COLEGIUL NAȚIONAL "GRIGORE MOISIL"
B-dul Timişoara nr. 33, Sector 6, Bucureşti
Tel: 021 413 26 96; 021 413 26 47. Fax: 021 440 10 06
Website: http://www.moisil.ro

ROMÂNIA
MINISTERUL
EDUCAȚIEI ȘI
CERCETĂRII

PROIECT
BACKTRACKING PE MATRICE

PROFESOR COORDONATOR:
Prof. Glaje Denisa

ELEVI:
Tiron Filip
Holban Vlad
Nicu Eduard
Largeanu Vlad
Cojocar Teodor

-Bucuresti 2026-

Tema proiectului:

Problema numarul 3: Sportivul

Cuprins:

Capitolul I: Descrierea temei

Pe un teren de dimensiune dreptunghiulara, cu denivelari, se afla un sporitv care doreste sa se antreneze pentru un concurs de alpinism. Cunoscand altitudinea fiecarei portiuni de teren, pozitia initiala a alpinistului, si pozitia finala in care doreste sa ajunga, sa se determine un traseu pentru care suma diferentelor de altitudine intre pozitiile consecutive de pe traseu sa fie maxima. Alpinistul nu are voie coboare.

Capitolul II: Descrierea limbajului de programare utilizat

Limbajul de programare utilizat este C++, datorita eficientei si popularitatii acestuia, fiind ideal pentru rezolvarea problemelor care necesita multa performanta, cum este cazul algoritmilor de cautare.

Programele scrise in C++ se executa foarte repede, ceea ce este esential cand algoritmul de Backtracking trebuie sa verifice un numar mare de trasee posibile pe matrice.

Capitolul III: Cerintele sistemului

Pentru a rula programul, avem neovie de:

- **32KB RAM**

Aceasta cerinta este extrem de redusa si indica faptul ca programul este optimizat pentru a rula eficient chiar si pe sisteme cu resurse limitate.

Acest minim de 32KB RAM subliniaza faptul ca programul nu foloseste structuri de date masive sau biblioteci externe care ar incarca excesiv memoria.

Capitolul IV: Descrierea programului

- **Variabile:** **Incepem prin a declara variabilele (precum dimensiunile matricei, matricea propriu-zisa, vectorii de solutie, pozitia initiala, pozitia finala, vectorii de deplasare si alte variabile folosite pentru rezolvarea problemei) globale.**

```
int n, m, a[25][25], sol_i[1000], sol_j[1000], sol_len, s_max = 0;
int poz_init_I, poz_init_J;
int poz_fin_I, poz_fin_J;

int di[] = {-1, -1, -1, 0, 0, 1, 1, 1};
int dj[] = {-1, 0, 1, -1, 1, -1, 0, 1};
```

- **Functii:**

1. **Prima este o functie de citire; intai, citim dimensiunile matriceii (n, m), dupa aceea folosim o structura repetitiva pentru a parcurge elementele si citim matricea. Dupa, citim pozitia initiala si pozitia finala, folosite pentru rezolvarea problemei.**

```
void citire(){
    cout << "Introduceti dimensiunea matricei si matricea in sine: ";
    cin >> n >> m;
    for(int i = 1; i<=n; i++)
        for(int j = 1; j<=m; j++)
            cin >> a[i][j];
    cout << "Introduceti pozitia initiala: ";
    cin >> poz_init_I >> poz_init_J;
    cout << "Introduceti pozitia finala: ";
    cin >> poz_fin_I >> poz_fin_J;
}
```

2. **A doua functie din program este cea de tiparire. O folosim pentru a afisa pe ecran vectorul de solutii in format (x, y). Pentru a face asta, parcurgem elementele vectorului de solutii si le afisam. Afisam o paranteza, abscisa raspunusului, virgula, ordonata si o paranteza. Pentru a realiza insiruirea de solutii, afisam cate o virgula dupa fiecare solutie, cu exceptia ultimei solutii.**

```
void tiparire(){
    for(int i = 1; i<=sol_len; i++) {
        cout<<"("<<sol_fin_i[i]<<", "<<sol_fin_j[i]<<")";
        if(i!=sol_len) cout<<", ";
    }
    cout<<"\n";
}
```

3. **A treia functie este cea de backtracking. Aceasta functie executa rezolvarea propriu-zisa a problemei. In momentul in care exploram fiecare celula, o marcam in 2 vectori temporari, pentru a tine minte intregul traseu parcurs pana atunci. Daca ajungem la destinatie, stocam solutia in vectorii finali. Altfel, continuam sa parcurgem toate cele 8 celule din jurul celei initiale, cu conditia ca acestea sa nu coboare in altitudine. Cu ajutorul parametrilui suma copiem in vectorul final solutia corecta doar daca este cea corecta (suma diferentelor de altitudine intre oricare doua celule parcurse trebuie sa fie maxima).**

```
void back(int i, int j, int suma, int pas) {
    sol_i[pas] = i; sol_j[pas] = j;
    if(i==poz_fin_I && j==poz_fin_J) {
        if(suma > s_max) {
            s_max = suma;
            sol_len = pas;
            for(int x = 1; x<=sol_len; x++) {
                sol_fin_i[x] = sol_i[x];
                sol_fin_j[x] = sol_j[x];
            }
        }
    } else {
        for(int k = 0; k < 8; k++) {
            int vi = i + di[k];
            int vj = j + dj[k];
            if(1<=vi && vi<=n && 1<=vj && vj<=m) {
                if(a[vi][vj] >= a[i][j])
                    back(vi, vj, suma + a[vi][vj] - a[i][j], pas + 1);
            }
        }
    }
}
```

- **Main:** **Ultima parte a programului este functia main. In aceasta parte a programului, apelam functiile de citire, backtracking si afisare.**

```
int main() {
    citire();
    back(poz_init_I, poz_init_J, a[poz_init_I][poz_init_J], 1);
    tiparire();
    return 0;
}
```

- **Programul complet**

```cpp
#include <iostream>
using namespace std;

int n, m, a[25][25], sol_i[1000], sol_j[1000], sol_fin_i[1000], sol_fin_j[1000], sol_len, s_max = 0;
int poz_init_I, poz_init_J;
int poz_fin_I, poz_fin_J;

int di[] = {-1, -1, -1, 0, 0, 1, 1, 1};
int dj[] = {-1, 0, 1, -1, 1, -1, 0, 1};

void citire(){
    cout << "Introduceti dimensiunea matricei si matricea in sine: ";
    cin >> n >> m;
    for(int i = 1; i<=n; i++)
        for(int j = 1; j<=m; j++)
            cin >> a[i][j];
    cout << "Introduceti pozitia initiala: ";
    cin >> poz_init_I >> poz_init_J;
    cout << "Introduceti pozitia finala: ";
    cin >> poz_fin_I >> poz_fin_J;
}

void tiparire(){
    for(int i = 1; i<=sol_len; i++) {
        cout<<"("<<sol_fin_i[i]<<", "<<sol_fin_j[i]<<")";
        if(i!=sol_len) cout<<", ";
    }
    cout<<"\n";
}

void back(int i, int j, int suma, int pas) {
    sol_i[pas] = i; sol_j[pas] = j;
    if(i==poz_fin_I && j==poz_fin_J) {
        if(suma > s_max) {
            s_max = suma;
            sol_len = pas;
            for(int x = 1; x<=sol_len; x++) {
                sol_fin_i[x] = sol_i[x];
                sol_fin_j[x] = sol_j[x];
            }
        }
    } else {
        for(int k = 0; k < 8; k++) {
            int vi = i + di[k];
            int vj = j + dj[k];
            if(1<=vi && vi<=n && 1<=vj && vj<=m) {
                if(a[vi][vj] >= a[i][j])
                    back(vi, vj, suma + a[vi][vj] - a[i][j], pas + 1);
            }
        }
    }
}

int main() {
    citire();
    back(poz_init_I, poz_init_J, a[poz_init_I][poz_init_J], 1);
    tiparire();
    return 0;
}
```

Capitolul V: Concluzii

In concluzie, programul demonstreaza o aplicare concreta si eficienta a metodei Backtracking. Aceasta metoda este esentiala pentru explorarea sistematica a tuturor solutiilor posibile intr-un spatiu de cautare mare, asa cum este cazul traseului pe matrice

Totodata, algoritmul nu doar gaseste un drum de la pozitia initiala la cea finala, ci identifica traseul optim, cel pentru care suma diferentelor de altitudine intre pozitiile consecutive este maxima.

Capitolul VI: Bibliografie si webografie

- **www.pbinfo.ro**
- **Manual Informatica Clasa a 11-a**

Repartizarea Sarcinilor

Tiron Filip	-rezolvarea problemei -documentatie scrisa in GitHub -prezentare multimedia
Holban Vlad	-rezolvarea problemei (debugging si animarea solutiei) -documentatie
Nicu Eduard	-documentatie scrisa in Word -rezolvarea problemei (debugging)
Largeanu Vlad	-rezolvarea problemei -documentatie scrisa in GitHub -prezentarea multimedia
Cojocar Teodor	-prezentare multimedia -poster