Proiect de lecție / unitate de învățare

Unitatea de învăţământ: Colegiul Național Petru Rareș Suceava
Disciplina: Informatică
Clasa: a XI-a intensiv informatică
Data[1]**:** _____
Profesor: Tiberiu Socaciu
Unitatea de învățare: Recursivitate
Tema lecției: Recursivitate. Contain Virus
Tipul lecției: practică, de laborator
Scopul lecției: sistematizare și consolidare abilități și deprinderi practice
Competențe generale:

a) **CG1:** Elaborarea algoritmilor de rezolvare a problemelor
b) **CG2:** Implementarea algoritmilor într-un limbaj de programare

Competențe specifice:

a) **CS1:** Analiza problemei în scopul identificării metodei de programare adecvate pentru rezolvarea problemei
b) **CS2:** Aplicarea creativă a metodelor de programare pentru rezolvarea unor probleme intradisciplinare sau interdisciplinare sau a unor probleme cu aplicatibilitate practică
c) **CS3:** Utilizarea tehnicilor moderne în implementarea aplicațiillor

Obiective operaţionale[2]**.** La finalul lecţiei, elevul poate:

a) **O1:** să demonstreaze posibilitatea de a folosi corect noțiunile prezentate
b) **O2:** să aplice noțiunile prezentate într-un caz concret
c) **O3 :** să analizeze relațiile dintre funcțiile programului
d) **O4:** să descrie în limbaj științific mecanismele de interdependență dintre modulele programului
e) **O5:** să formuleze propria opțiune pentru o decizie optimă

Strategii didactice:

a) **Mijloace de învățământ:** calculatoarele din laboratoarele de informatică, directorul FTP_Elevi -> Materiale_Profesori prezent pe toate cele 30 stații de lucru, tableta grafică precum și resursele incluse în prezentul plan de lecție[3] respectiv:

Link pentru enunțul și tema lecției	TBC[4]
Link pentru mind map-ul lecției[5]	TBC[6]
Link pentru produsul final al lecției[7]	TBC[8]
Link și cod QR pentru activitatea independentă[9] și pentru tema acasă a elevilor	TBC[10]
Generator de numere aleatoare	https://g.co/kgs/s3eGCQ4

b) **Metode de învățământ:** expunerea, algoritmizarea, demonstrația, modelarea, problematizarea, conversația euristică, descoperirea sau brainstormingul.
c) **Forme de activitate:** individuală, frontală, lucrul pe grupe pentru orele de laborator, lucrul pe grupe mici, lucrul în perechi.
d) **Metode de evaluare:** evaluare orală, evaluare scrisă, observarea sistematică a elevilor sau prin teste și/sau fișe de evaluare.

Bibliografie:

a) cursul *Make Technology your Friend*
b) site-ul LeetCode, www.leetcode.ro
c) Eusebiu Toader, Tiberiu Socaciu, Pregătirea interviului de angajare ca programator C++. 33 teme de interviu rezolvate, MatrixRom, 2023,

---

[1] Se va completa, dupa caz
[2] rezultă din derivarea competențelor specifice
[3] link-ul lecței va fi disponibil elevilor și înafara orelor de curs pentru ca aceştia să poată accesa oricând materialele folosite
[4] Se va complete ulterior
[5] plan de lecție alternativ, necesar pentru susținerea activităților A și B din ”dirijarea învățării”
[6] Se va complete ulterior
[7] activitatea C
[8] Se va complete ulterior
[9] în clasă
[10] Se va complete ulterior

https://www.matrixrom.ro/produs/pregatirea-interviului-de-angajare-ca-programator-c-33-teme-de-interviu-rezolvate/.

Succesiunea activităţilorde instruire

Nr. crt.	Evenimentul didactic	Durata (min.)	Obiective operaţionale	Activitatea de instruire		Strategia didactică			
				Activitatea profesorului	Activitatea elevilor	Mijloace de învăţământ	Metode şi procedee	Forme de organizare	Forme de evaluare
1.	**Moment organizatoric Prezentarea temei abordate şi a obiectivelor lecţiei Captarea atenţiei**	2		Salutul, efectuarea prezenței, verificarea funcționării rețelei de calculatoare, a FTP-ului. Prezentarea temei și a obiectivelor lecției.		AdServio	Conversația euristică	Frontal	Observarea sistematică a elevilor
2.	**Dirijarea învăţării**	10	***O1***	Activitatea A	Notează lecția pe caiet sau calculator, adresează întrebări	Videoproiector	Învățare activă, prin cooperare	Frontal	
		10	***O2***	Activitatea B	Notează lecția pe caiet sau calculator, adresează întrebări	Videoproiector	Învățare activă, prin cooperare	Individual Frontal	Observarea sistematica a elevilor
		10	***O3***	Activitatea C	Notează lecția pe caiet sau calculator, adresează întrebări	Videoproiector	Învățare activă, prin cooperare	Frontal	
3	**Transferul cunostinţelor/ consolidarea cunoştinţelor.**	10	O4	Problematizare legată de relația dintre funcțiile programului. Se invită un elev la tablă pentru a realiza un modul necesar rezolvării problemei	Implementează funcția pe calculator	Rețeaua de calculatoare Videoproiector	Algoritmizarea	Individual Frontal	Orala
4	**Fixarea cunoştinţelor şi asigurarea feedback-ului**	7	O5				Problematizare a	Frontal	Scrisa
5	**Tema pentru acasă**	1	O1, O2, O3, O4, O5	Solicită elevilor să completeze ca temă pentru acasă o problema de pe LeetCode, aleasa aleator, impreuna Genereaza sarcina de lucru pentru acasa in Classroom	Noteaza tema de casa pe caiet au calculator	Generator de numere aleatoare Google Classrom			Scrisa

Enunt

Un virus se răspândește rapid, iar sarcina ta este să carantinezi zona infectată prin instalarea pereților.

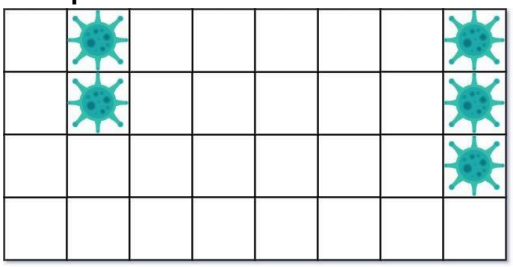
Lumea este modelată ca o grilă binară m x n esteinfectată, unde esteinfectată[i][j] == 0 reprezintă celule neinfectate și esteinfectată[i][j] == 1 reprezintă celulele contaminate cu virusul. Un perete (și un singur perete) poate fi instalat între oricare două celule adiacente cu 4 direcții, la limita comună.

În fiecare noapte, virusul se răspândește în toate celulele vecine în toate cele patru direcții, cu excepția cazului în care este blocat de un perete. Resursele sunt limitate.

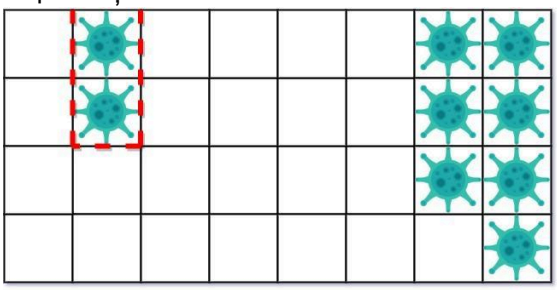
În fiecare zi, puteți instala pereți în jurul unei singure regiuni (adică zona afectată (bloc continuu de celule infectate) care amenință celulele cele mai neinfectate în noaptea următoare). Nu va exista niciodată o cravată.

Returnați numărul de pereți folosiți pentru carantinarea tuturor regiunilor infectate. Dacă lumea se va infecta pe deplin, întoarceți numărul de pereți folosiți.

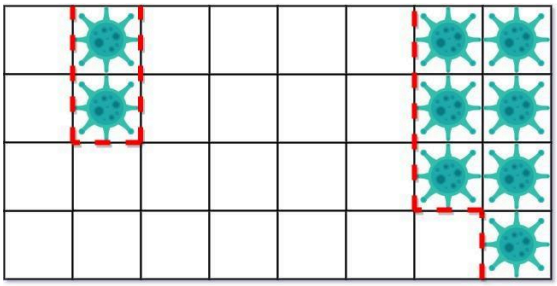
Exemplu 1:



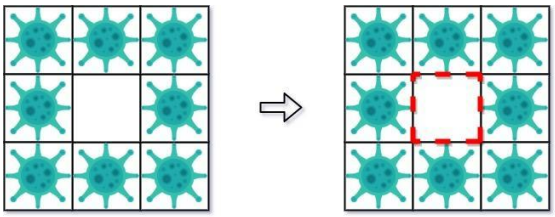
Intrare: este infectat = [[0,1,0,0,0,0,0,1],[0,1,0,0,0,0,0,1],[0,0,0,0,0,0,0,1],[0,0,0,0,0,0,0,0]]
ieșire: 10
Explicație: Există 2 regiuni contaminate.
În prima zi, adăugați 5 pereți pentru a carantina regiunea virală din stânga. Placa după ce virusul se răspândește este:



În a doua zi, adăugați 5 pereți pentru a carantina regiunea virală din dreapta. Virusul este complet ținut sub control.



Exemplu 2:



Intrare: este infectat = [[1,1,1],[1,0,1],[1,1,1]]
ieșire: 4
Explicație: Chiar dacă există o singură celulă salvată, există 4 pereți construiți.
Observați că pereții sunt construiți numai pe limita comună a două celule diferite.

Exemplu 3:
Intrare: este infectat = [[1,1,1,0,0,0,0,0,0],[1,0,1,0,1,1,1,1,1],[1,1,1,0,0,0,0,0,0]]
ieșire: 13
Explicație: Regiunea din stânga construiește doar două ziduri noi.

Constrângeri:

- m == este infectată.lungimea
- n == este infectată[i].lungimea
- 1 <= m, n <= 50
- Este infectată[i][j] este fie 0 sau 1.
- Există întotdeauna o regiune virală contiguă pe tot parcursul procesului descris, care va **infectați strict mai multe pătrate necontaminate** în runda următoare.

Rezolvae

```
# define MAXM      50
# define MAXN      50
# define MAXPERETI 20
# include <fstream>
# include <iostream>
using namespace std;

struct celula { int i, j, infectat, culoare; };
struct perete { celula celula1, celula2; };
celula grid[MAXM][MAXN];
int m, n;
perete lista_de_pereti[MAXPERETI];
int numar_pereti;

int exista_punct_necolorat_dar_virusat(int &i, int &j)
{
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    if (grid[i][j].infectat and grid[i][j].culoare == 0)
      return 1;
return 0;
}

void fill(int i, int j, int culoare)
{
if (i < 0 or j < 0 or i >= m or j >= n) return;
if (grid[i][j].infectat and grid[i][j].culoare == 0)
  {
  grid[i][j].culoare = culoare;
  fill(i - 1, j, culoare);
  fill(i + 1, j, culoare);
  fill(i, j - 1, culoare);
  fill(i, j + 1, culoare);
  }
```

```
}

void sterge_culorile()
{
for (int i = 0; i < m; i++)
  for (int j = 0; j < n; j++)
    grid[i][j].culoare = 0;
}

int numara_regiunile_infectate()
{
int culoare = 0;
int i, j;
sterge_culorile();
while (exista_punct_necolorat_dar_virusat(i, j))
  {
  culoare++;
  fill(i, j, culoare);
  }
return culoare;
}

int izoleaza_regiunea(int numar_regiune)
{
int numar_garduri_puse = 0;
// pun garduri si numar gardurile
// pun celulele ca neinfectate
// virusul din celulele celelalte se intinde
// atentie, virusul nu trece de garduri DEJA existente!!
return numar_garduri_puse;
}

void afiseaza_grid()
{
for (int i = 0; i < m; i++)
  {
  cout << "[ ";
  for (int j = 0; j < n; j++)
    cout << grid[i][j].infectat << " ";
  cout << "]" << endl;
  }
}

int rezolva_problema()
{
int minimul = -1; // e santinela, caci solutia >= 0
int numar_de_regiuni = numara_regiunile_infectate();
if (numar_de_regiuni == 0)
  return 0;
for (int i = 1; i <= numar_de_regiuni; i++)
  {
  celula grid_backup[MAXM][MAXN];
  perete lista_de_pereti_backup[MAXPERETI];
  int numar_pereti_backup;
  // salvez gridul si lista de garduri inainte de simulare
  numar_pereti_backup = numar_pereti;
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
      grid_backup[i][j] = grid[i][j];
  for (int i = 0; i < m; i++)
    lista_de_pereti_backup[i] = lista_de_pereti[i];
```

```
  // de facut gard in jurul acestei regiuni
  // de calculat cat gard folosesc
  int numar_garduri_puse = izoleaza_regiunea(i);
  // de apelat recursiv rezolvarea problemei
  // si obtinut cat gard folosesc
  int rezultat_subproblema = rezolva_problema();
  // de calculat suma celor doua
  int solutia_cazului = numar_garduri_puse
    + rezultat_subproblema;
  // minimul dintre cea mai buna solutie de pana acum
  // si solutia curenta
  if ( (minimul == -1) or (minimul != -1 and
    solutia_cazului < minimul) )
    minimul = solutia_cazului;
  // restaurez gridul si lista de garduri dupa simulare
  numar_pereti = numar_pereti_backup;
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
      grid[i][j] = grid_backup[i][j];
  for (int i = 0; i < m; i++)
    lista_de_pereti[i] = lista_de_pereti_backup[i];
  }
return minimul;
}

int main()
{
ifstream fin("matrici.in");
while (fin >> m >> n)
  {
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
      {
      grid[i][j].i       = i;
      grid[i][j].j       = j;
      grid[i][j].culoare = 0;
      fin >> grid[i][j].infectat;
      }
  cout << "Raspunsul la problema: " << endl;
  afiseaza_grid();
  int numar_culori = numara_regiunile_infectate();
  numar_pereti = 0;
  int raspunsul_la_problema = rezolva_problema();
  cout << "este " << numar_pereti << " pereti de construit.";
  cout << endl << endl;
  }
}
```