sitaru, student University of Oxford, Hertford College
- Costin-Andrei Oncescu, student University of Oxford, St. John's College
- George Chichirim, student University of Oxford, Keble College
- Stelian Chichirim, student Universitatea din București
- Theodor Pierre Moroianu, student Universitatea din București