Descrieri ale soluțiilor problemelor de la secțiunea gimnaziu

Ciulini (clasele 5-6)

Pentru **cerința 1** putem simula ordinea de ridicare prin două variabile care își modifică alternativ valoarea. Una va păstra poziția la care ne aflăm în dreapta (și va crește cu 1 de câte ori vârtejul ajunge acolo), iar cealaltă va păstra poziția din stânga (și va scădea cu 1 de câte ori vârtejul ajunge acolo).
Această simulare este utilă și pentru a obține o parte a punctelor pentru **cerința 2**.
Totuși, în cazul cerinței 2, dacă valoarea lui n este mare, acest algoritm nu se va încadra în timp.
Observăm că pentru fiecare poziție din șir contează de fapt distanța față de mijloc și paritatea lui n. Astfel, putem face calculele doar analizând cazuri cu ajutorul unor if-uri.

Codul propus rezolvă problema „Ciulini” utilizând un algoritm eficient bazat pe calcule aritmetice simple. Programul citește datele din fișierul ciulini.in și scrie rezultatul în ciulini.out.

Explicația implementării

1. **Citirea datelor de intrare**
    - Se citește un număr c care determină tipul cerinței:
        - c = 1: Se cere ordinea în care sunt ridicați ciulinii.
        - c = 2: Se cere poziția la care va fi ridicat un ciulin dat.
    - Se citește n, numărul total de ciulini.
2. **Determinarea poziției inițiale a ridicării**
    - Dacă n este par, mijlocul este n/2.
    - Dacă n este impar, mijlocul este (n/2) + 1.

Cerința 1: Afișarea ordinii ridicării

- Se inițializează două variabile:
    - st = mij (pentru ridicarea ciulinilor din stânga).
    - dr = mij + 1 (pentru ridicarea ciulinilor din dreapta).
- Se parcurg alternativ pozițiile din dreapta și din stânga și se scriu în fișier.
- Dacă n este impar, ultimul ciulin rămas (cel mai din stânga) este afișat separat.

Exemplu:
Intrare:
1
7

Execuție:

- mij = 4
- Se extrag în ordinea: 4, 5, 3, 6, 2, 7, 1

Ieșire:
4 5 3 6 2 7 1

Cerința 2: Determinarea poziției unui ciulin ridicat

- Se citește k, poziția ciulinului pentru care trebuie determinată ordinea ridicării.
- Se compară k cu mij pentru a vedea dacă se află în dreapta sau în stânga:
  - Dacă k > mij, atunci este pe dreapta, iar poziția sa de ridicare este 2 * (k - mij).
  - Dacă k <= mij, atunci este pe stânga, iar poziția de ridicare este 2 * (mij - k) + 1.

Exemplu:
Intrare:
2
8 5
Execuție:

- mij = 4
- 5 este în dreapta lui 4, deci poziția sa este 2 * (5 - 4) = 2.

Ieșire:
2

Complexitate

- **Cerința 1:** O(n), deoarece se parcurge o singură dată secvența de ciulini.
- **Cerința 2:** O(1), deoarece poziția unui ciulin se calculează direct prin formule aritmetice.

Acest algoritm este foarte eficient și poate rula rapid chiar și pentru valori mari ale lui n.

Șir (clasele 5-6)

Șirul dat este format din grupuri complete, primul grup fiind format din 1, al doilea din 1 și din 2, al treilea din 1, 2, 3. In general grupul g este format din valorile 1, 2, …, g si are deci g elemente.
Dacă adunăm valorile ce reprezintă numărul de elemente din fiecare grup obținem o sumă gauss, deci deducem că numărul grupurilor din care fac parte valorile date p și q sunt valori de ordrinul radical(p sau q).
Primul pas în rezolvarea problemei este să determinăm grupurile din care fac parte cele două valori date.
Acest lucru se poate face calculând o sumă gauss până când ajungem să includem elementul pentru care facem calculul (p sau q). Se poate obține grupul și un calcul direct (folosind rădăcini ale unei ecuații de grad ||), acest lucru nu era necesar pentru obținerea punctajului maxim (și în plus ar fi mai greoi de implementat).
Odată ce am determinat grupurile din care fac parte gp și gq, avem mai multe cazuri de analizat

- Grupurile complete cuprinse între gp+1 și gq-1. Valorile de la 1 la gp+1 apar de același număr de ori în aceste grupuri. Valoarea gp+2 are număr de apariții cu 1 mai mic, valoarea gp+3 are număr de apariții cu încă 1 mai mic etc. Putem contoriza aceste apariții folosind un vector de frecvență.
- Grupurile incomplete gp respectiv gq. În funcție de pozițiile pe care se află valorile p și u în aceste grupuri, avem de incrementat în vectorii de frecvență valorile pentru anumite elemente.

Tratarea tuturor cazurilor solicită atenția la implementare.

Prieteni (clasele 7-8)

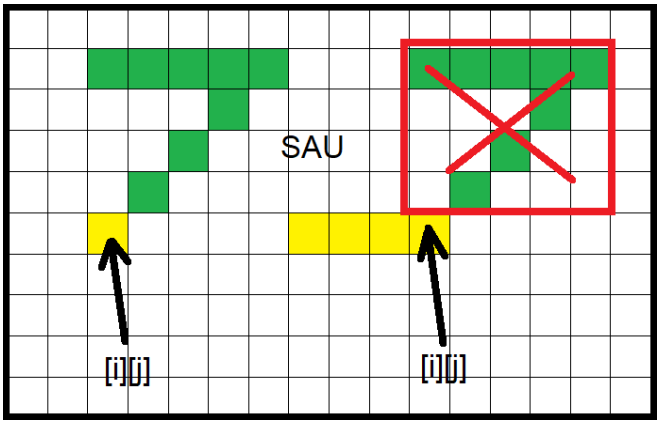
Se observă că lungimea secvenței Y nu poate fi decât diferența de lungimi dintre cele două lungimi ale șirurilor date, A și B (o notăm L).
Mai departe, o primă abordare ar fi să simulăm operația descrisă, adică să luăm fiecare secvență de lungime L din A, să o ștergem și să o încercăm apoi în fiecare poziție pe șirul rezultat. Se obține timp de calcul de ordin n^3.
Putem reduce timpul la n^2 cu următoarea observație:
Fie i cel mai mic indice unde A[i] diferă de B[i]. Dacă secvența Y a fost aplicată intersectând elemente din secvența X, atunci ea nu putea fi aplicată decât la poziția i. Similar, dacă nu a fost aplicată intersectând secvența X, atunci a fost aplicată intersectând secvența Z. În acest caz determinăm determinăm unica poziție unde s-ar fi aplicat Y plecând simultan de la finalul șirurilor A și B până găsim valori diferite în cele două.
Analizând cele două cazuri, observă că avem doar două variante de a încerca unde aplicăm secvența Y, reducând astfel complexitatea cu o dimensiune.

Zorro (clasele 7-8)

Problema se poate rezolva cu abordări specifice algoritmilor de sume parțiale și subsecvență de sumă maximă.

Soluție cu timp de calcul de ordin n^3
Putem fixa linia de sus și linia de jos a unui Z. Apoi, fixăm poziția cea mai din dreapta ce pe linia de sus. Poziția corespunzătoare de pe linia de jos este acum unic determinată.
Pentru poziția determinată sus ne interesează subsecvența de sumă maximă formată din cel puțin două elemente, care se află pe linia de sus și se termină la poziția fixată de noi.
Pentru poziția determinată jos, ne interesează o subsecvență de sumă maximă determinată similar, dar către dreapra.
Suma de pe zona diagonală a Z-ului o putem determina dacă precalculăm pentru diagonale un șir de sume parțiale (SD[i][j] = suma tuturor elementelor care se află pe aceeași diagonală cu i,j și care se află mai la dreapta sau sus - calcul care se poate face și la citire asfel: SD[i][j] = SD[i-1][j+1] + a[i][j]).

Soluție cu timp de calcul de ordin n^2.
Am amintit și la soluția anterioară de subsecvența de sumă maximă. Pe un vector, pentru a determina această valoare la poziția i (subsecvența de sumă maximă terminată la poziția i) avem S[i] = maxim(S[i-1] + v[i], v[i]).
Astfel de sume vom calcula noi pentru fiecare linie a matricei, atât către stânga cât și către dreapta.
În plus, ideea de la algoritmul de subsecvență de sumă maximă o vom mai folosi o dată astfel: La un moment dat tinem in D[i][j] suma maximă de la o subsecvență ce formează un Z incomplet (fără latura de jos).



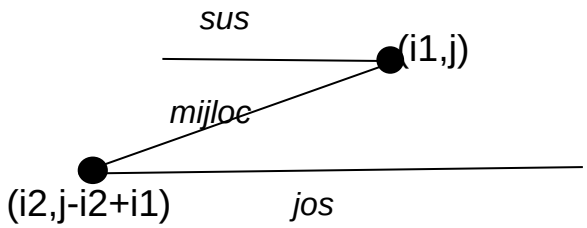
În figura anterioară am reprezentat cu verde un astfel de Z. Când coborâm pe diagonală pentru a calcula valoarea următoare, fie adăugăm elementul următor la Z, extinzându-l (ca în stânga) fie repornim un nou Z dacă acest lucru este convenabil.

Soluție O(n^3)

Folosim 3 matrice corespunzătoare celor 3 laturi ale literei Z (latura de sus, latura de jos si cea din mijloc) cu următoarea semnificație:

- `sus[i][j]` = lungimea subsecvenței de sumă maximă care conține cel puțin două elemente, **parcursă pe linia i de la stânga la dreapta** și se termină cu elementul `a[i][j]`
- `jos[i][j]` = lungimea subsecvenței de sumă maximă care conține cel puțin două elemente, **parcursă pe linia i de la dreapta la stânga** și se termină cu elementul a[i][j]
- în matricea mijloc se reține sumele parțiale pe fiecare linie paralelă cu diagonala secundară începând de sus în jos

Fixăm două linii i1 si i2 (i1<i2) și formăm toate literele Z având un vârf pe poziția `(i1, j)` și al doilea vârf pe poziția `(i2 , j-i2+i1)`. Folosind valorile din cele 3 matrice putem afla care sunt laturile cu suma maximă având cele două vârfuri: `sus[i1][j], jos[i2][j-i2+i1],` respectiv `mijloc[i2-1][j-i2+i1+1]-mijloc[i1][j]`



Soluție O(n^2)

Folosim tot 3 matrice corespunzătoare celor 3 laturi, fiecare matrice încercând să prelungească subsecventa de suma maximă pe latura sa, astfel:

- `sus[i][j]` = lungimea subsecvenței de sumă maximă care conține cel puțin două elemente, parcursă pe linia i de la stânga la dreapta și se termină cu elementul `a[i][j]`
- `mijloc[i][j]` = lungimea subsecvenței de sumă maximă care prelungește litera Z cu latura din mijloc și se termină cu elementul `a[i][j]`; latura se poate lega fie direct de o linie orizontală, fie să prelungească o linie oblică, alegându-se varianta cea mai bună
- `jos[i][j]` = lungimea subsecvenței de sumă maximă care prelungește litera Z cu o linie orizontală și se termină cu elementul `a[i][j]`; latura se poate lega fie direct de o linie oblică, fie să prelungească o linie orizontală.