Algoritmi care lucrează cu tablouri bidimensionale

Consultații pentru concursul Mate-Info UBB 2026

stud. Rareș-Andrei Cotoi

November 22, 2025

Facultatea de
Matematică și Informatică
UBB Cluj-Napoca

Cuprins

Tablouri bidimensionale

1 Recapitulare - Need to know

Un tablou **bidimensional** este o structură de date cu *două dimensiuni*: una corespunzătoare **liniilor** și alta **coloanelor**.

Fiecare element este accesibil folosind doi indici: unul pentru linia pe care se află și altul pentru coloana pe care elementul se află.

De exemplu, dacă considerăm $mat = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \Rightarrow mat[1][2] = 2$ și $mat[3][1] = 7$

(indexăm începând cu poziția 1).

Așadar, dimensiunea $n \times m$ a unei matrice este determinată de 2 variabile: numărul de linii (n) și numărul de coloane (m). În cazul în care $n = m$, spunem că matricea este **pătratică**.

Matrici pătratice

1 Recapitulare - Need to know

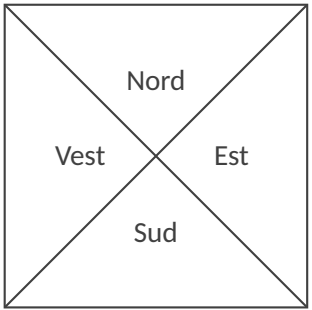
Datorită faptului că numărul de linii din matrice este identic cu numărul coloanelor, matricile pătratice au câteva proprietăți particulare (față de matricile uzuale).

Diagonala principală: Într-o matrice pătratică, diagonala principală este determinată de toate elementele pentru care indiciele liniei este egal cu indicele coloanei, adică $i = j \; \forall \; i = \overline{1,n}, j = \overline{1,n}$. De exemplu, pentru matricea $mat = \begin{pmatrix} \boxed{1} & 2 & 3 \\ 4 & \boxed{5} & 6 \\ 7 & 8 & \boxed{9} \end{pmatrix}$, elementele de pe diagonala principală sunt $mat[1][1] = 1, mat[2][2] = 5$ și $mat[3][3] = 9$.

Diagonala secundară: Într-o matrice pătratică, diagonala secundară este determinată de toate elementele pentru care suma celor 2 indici este $n + 1$, adică $i + j = n + 1 \; \forall \; i = \overline{1, n}, j = \overline{1, n}$. De exemplu, pentru matricea $mat = \begin{pmatrix} 1 & 2 & \boxed{3} \\ 4 & \boxed{5} & 6 \\ \boxed{7} & 8 & 9 \end{pmatrix}$, elementele de pe diagonala principală sunt $mat[1][3] = 3, mat[2][2] = 5$ și $mat[3][1] = 7$.

Existența celor 2 diagonale determină în matrice 4 zone diferite: *Nord*, *Sud*, *Est* și *Vest*.

Zonele matricei pătratice



- **Nord**: Elemente situate deasupra diagonalei principale și a diagonalei secundare, unde $i < j$ și $i + j < n + 1$.
- **Sud**: Elemente situate sub diagonala principală și sub diagonala secundară, unde $i > j$ și $i + j > n + 1$.
- **Vest**: Elemente situate sub diagonala secundară, dar deasupra diagonalei principale, unde $i > j$ și $i + j < n + 1$.
- **Est**: Elemente situate deasupra diagonalei secundare, dar sub diagonala principală, unde $i < j$ și $i + j > n + 1$.

Cuprins

Sume parțiale

2 Recapitulare - Nice to know

Considerăm o matrice de dimensiune $7x7$, $mat =$ $\begin{pmatrix} 1 & 2 & 3 & 2 & 1 & -1 & 5 \\ 9 & 3 & 3 & 2 & 1 & -3 & 9 \\ 2 & 5 & 4 & -1 & 1 & -1 & 7 \\ 6 & 2 & 5 & -2 & 3 & -5 & 6 \\ 4 & 2 & 6 & 6 & 2 & -1 & 5 \\ -1 & 2 & 3 & 2 & 1 & -6 & 4 \\ -1 & -2 & 5 & 2 & 2 & -1 & 5 \end{pmatrix}$.

Cum putem afla suma elementelor din chenarul marcat cu roșu? Dar a celor din chenarul marcat cu verde? Există o soluție mai eficientă pentru determinarea acestor sume?

Sume parțiale

Pentru fiecare set de sume calculate, soluția brută (parcurgerea element cu element) are complexitatea $O(n^2)$. Totuși, suma oricărui chenar delimitat de două colțuri $(stSus, drJos)$ poate fi determinat mai eficient, astfel:

- Într-o matrice auxiliară s, păstrăm la poziția $s[i][j]$ suma elementelor din submatricea cu colțul stânga sus la coordonatele $(1, 1)$ și colțul dreapta jos la coordonatele (i, j). La citirea unui element $mat[i][j]$, vom avea

$$s[i][j] = \begin{cases} 0, \text{ dacă } i = 0 \text{ sau } j = 0; \\ s[i-1][j] + s[i][j-1] - s[i-1][j-1] + mat[i][j], \text{ altfel} \end{cases}$$

- Apoi, suma elementelor dintr-o submatrice cu colțul stânga sus la coordonatele $(st1, dr1)$ și colțul dreapta jos la coordonatele $(st2, dr2)$ va fi $s[st2][dr2] - s[st1-1][dr2] - s[st2][dr1-1] + s[st1-1][dr1-1]$.

Pentru matricea anterioară $mat \Rightarrow S = \begin{pmatrix} 1 & 3 & 6 & 8 & 9 & 8 & 13 \\ 10 & 15 & 21 & 25 & 27 & 23 & 37 \\ 12 & 22 & 32 & 35 & 38 & 33 & 54 \\ 18 & 30 & 45 & 46 & 52 & 42 & 69 \\ 22 & 36 & 57 & 64 & 72 & 61 & 93 \\ 21 & 37 & 61 & 70 & 79 & 62 & 98 \\ 20 & 34 & 63 & 74 & 85 & 67 & 108 \end{pmatrix}$. Cum chenarul roșu era delimitat de colțul stânga sus $(2, 2)$ și colțul dreapta jos $(6, 6)$, suma elementelor din chenarul roșu este $suma = s[6][6] - s[1][6] - s[6][1] + s[1][1]$. Complexitatea timp pentru extragerea sumei este, în acest caz, $O(1)$.

Cuprins

3 Probleme

Problema 1

3 Probleme

16. Se consideră algoritmul `ceva(A, n, r, c, nr, x)`, unde n este număr natural ($1 < n \leq 10$), A este o matrice cu $n \times n$ elemente numere naturale ($A[1][1]$, $A[1][2]$, ..., $A[n][n]$), r, c, x sunt numere naturale ($1 \leq r, c, x \leq 10$), iar nr este număr întreg ($-10^3 \leq nr \leq 10^3$). Dacă $n = 4$ și inițial $A[3][2] = 5$ și $A[1][4] = 8$, care din secvențele de apeluri de mai jos va modifica elementele matricei A astfel încât $A[3][2]$ să fie egal cu 50 și $A[1][4]$ să fie egal cu 16?

```
Algorithm ceva(A, n, r, c, nr, x):
    If (r + x - 1 ≤ n) AND (c + x - 1 ≤ n) then
        For i ← r, r + x - 1 execute
            For j ← c, c + x - 1 execute
                A[i][j] ← A[i][j] * nr
            EndFor
        EndFor
    EndIf
EndAlgorithm
```

A.
```
ceva(A, n, 3, 2, 5, 1)
ceva(A, n, 2, 1, 4, 3)
ceva(A, n, 3, 3, 16, 2)
```

B.
```
ceva(A, n, 1, 3, 2, 2)
ceva(A, n, 3, 2, 2, 3)
ceva(A, n, 2, 1, 10, 3)
```

C.
```
ceva(A, n, 2, 3, 4, 2)
ceva(A, n, 1, 1, 2, 4)
ceva(A, n, 3, 1, 2, 1)
ceva(A, n, 2, 1, 5, 3)
```

D.
Niciuna din secvențele de apeluri nu produce modificarea precizată în enunț.

Sursa: Admitere UBB iulie 2025

Problema 1 - soluție

3 Probleme

Observăm că, la fiecare apel, algoritmul multiplică valoarea $A[i][j]$ cu parametrul nr, pentru fiecare $i = \overline{1, r + x - 1}, j = \overline{1, c + x - 1}$ **doar dacă se respectă condiția inițială, de la linia 2**.

Analizăm varianta A: după apelul `ceva(A, n, 3, 2, 5, 1)`: $A[3][2] = 25, A[1][4] = 8$; după apelul `ceva(A, n, 2, 1, 4, 3)`: $A[3][2] = 100, A[1][4] = 8$, deci varianta A nu poate fi corectă;

Varianta B: după apelul `ceva(A, n, 1, 3, 2, 2)`: $A[3][2] = 5, A[1][4] = 16$; după apelul `ceva(A, n, 2, 1, 4, 3)`: $A[3][2] = 5, A[1][4] = 16$ (**Atenție:** acest apel nu produce niciun efect, deoarece nu se respectă condiția de la linia 2); după apelul `ceva(A, n, 2, 1, 4, 3)`: $A[3][2] = 50, A[1][4] = 16$, deci varianta B este corectă.

Problema 1 - soluție

3 Probleme

Varianta C: după apelul `ceva(A, n, 2, 3, 4, 2)`: $A[3][2] = 5, A[1][4] = 8$ (apelul nu afectează elementele de interes pentru noi); după apelul `ceva(A, n, 1, 1, 2, 4)`: $A[3][2] = 10, A[1][4] = 16$; după apelul `ceva(A, n, 3, 1, 2, 1)`: $A[3][2] = 10, A[1][4] = 16$ (apelul nu afectează elementele de interes pentru noi); după apelul `ceva(A, n, 2, 1, 5, 3)`: $A[3][2] = 50, A[1][4] = 16$, deci varianta C este corectă.

Evident, varianta D nu se aplică, deci răspunsul final este **BC**.

Problema 2

3 Probleme

2. Se consideră algoritmul `creareTablou(n, m, x)`, unde n, m sunt numere naturale ($1 \leq n, m \leq 100$), iar x este un tablou bidimensional cu $n * m$ elemente numere întregi ($x[1][1]$, $x[1][2]$, ..., $x[n][m]$, $0 \leq x[i][j] \leq 10^4$, pentru $i = 1, 2, ..., n$; $j = 1, 2, ..., m$).

```
Algorithm creareTablou(n, m, x):
    k ← 0
    For i ← 1, n execute
        For j ← 1, m execute
            If k MOD 2 ≠ 0 then
                x[i][j] ← k * k
            EndIf
            Write x[i][j], " "
            k ← k + 1
        EndFor
        Write new line
    EndFor
EndAlgorithm
```

Ce afișează acest algoritm dacă elementele tabloului x sunt inițializate cu 0?

A. Algoritmul afișează elementele tabloului bidimensional x, în care se află valori egale cu 0 și primele ($n * m$) **DIV** 2 pătrate perfecte impare.

B. Algoritmul afișează elementele tabloului bidimensional x, în care se află valori egale cu 0 și primele pătrate perfecte pare.

C. Algoritmul afișează elementele tabloului bidimensional x, în care se află șirul primelor ($n * m$) **DIV** 2 pătrate perfecte pare.

D. Algoritmul afișează elementele tabloului bidimensional x, în care – dacă am așeza elementele linie după linie – pătratele perfecte impare ar apărea în ordine crescătoare, eventual precedate și/sau urmate de valori egale cu 0.

Sursa: Admitere UBB septembrie 2023

Problema 2 - soluție

3 Probleme

Elementele care la finalul execuției algoritmului nu vor fi 0 vor fi cu siguranță pătrate perfecte impare, lucru impus de condiția `If` k `MOD` $2 \neq 0$. Cum k este incrementat cu 1 la fiecare element parcurs, condiția menționată anterior va fi îndeplinită din 2 în 2 pași, astfel că, la finalul execuției, vom avea jumătate din elementele matricei egale cu 0, și jumătate din elemente pătrate perfecte de numere impare. De aici, variantele B și C se exclud automat. Varianta A este adevărată, iar varianta D este, de asemenea, adevărată, deoarece pătratele perfecte sunt plasate în matrice în ordine crescătoare (k începând de la 0 și fiind incrementat la fiecare pas). Așadar, răspunsul corect este **AD**.

Problema 3

3 Probleme

16. Se consideră algoritmul `buildMatrix(n, x, m, y)`, unde n și m sunt numere naturale ($1 \le n, m \le 100$), x și y sunt vectori cu n, respectiv m elemente numere întregi ($x[1], x[2], ..., x[n]$, unde $-10^3 \le x[i] \le 10^3$, pentru $i = 1, 2, ..., n$ și $y[1], y[2], ..., y[m]$, unde $-10^3 \le y[j] \le 10^3$, pentru $j = 1, 2, ..., m$). Algoritmul `max(a, b)` returnează valoarea maximă dintre numerele a și b. Algoritmul `zeros(n, m)` returnează o matrice cu n linii și m coloane, cu toate elementele egale cu 0.

```
Algorithm buildMatrix(n, x, m, y):
    A ← zeros(n + 1, m + 1)
    For i ← 1, n execute
        For j ← 1, m execute
            If x[i] = y[j] then
                A[i + 1][j + 1] ← A[i][j] + 1
            Else
                A[i + 1][j + 1] ← max(A[i][j + 1], A[i + 1][j])
            EndIf
        EndFor
    EndFor
    Return A[n + 1][m + 1]
EndAlgorithm
```

Care dintre următoarele afirmații sunt adevărate?

A. În urma apelului `buildMatrix(3, [3,2,3], 3, [2,3,3])`, se returnează valoarea 2.

B. Algoritmul `buildMatrix(n, x, m, y)` returnează valoarea 0 dacă și numai dacă x și y au aceeași lungime și nu au nici un element comun.

C. Dacă $n = m$, iar elementele vectorului y reprezintă o permutare a elementelor vectorului x, atunci algoritmul `buildMatrix(n, x, m, y)` va returna valoarea n.

D. Există date de intrare pentru care valoarea returnată de algoritm este egală cu $n + m$.

Sursa: Admitere UBB septembrie 2025

Problema 3 - soluție

3 Probleme

Prin calcul, varianta A este adevărată. Observăm că impactul algoritmului este de a modifica valoarea curentă $A[i+1][j+1]$ cu $A[i][j]+1$, dacă întâlnim valori egale la pozițiile i, j în cei doi vectori, sau cu maximul dintre $A[i][j+1]$ și $A[i+1][j]$, în caz contrar. Varianta B este falsă, deoarece algoritmul returnează 0 și dacă șirurile **nu** au aceeași lungime, dar nu au niciun element comun. Varianta A este un contra exemplu pentru varianta C, iar varianta D este falsă, deoarece, chiar dacă am avea același vector (caz care produce valoarea maximă returnată), se va returna valoarea $min(n, m)$. Răspunsul corect este, astfel, **A**.

Notă: Acest algoritm este cunoscut și ca **metoda de determinare a celui mai lung subșir comun (LCS), folosind metoda Programării dinamice**.

Problema 4

3 Probleme

4. Cristian a primit un seif în care tatăl lui îi ascunde o jucărie. Seiful are un cod inițial setat de tatăl său, reprezentat de o matrice 3×3 și 9 butoane, câte unul pe fiecare element al matricei. Apăsând pe butonul de pe linia i și coloana j, Cristian observă că se adună 1 la valoarea fiecărui element care se află pe linia i sau pe coloana j. Toate adunările se fac modulo 4, astfel încât dacă adunăm la 3 o unitate obținem 0. Cristian reușește să deschidă seiful dacă găsește o secvență de pași, un pas însemnând apăsarea unui singur buton și actualizarea elementelor matricei, și ajunge la o matrice cu toate elementele cu valoarea 0. Când Cristian este cuminte, tatăl lui îi setează un cod inițial din care se poate ajunge la matricea nulă. Spre exemplu pornind de la matricea cu codul $\begin{pmatrix} 2 & 1 & 3 \\ 1 & 1 & 0 \\ 3 & 2 & 3 \end{pmatrix}$, Cristian poate ajunge la matricea $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ realizând pașii $\begin{pmatrix} 2 & 1 & \boxed{3} \\ 1 & 1 & 0 \\ 3 & 2 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 2 & 0 \\ 1 & \boxed{1} & 1 \\ 3 & 2 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 3 & 0 \\ 2 & \boxed{2} & 2 \\ 3 & 3 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 0 & 0 \\ \boxed{3} & 3 & 3 \\ 3 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, unde $\boxed{\ }$ marchează butonul apăsat la fiecare pas. Când Cristian nu este cuminte, tatăl lui setează un cod inițial din care nu se poate ajunge la matricea nulă, cum este cel dat de matricea $\begin{pmatrix} 0 & 0 & 3 \\ 2 & 2 & 3 \\ 2 & 2 & 1 \end{pmatrix}$. Pentru câte dintre codurile inițiale $\begin{pmatrix} 3 & 1 & 0 \\ 1 & 2 & 2 \\ 2 & 3 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 & 2 \\ 3 & 3 & 1 \\ 1 & 3 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \\ 2 & 3 & 1 \end{pmatrix}$ Cristian poate găsi jucăria?

A) 1 B) 2 C) 3 D) 4

Sursa: Admitere Universitatea București iulie 2024

Problema 4 - soluție

3 Probleme

Considerăm suma elementelor $\texttt{mod}\ 4$ de pe linia i ca fiind $sl[i]$ și suma elementelor $\texttt{mod}\ 4$ de pe coloana j ca fiind $sc[j]$. Pentru exemplul dat, avem $sl = [2, 2, 0]$ și $sc = [2, 0, 2]$.

O observație esențială este faptul că dacă putem ajunge de la una dintre matrici la O_3 prin apăsări de butoane, trebuie să putem ajunge și de la O_3 la matricea inițială, tot prin apăsarea de butoane, în ordine inversă. De asemenea, observăm că, la apăsarea unui buton a_{ij}, efectul produs este:

- $sl[i] = (sl[i] + 3)\ \texttt{MOD}\ 4$;
- $sc[i] = (sc[i] + 3)\ \texttt{MOD}\ 4$;
- La toate celelalte sume de pe linii și coloane se va adăuga 1.

Problema 4 - soluție

3 Probleme

Dacă începem să construim o matrice pornind de la O_3, inițial, toate sumele de pe linii și coloane vor fi 0. Cum la fiecare pas adăugăm, pentru fiecare sumă, 3 sau 1, iar operațiile se efectuează `mod` 4, deducem faptul că, pe tot parcursul construcției, **sumele vor avea mereu aceeași paritate**. Așadar, concluzionăm că jucăria poate fi găsită de Cristian doar dacă sumele de pe liniile și coloanele matricei inițiale au aceeași paritate.

Dintre matricile date, singura care respectă condiția identificată este $\begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix}$, deci răspunsul corect este **A**.

Algoritmi care lucrează cu tablouri bidimensionale

Mulțumesc pentru atenție!