Olimpiada Societății pentru Excelență și Performanță în Informatică
Descrierea Soluțiilor
Clasa a IX-a

1 Problema Aproape

Propunător: Alexandru Petrescu, University of Oxford, Keble College

Soluție parțială – 70 de puncte

Este suficient să iterăm prin toate numerele care au același număr de cifre ca numărul N și să verificăm dacă respectă proprietățile unui număr aproape de N, respectiv unui număr aproape de cel puțin un număr aproape de N.

Soluție oficială – 100 de puncte

Vom începe prin a defini urmatoarele proprietăți ale numarului N:

- cnt_1 - numărul de cifre care pot fi fie doar incrementate cu 1, fie doar decrementate cu 1.
- cnt_2 - numărul de cifre care pot fi și incrementate și decrementate cu 1.
- $aproape_2$ - numărul de moduri în care putem modifica numărul N prin incrementarea sau decrementarea unei singure cifre cu 2.

Folosind aceste 3 rezultate, se pot rezolva toate cele 3 cerințe, confrom următoarelor formule.

Cerința 1

Deoarece o cifră este fie numarata de cnt_1 fie de cnt_2, numărul de cifre ale numărului N este:

$$cif_1 + cif_2$$

Cerința 2

Deoarece o cifră numarata de cnt_1 poate fi modificată într-un singur mod, iar o cifră numarată de cnt_2 poate fi modificată în 2 moduri, numărul de numere aproape de N este:

$$cif_1 + 2 \cdot cif_2$$

Cerința 3

Pentru a calcula cât mai ușor răspunsul pentru cerința 3 vom împărți numerele de numere aproape de numărul N în 3 categori:

- Numerele care nu diferă prin nicio cifră față de numarul N. Singurul număr din această categorie este chiar N.
- Numerele care diferă prin exact o cifră față de numărul N. Există $aproape_2$ numere de acest fel, deoarece dacă modificăm o cifră de 2 ori și nu ne întroarcem tot la numărul N atunci am decrementat cu 1 aceeași cifră a numărului de 2 ori.
- Numerele care diferă prin exact două cifre față de numărul N. În această categorie există 3 cazuri de perechi de cifre:
  - Ambele cifre sunt numărate de cnt_1. În acest caz există o singură modalitate de a modifica numărul N și există $\frac{cnt_1 \cdot (cnt_1 - 1)}{2}$ astfel de perechi.
  - O cifră este numarată de cnt_1, iar cealaltă este numarată de cnt_2. În acest caz există 2 modalități de a modifica numărul N și există $cnt_1 \cdot cnt_2$ astfel de perechi.
  - Ambele cifre sunt numărate de cnt_1. În acest caz există 4 modalități de a modifica numărul N și există $\frac{cnt_2 \cdot (cnt_2 - 1)}{2}$ astfel de perechi.

Adunand numărul de numere din fiecare categorie, numărul de numere aproape de cel puțin un număr aproape de N este:

$$1 + aproape_2 + \frac{cnt_1 \cdot (cnt_1 - 1)}{2} + 2 \cdot cnt_1 \cdot cnt_2 + 4 \cdot \frac{cnt_2 \cdot (cnt_2 - 1)}{2}$$

2 Problema Cochilie

Propunător: Szabo Zoltan, Liceul Tehnologic "Petru Maior" Reghin / ISJ Mureș - Tg.Mureș

Observăm că dimensiunile cochiliei reprezintă numere Fibonacci, valoarea lor fiind în functie de paritatea lui N.

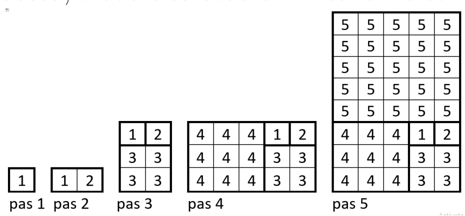
Pentru cerința 1:

Dimensiunea cochiliei în funcție de N:

1. dacă N este impar, avem $FIB(N+1)$ linii și $FIB(N)$ coloane
2. dacă N este par, avem $FIB(N)$ linii și $FIB(N+1)$ coloane

Pentru cerința 2:

Observăm, că multe linii ale cochiliei vor fi identice. Întrucât pătratele adăugate, în funcție de N, pot avea pasul final SUS, JOS, STANGA sau DREAPTA, de aceea, trebuie să studiem 4 cazuri distincte în funcție de N modulo 4.



Observăm, că linia cu construcția cea mai simplă va conține una sau două valori:

1. Dacă $N = 4k + 1$,

prima linie (primele $FIB(N)$ linii) va/vor conține o singură valoare N de $FIB(N)$ ori;

2. Dacă $N = 4k + 2$,

prima linie (primele $FIB(N-1)$ linii) va/vor conține două valori, și anume $N-1$ de $FIB(N-1)$ ori, urmate de N de $FIB(N)$ ori;

3. Dacă $N = 4k + 3$,

ultima linie (ultimele $FIB(N)$ linii) va/vor conține o singură valoare N de $FIB(N)$ ori;

2. Dacă $N = 4k$,

ultima linie (ultimele $FIB(N-1)$ linii) va/vor conține două valori, și anume N de $FIB(N)$ ori, urmate de $N-1$ de $FIB(N-1)$ ori;

Avem următoarele configurații distincte de linii:

- Pentru N par avem $N/2$ configurații distincte
- Pentru N impar avem $(N+1)/2$ configurații distincte

De exemplu, pentru $N = 5$ avem $(5+1)/2 = 3$ configurații distincte de linii, și anume:

5 5 5 5 5 (de 5 ori) (de $FIB(5)$ ori)

4 4 4 1 2 (o dată) (de $FIB(1)$ ori)

4 4 4 3 3 (de 2 ori) (de $FIB(3)$ ori)

Observăm, că pentru diferitele valori ale lui N modulo 4, avem aceeași regulă, dar cu valori diferite de pornire. În funcție de cele 4 cazuri menționate mai sus, putem porni de prima sau ultima linie, cu 1 sau două valori numere naturale.

Vom construi alternativ câte o linie de sus respectiv de jos, până ce realizăm configuratia tuturor liniilor.

nr	nr aparitii linie	valori
1	$FIB(9)$	9
2	$FIB(5)$	8,5,6
3	$FIB(1)$	8,4,1,2,6
4	$FIB(3)$	8,4,3,6
5	$FIB(7)$	8, 7

Pentru cazul $N = 4k + 1$, vom avea L=(N+1)/2 linii. Modul de construire este următorul:

Construim un tablou pentru numărul de apariții, cu valoarea de pornire NRAP(1)=N, și un tablou bidimensional V pentru valori, cu valoarea de pornire V(1)=[N].

Linia următoare pe care o construim va fi ultima linie: NRAP(L)=N-2, V(L)=[N-1,N-2].

Linia următoare pe care o construim va fi linia 2: NRAP(2)=N-4, V(2)=[N-1,N-4,N-3].

Linia următoare pe care o construim va fi penultima linie: NRAP(L-1)=N-6, V(L)=[N-1,N-5,N-6,N-3].

Linia următoare pe care o construim va fi linia 4: NRAP(2)=N-8, V(3)=[N-1,N-5,N-8,N-7,N-3].

șamd...

Algoritmul continuă, până când toate liniile se vor construi.

Regula de construcție a elementelor este următoarea: Pentru numărul de apariții Descreștem valoarea anterioară cu 2 de la un pas la altul. Pentru a construi elementele șirului de valori V[p] în functie de linia precedenta, ne folosim de formula recursivă a șirului lui Fibonacci, F(N)=F(N-1)+F(N-1). Având valorile șirului curent, depistăm cea mai mică valoare K din șir și o vom înlocui cu valorile K-1, K-2, scriindu-le în ordine crescătoare, dacă ne aflăm pe primele L/2 linii, și in ordine descrescătoare dacă ne aflăm pe ultimele L/2 linii.

Fiecare element x luat de FIB(x) ori, va reprezenta o linie a cochiliei.

Pentru a calcula continutul liniei cu nr de ordine K, vom scadea pe rand numarul de linii FIB(N), FIB(n-4), ..., până când se va ajunge la linia dorita.

Se vor prelucra cu atentie toate cele 4 cazuri al lui N ($4k$, $4k + 1$, $4k + 2$, $4k + 3$)

3 Problema Logic

Propunător: Alin Burța, Colegiul Național "B.P. Hasdeu", Buzău

Cerința 1, complexitate $O(2^N)$:

Simulăm comportamentul circuitului în functie de șirul de biți aflat la intrare (avem un șir cu 2^N biți).

Pornim de pe linia N a circuitului:

- primii doi biți reprezintă cele două intrări ale primului circuit de pe linia N, următorii doi biți reprezintă cele două intrări ale celui de-al doilea circuit de pe linia N etc;
- pentru fiecare circuit de pe linia N calculăm valoarea obținută prin aplicarea operației SI ori SAU asupra biților de la intrare, iar în final vom avea un șir de 2^{N-1} biți. Sirul de biți obținut reprezintă intrarea în circuitele de pe linia $N-1$.

Aplicăm procedeul descris până ce ajungem pe linia 1, unde se găsește un singur circuit, iar șirul de biți de la intrarea în acesta are lungimea 2.

Cerința 2 - soluția 1, complexitate $O(2^N)$:

Fie: $T[i][j][k]$ = numărul de intrări pentru care la ieșirea din circuitul de pe linia i și coloana j se obține rezultatul k, unde:
$(k = 0, 1)$
$i = 1..N$
$j = 1..2^{N-1}$

- Dacă $i = N$ (ultimul nivel)
  Dacă circuitul (N,j) este SI atunci:
  $T[N][j][1] = 1$ (e o singura combinație: 11)
  $T[N][j][0] = 3$ (sunt 3 combinații: $00, 01, 11$)
  Daca circuitul (N, j) este SAU atunci:
  $T[N][j][1] = 3$ (sunt 3 combinații: $01, 10, 11$)
  $T[N][j][0] = 1$ (e o singura combinație: 00)
- Daca $i \leq N-1$ știm că, pentru oricare circuit aflat pe linia i și coloana j, intrările acestuia sunt ieșirile circuitelor de pe linia $i+1$ și coloana $2j-1$, respectiv linia $i+1$ și coloana $2j$.
  - Cazul 1 : (i, j) este circuit SI:
    $T[i][j][1] = T[i+1][2j-1][1] * T[i+1][2j][1]$
    (obțin 1 la ieșire doar dacă la intrare am 1 si 1)
    $T[i][j][0] = T[i+1][2j-1][0] * T[i+1][2j][0] + T[i+1][2j-1][0] * T[i+1][2j][1] + T[i+1][2j-1][1] * T[i+1][2j][0]$ adică:
    = în câte moduri obțin 0 în circuitul $(i+1, 2j-1)*$
    în câte moduri obțin 0 în circuitul $(i+1, 2j)$
    + în câte moduri obțin 0 în circuitul $(i+1, 2j-1)*$
    în câte moduri obțin 1 în circuitul $(i+1, 2j)$
    + în câte moduri obțin 1 în circuitul $(i+1, 2j-1)*$
    în câte moduri obțin 0 în circuitul $(i+1, 2j)$

- Cazul 2 : (i, j) este circuit SAU:
  T[i][j][0] = T[i+1][2j-1][0] * T[i+1][2j][0]
  (obtin 0 la ieșire doar daca la intrare am 0 si 0)
  $T[i][j][1] = T[i+1][2j-1][1] * T[i+1][2j][1] + T[i+1][2j-1][0] * T[i+1][2j][1] + T[i+1][2j-1][1] * T[i+1][2j][0]$ adică:
  = în câte moduri obțin 1 în circuitul $(i+1, 2j-1)*$
  în câte moduri obțin 1 în circuitul $(i+1, 2j)$
  + în câte moduri obțin 0 în circuitul $(i+1, 2j-1)*$
  în câte moduri obțin 1 în circuitul $(i+1, 2j)$
  + în câte moduri obțin 1 în circuitul $(i+1, 2j-1)*$
  în câte moduri obțin 0 în circuitul $(i+1, 2j)$

Rezultatul se obține în $T[1][1][k], k = 0$ sau 1, după caz.

Cerința 2 - soluția 2, complexitate $O(2^{2^N})$:

Soluția aceasta obține doar 11 puncte din cele 70.

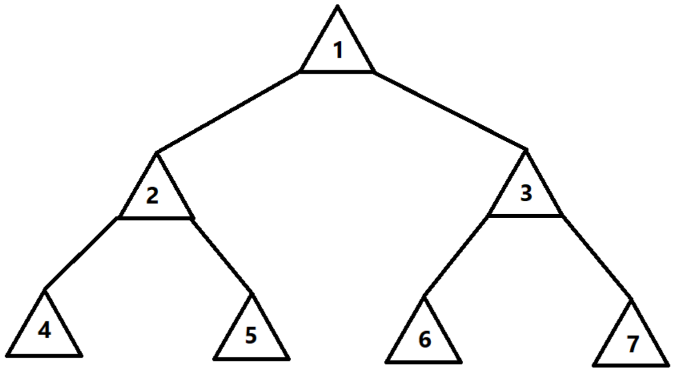
- Generăm printr-un procedeu oarecare, pe rând, toate șirurile de 2^N biți, care ar putea fi introduse la intrarea în circuit.
- Pentru fiecare șir de biți calculăm ce valoare se va obține la ieșire, folosind algoritmul descris la cerința 1 și contorizăm ca soluție dacă valoarea obținută este valoarea cerută.

Metoda va funcționa doar pe circuite cu nivele puține, $N \leq 4$.

4 Soluție alternativă

Bogdan-Ioan Popa, Universitatea din București, Facultatea de Matematică și Informatică

Vom numerota circuitele logice simple cu numerele $1, 2, 3, ...$ de la stânga la dreapta, de sus în jos, așa cum se vede în exemplul de mai jos:



Se observă că pentru un circuit logic simplu numerotat cu i, el este legat cu circuitele $2*i$ și $2*i+1$, situate pe nivelul următor. Se definește subcircuitul i ca fiind circuitul format din circuitul logic simplu numerotat cu i reunit cu subcircuitele $2*i$ și $2*i+1$. Fie $op[i]$ = operația asociată circuitului logic simplu numerotat cu i. Fie L = numărul total de circuite logice simple din CLP.

Cerința 1, complexitate $O(2^N)$:

Fie S un șir binar de lungime 2^N pentru care vom dori să calculăm valoarea la ieșirea din circuit. Fie $E[i]$ = valoarea evaluată la ieșirea din subcircuitul i. Vom face următoarea inițializare $E[L+i] = S[i]$ pentru fiecare i de la 1 la 2^N. Mai departe, pentru fiecare i de la L la 1 vom calcula $E[i]$:

- Dacă $op[i] = \&$, atunci $E[i] = E[2*i] \& E[2*i+1]$
- Dacă $op[i] = |$, atunci $E[i] = E[2*i] | E[2*i+1]$

Răspunsul se găsește în $E[1]$.

Cerința 2, complexitate $O(2^N)$:

Fie $cnt[i][res]$ = numărul de intrări de lungime corespunzătoare subcircuitului i, care evaluate de către subcircuitul i se obține rezultatul res (res poate fi doar 0 sau 1). Pentru fiecare i de la 1 la 2^N se inițializează $cnt[L+i][0] = cnt[L+i][1] = 1$ (Putem să ne imaginăm ca fiecare subcircuit fictiv $L+i$ este de fapt un singur bit). Mai departe, pentru fiecare i de la L la 1 vom calcula $cnt[i][res]$. Pentru fiecare le între 0 și 1 și fiecare ri între 0 și 1 vom face prelucrările de mai jos:

- Dacă $op[i] = \&$, atunci $cnt[i][le \& ri]$ se adună cu $cnt[2*i][le]*cnt[2*i+1][ri]$.
- Dacă $op[i] = |$, atunci $cnt[i][le|ri]$ se adună cu $cnt[2*i][le]*cnt[2*i+1][ri]$.

Expresia $cnt[2*i][le]*cnt[2*i+1][ri]$ reiese din următorul fapt: Pentru fiecare intrare oferită subcircuitului $2*i$ ce întoarce rezultatul le ($cnt[2*i][le]$ astfel de intrări) avem $cnt[2*i+1][ri]$ intrări oferite subcircuitului $2*i+1$ pentru care acesta întoarce ri.

Nu uităm ca la fiecare pas să facem operațiile modulo 666013. Răspunsul se găsește în $cnt[1][res]$, unde res este 0 sau 1, după caz.

Echipa

Setul de probleme pentru această rundă a fost pregătit de:

- student Andrei Arhire, Universitatea ”Alexandru Ioan Cuza” din Iași, Facultatea de Informatică.
- profesor Alin Burța, Colegiul Național ”B.P. Hasdeu”, Buzău.
- studentă Mihaela Cișmaru, Universitatea Politehnica București, Facultatea de automatică și calculatoare.
- software engineer Vlad Gavrilă, Google, Zurich.
- student Ștefan Manolache, University of Oxford, Lady Margaret Hall College.
- profesor Adrian Panaete, Colegiul Național ”A.T.Laurian”, Botoșani.
- student Alexandru Petrescu, University of Oxford, Keble College.
- student Bogdan-Ioan Popa, Universitatea din București, Facultatea de Matematică și Informatică.
- profesor Zoltan Szabo, Liceul Tehnologic ”Petru Maior” Reghin / ISJ Mureș - Tg.Mureș.
- student Cezar Trișcă-Vicol, University of Oxford, Christ Church College.
- student Theodor-Gabriel Tulbă-Lecu, Universitatea Politehnica București, Facultatea de automatică și calculatoare.