Vilniaus Universitetas
Matematikos ir informatikos fakultetas

Jonas Antanaitis
Kompiuterinis modeliavimas, 2 grupė

Kurso Daugiamatės duomenų struktūros projektas

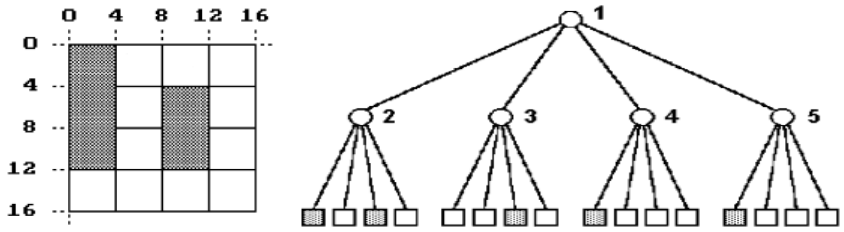
Atstumo žemėlapių, skirtų optimalaus kelio radimui, atvaizdavimas panaudojant ketvirtainius medžius

Vilnius, 2014

Šio projekto tikslas buvo panaudojant ketvirtainio medžio duomenų struktūras išspręsti optimalaus kelio radimo uždavinį. Jam įgyvendinti buvo remiamasi [1] pasiūlytais metodu, kuriame atstumų žemėlapiai atvaizduojami ketvirtainio medžio tiesine matrica. Tyriame buvo implementuoti šie algoritmai, bei sukurta grafinė vartotojo sąsaja.

Ketvirtainis medis. Jo atvaizdavimas.

Ketvirtainis medis buvo atvaizduojamas 4xL matrica $R = [P_0,P_1,...,P_L]$, kur L yra medžio mazgų skaičius, o $P_i=[Q_1, Q_2, Q_3, Q_4]$ atitinkamo mazgo vaikai. Jų reikšmės gali buti teigiamas skaičius (Nuorodą į žemesnio 1 lygiu medžio elementą), neigiamas skaičius (Laisva erdvė) ir 0 (Kliūtis).

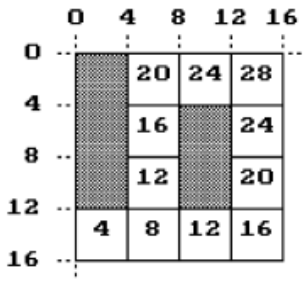


P_i/Q_{ij}	0	1	2	3
1	2	3	4	5
2	0	−1	0	−1
3	−1	−1	0	−1
4	0	−1	−1	−1
5	0	−1	−1	−1

1 pav. Plokštuma (viršuje kairėje), ją atitinkantis ketvirtainis medis (viršuje dešinėje) ir jo matrica (apačioje).

Atstumo žemėlapis

Ieškoti atstumams buvo naudojami atstumų žemėlapiai. Plokštumoje jie rodo atstumus nuo tam tikro taško (Pavyzdyje nuo (0,16) taško). Juos nesunku pritaikyti ketvirtainio medžio matricoje, atstumus vaizduojant neigiamais skaičiais. Didesnius laisvos erdvės plotus arba kliūtis galima atvaizduoti matricos elementu ir taip galima sutaupyti atminties. Pasirinkus tikslo koordinatę vienų ketvirtainio medžio ją



2 pav. Atstumų žemėlapis.

Pagalbinė matrica

Norint rasti kiekvieno medžio elemento atitinkančio kvadratinio bloko koordinates x, y ir jo kraštinę d, buvo naudojama pagalbinė matrica. Pagalbinė matrica sudaryta iš vienetų ir nulių. Pavyzdžiui x_0y_0 visada lygus 00, nes jie atitinką visą atstumų žemėlapio plotą, sekantis elementas atitinką vieno iš keturių (00, 01, 10, 11) jį dalinančių kvadratų numerį.

$$A = \begin{pmatrix} x_0y_0 \\ x_1y_1 \\ . \\ x_ny_n \end{pmatrix}$$

Kvadratinio bloko koordinatės ir kraštinės ilgis apskaičiuojamas iš formulių:

$$x_A = \sum_{i=0}^{n} x_i \times 2^{N-i-1},$$

$$y_A = \sum_{i=0}^{n} y_i \times 2^{N-i-1},$$

$$d_A = 2^{N-n-1}.$$

Projekte buvo naudojama klasė Blokai, kurios konstruktorius iš pagalbinės matricos sudarydavo ją atitinkantį bloko objektą ir su juo atlikdavo visas tarpusavio lyginimo operacijas.

Algoritmai

Visi projekte naudoti algoritmai veikia panašiu principu: pradedant nuo viršutinį medžio elementą atrinkančia pagalbinę matrica pereinama per visus keturis jos vaikus, atliekami lyginimo ar paieškos veiksmai, tenkinant sąlygą funkcija yra kviečiama dar kartą su papildyta pagalbine matrica. Tokiu principu buvo atliekami tokie veiksmai:

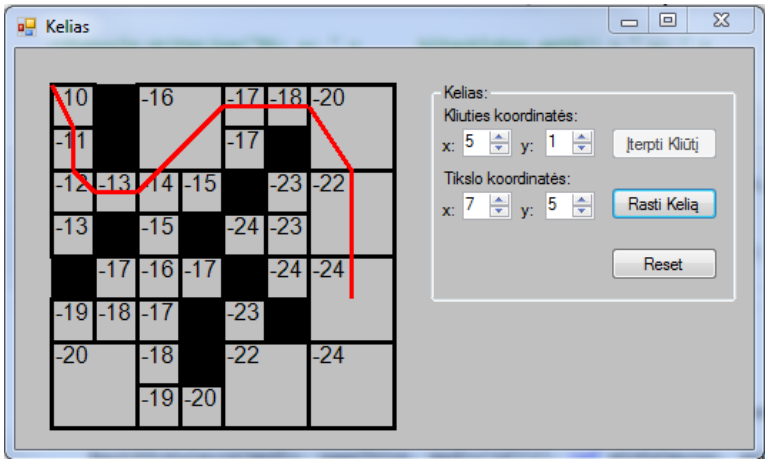
- Kliūties įterpimas į medį.
- Tikslo ir pradžios koordinačių pagalbinių matricų radimas.
- Atstumų užpildymas nuo pradinės koordinatės.
- Kaiminio bloko su minimalia reikšme radimas.
- Medžio piešimas.

1 Pavyzdys. Kliūties įterpimas į medį:

```
////////////////////////////////////////////////////////////////////////////////////////////////
//Medodas, skirtas įterpti kliutį į ketvirtainį medį:
static void Iterpimas(List<List<int>> medis, Kliutis kliutis, int id, List<List<int>> pagalbine) {

    for (int i = 0; i < 4; i++) {

        //Papildome pagalbinę matrica vienu iš 4 jos vaikų:
        if (i == 0) { pagalbine.Add( new List<int>(new int[] { 0, 0 }) ); }
        if (i == 1) { pagalbine.Add( new List<int>(new int[] { 1, 0 }) ); }
        if (i == 2) { pagalbine.Add( new List<int>(new int[] { 0, 1 }) ); }
        if (i == 3) { pagalbine.Add( new List<int>(new int[] { 1, 1 }) ); }

        //Sukuriamas naujas blokas:
        Blokas blokas = new Blokas(pagalbine);

        //Atliekami paieškos veiksmai:
        if (blokas.ArSutampa(kliutis)) {
            //Jeigu blokas sutampa su kliūtimi priliginame ji 0:
            medis[id][i] = 0;
        } else if (blokas.ArViduje(kliutis)) {
            //Jeigu kliūtis yra viduje bloko kviečiame šią f-ja su nauja pagalbine matrica:
            if (medis[id][i] != 0) {
                if (medis[id][i] < 0) {
                    medis[id][i] = medis.Count;
                    id = medis.Count;
                    medis.Add(new List<int>(new int[] { -1, -1, -1, -1 }));
                    Iterpimas(medis, kliutis, id, pagalbine);
                } else {
                    id = medis[id][i];
                    Iterpimas(medis, kliutis, id, pagalbine);
                }
            }
        }

        //Pašaliname pridėta pagalbinės matricos elementą:
        pagalbine.RemoveAt(pagalbine.Count - 1);
     }
}
```

GUI

Optimalaus kelio radimas yra vizualus uždavinys, todėl algoritmų testavimui ir naudojimui buvo sukurta grafinė vartotojo sąsaja. Ja galima įterpti į medį pasirinktas kliūtis, ir surasti kelia iki pasirinkto taško:



3 pav. Grafinė vartotojo sąsaja.

Literatūra:

1. "*Low-cost implementation of distance maps for path planning using matrix quadtrees and octrees",* Jozef Voros, Faculty of Electrical Engineering and Information Technology, Slovak Technical University, Ilkovicova 3, 812 19 Bratislava, Slovak Republic Accepted 29 June 2001