Sejtautomaták

Tuhári Richárd

2018. május 6.

Tartalomjegyzék

1. Bevezető

1.0.1. Egy rövid leírás

A sejtautomatákat előszeretettel alkalmazzák a mikrostruktúrák modellezésére. A felépítése a következő:

Adott egy élettér leggyakrabban négyzetrács, de célszerű alkalmazni a méhkas szerű hatszögeken alapuló struktúrát is. Legyen akármilyen felosztás is, a rácsok által közrefogott cellákat sejteknek nevezzük. A sejtek felvehetnek diszkrét állapotokat, melyek változnak a diszkréten léptetett időben, általában a szomszédos cellák állapotától függően.

Magukat a sejtautomatákat Neumann János vezette be, hogy matematikai modellt alkosson a gépek önreprodukciójához. A cél annak következményeinek vizsgálata volt, ha létezne egy önreprodukáló gép. Erre a célra meg is alkotta a Konstruktor nevű modellt egy négyzetrácson, mely 29 féle állapotot képes felvetetni a sejtekkel.

1.0.2. Conway életjátéka

Egyik legismertebb közülük John Conway életjátéka. Az élettér itt egy sima négyzetrács, ahol minden sejtnek a szomszédai az ő körülötte elhelyezkedő 8 cellában elhelyezkedő sejtek. Kétféle állapot létezik, az élő és a halott. Működése a következőképp alakul:

- Megadott, n db szomszéd esetén nem változik a sejt állapota.
- n+1 db élő szomszéd esetén a sejt élő lesz, vagy az marad.
- Máskülönben a sejt elpusztul.

2. A modellek

2.1. Conway életjátéka

A program megírásasorán törekedtem az általános program megírására. Így az bemenetként a következőket tartalmazza:

```
cout<< " Value of n: ";
cin >> n;
cout<< " Size of niche: ";
cin >> m;
cout << " Number of periods: ";
cin >> T;
cout << "Choose the boundary condition: ";
cout << " Open(0 0 0),Periodic(1 0 0), Living(0 1 0), Random but stable(0 0 1)
cin >> per >> el >> vel;
```

Legfelül a feledatokban is használt n érték látható. m a négyzet alakú élettér szélének mérete, viszont abból 2 levonódik a határok kezelésére. Alatta T, az hogy meddig menjen a folyamat és végül a határfeltételek.

A jegyzőkönyv nem tartalmaz folyamatábrákat, mivel a program automatikusan kiírja azokat a terminálba. Amennyiben lassítani vagy gyorsyítani szeretnénk, azt csak programon belül tehetjük meg, a microseconds vátozó értékének átírásával. Javasolt fordítás: gcc -lstdc++ JConway.cpp A program és kimenete megtalálható a csatolmányok között és a függelékben is.

2.2. Homokdomb

A programbéli randomot használva látszik, hogy nem random a dolog, valószínűleg más seed kéne a kettőnek. Ez ki van kommentelve és a középre ejtett felett található. Mint a fentebbit, ezt is a program kimenetén ábrázolom mely megtalálható mind a függelékben, mind a csatolmányban.

3. Diszkusszió

A modellezett dolgokat az élet sok területén fel lehet használni. Előbbi (nyilván sokkal bonyolultabban) jó lehet például a rákos sejtek létrejöttének statisztikai elemzésére, illetve már meglévők megélésének vizsgálatára. Hogy ezt lássuk ismernünk kell, hogy a daganatok elsődleges forrása az ha a tesben a helyreállításért felelős, rossz "alaprajzzal" rendelkezik, ezáltal létrehozva a hibás egyedeket. Ezt jelölhetné valami más (esetünkben mondjuk nem X, hanem Y) és szívósabb is lenne (kisebb n például).

4. Függelék

4.1. Conway

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
#include <cstdlib>
#include <unistd.h>
#define NUM_LINES 2
using namespace std;
int i,j,k,l,t;
int random_variable;
int num_neighbour;
int n=2;
int m=15;
bool vel=0;
bool el=0;
bool per=0;
int T=100;
unsigned int microseconds=1000000;
int main(){

        cout<< " Value of n: ";
        cin >> n;
        cout<< " Size of niche: ";
        cin >> m;
        cout << " Number of periods: ";
        cin >> T;
        cout << "Choose the boundary condition: ";
        cout << " Open(0 0 0),Periodic(1 0 0), Living(0 1 0), Random but stable(0 0 1)
```

```
        cin >> per >> el >> vel;



        char terminal_clearline [4];
        char terminal_moveup [4];

        sprintf(terminal_clearline, "%c[2K", 0x1B);
        sprintf(terminal_moveup, "%c[1A", 0x1B);

//Kezd s
        int niche[m+1][m+1]={};
        int niche2[m+1][m+1]={};
        for(i=0;i<m+1;i++){
                for(j=0; j<m+1; j++){
                        random_variable = std::rand();
                        if(vel==0 && el==0){
                                if(random_variable % 2!=0 && i!=0 && j!=0 && i!=m && j
                                        niche[i][j]=1;
                                }
                        }
                        if(el==1 && vel==0){
                                if(random_variable % 2!=0 && i!=0 && j!=0 && i!=m && j
                                        niche[i][j]=1;
                                }
                                if(i==0 || j==0 || i==m || j==1){niche[i][j]=1;}
                        }
                        if(el==0 && vel==1){
                                if(random_variable % 2!=0){
                                        niche[i][j]=1;
                                }
                        }
                        //cout << niche[i][j] << ' ';
                }
                //cout << endl;
        }
//Folyamat

        for(t=0;t<T;t++){
                for(int i = 1; i < m; i++)
                        if (i > 10)
                            cout<<" "<<i;
                        else
                            cout<<"  "<<i;
                cout<<endl;
//  let
                for(i=1;i<m;i++){
                        for(j=1; j<m; j++){
```

```
                                num_neighbour=0;
                                for(k=(i-1);k<(i+2);k++){
                                        for(l=(j-1);l<(j+2);l++){
                                                num_neighbour+=niche[k][l];
                                        }
                                }

                                if(per==1){
                                        if(j==1){num_neighbour+=(niche[i][m-1]+niche[i
                                        if(i==1){num_neighbour+=(niche[m-1][j]+niche[m

                                        if(j==m-1){num_neighbour+=(niche[i][1]+niche[i
                                        if(i==m-1){num_neighbour+=(niche[1][j]+niche[1

                                }

                                num_neighbour-=niche[i][j];
                                if(num_neighbour==n){
                                        t+=0;
                                }
                                if(num_neighbour==n+1){
                                        niche2[i][j]=1;
                                }
                                if(num_neighbour<n || num_neighbour>(n+1)){
                                        niche2[i][j]=0;
                                }
                        }
                }

                for(i=1;i<m;i++){
                        for(j=1;j<m;j++){
                                niche[i][j]=niche2[i][j];
                        }
                }

//Rajz
                for(int i = 1; i < m; i++)
                {
                        cout<<(char)(i + 48);
                        for(int j = 1; j < m; j++)
                        if(niche[i][j]==1)
                                cout<<" x ";
                        else
                                cout<<" . ";

                        cout<<(char)(i + 48)<<endl;
                }
//Take your time
```

```
                usleep(microseconds);


//T rl s
                if(t!=T-1){
                        for (int sor=0;sor<m;sor++){
                        printf("%s", terminal_moveup);
                        printf("%s", terminal_clearline);
                        }
                }
        }

        return 0;
}
```

4.2. Homokdomb/dűne)

Dűne abban az eseben lesz, ha a programbéli randomot használjuk.

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
#include <cstdlib>
#include <unistd.h>
#define NUM_LINES 2
using namespace std;
int m=8;
int random_variable;
int random_variable2;
unsigned int microseconds=800000;
int t;
int slide;
int main(){
        char terminal_clearline [4];
        char terminal_moveup [4];

        sprintf(terminal_clearline, "%c[2K", 0x1B);
        sprintf(terminal_moveup, "%c[1A", 0x1B);


        int niche[m][m]={};



        for(t=0;t<200;t++){
                for(int i = 1; i < m; i++)
                        if (i > 10)
                            cout<<" "<<i;
```

```
        else
            cout<<"  "<<i;
cout<<endl;

        //random_variable = std::rand();
        //random_variable = random_variable % m;
        //random_variable2 = std::rand();
        //random_variable2 = m-random_variable % m;
        random_variable=4;
        random_variable2=4;
        niche[random_variable][random_variable2]+=1;
        slide=1;
        while(slide!=0){
                slide=2;
                for(int j=1;j<m;j++){
                        for(int k=1;k<m;k++){
                                if(niche[j][k]==4){
                                        niche[j][k]=0;
                                        niche[j+1][k]+=1;
                                        niche[j-1][k]+=1;
                                        niche[j][k+1]+=1;
                                        niche[j][k-1]+=1;
                                        slide=3 ;
                                }
                if(slide==2){slide=0;}
                        }
                }
        }

for(int i = 1; i < m; i++)
{
        cout<<(char)(i + 48);
        for(int j = 1; j < m; j++){
                cout << " " << niche[i][j]  << " ";}
                /*if(niche[i][j]==0 || niche[i][j]==1 || niche[i][j]==
                        if(niche[i][j]==0){cout<<" 0 ";}
                        if(niche[i][j]==1){cout<<" 1 ";}
                        if(niche[i][j]==2){cout<<" 2 ";}
                        if(niche[i][j]==3){cout<<" 3 ";}
                }
                else{cout<<" 0 ";}
        }*/
        cout<<(char)(i + 48)<<endl;
}

usleep(microseconds);

if(t!=199){
```

```
                        for (int sor=0;sor<m;sor++){
                        printf("%s", terminal_moveup);
                        printf("%s", terminal_clearline);
                        }
                }
        }

        return 0;
}
```

5. Hivatkozások

https://stegerjozsef.web.elte.hu/teaching/szamszim/index.php