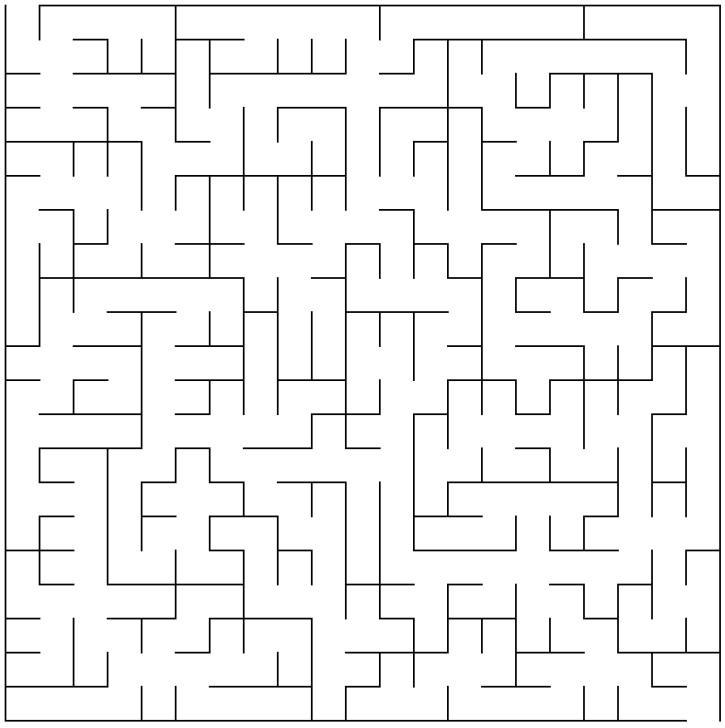
Gymnázium Praha 6, Arabská 14

předmět Programování, vyučující Jan Lána

Generování bludiště pomocí Aldous–Broderova algoritmu

Ročníkový projekt



OLIVER HURT, 1.E

KVĚTEN 2020

Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 2. 5. 2022 ……

1. Anotace

Cílem práce bylo napsat program v programovacím jazyce Java, který by použil Aldous–Broderův algoritmus pro vytvoření bludiště a následně ho vykreslil na obrazovku. Práce obsahuje stručný popis algoritmu, popis použitých tříd, vykreslování na obrazovku pomocí knihovny Swing a problémy na které jsem narazil po cestě.

The goal of this project was to write a program in the Java programming language that would generate a maze using Aldous–Broder algorithm and draw the maze on the screen. This paper includes a short description of the algorithm, description of classes that were used, how drawing was done using the java Swing library and problems i encountered along the way.

2. Zadání

Zadání mé ročníkové práce znělo následovně:

„Cílem práce bude generovat bludiště pomocí Aldous–Broderova algoritmu. Bludiště se bude skládat z ascii znaků "_", "|" a " ". Uživatel zadá šířku a výšku bludiště a na výstup ho program vygeneruje.“

3. Úvod

Téma – generovat bludiště pomocí Aldous–Broderova algoritmu – jsem si vybral, protože mi přišlo zajímavé a přínosné. Na práci jde navíc navazovat – např. implementace dalších algoritmů, měření rychlosti a porovnávání algoritmů, různé grafické zpracování apod. Na tomto ročníkovém projektu jsem se hodně naučil o programovacím jazyku Java i o vytváření GUI (Graphical User Interface). Cíl jsem splnil a s kódem tohoto projektu jsem spokojen.

Výsledná aplikace zobrazuje menu, ve kterém si navolíte velikost bludiště a kliknete na tlačítko *generovat*. Poté se vám v novém okně zobrazí dané bludiště. Projekt se skládá z třídy HlavniMenu – uvítací menu, třídy SecondFrame – samotné generování bludiště a jeho vykreslení a třídy Cell – objekt typu *cell* je daná buňka bludiště. Podrobněji je popis třídy Cell rozepsán níže.

4. Aldous–Broderův algoritmus

Aldous–Broderův algoritmus je algoritmus, určený pro generování bludišť. Všechny buňky mají stejnou pravděpodobnost navštívení. Bludiště nemá žádný sklon, proto tento algoritmus nemá žádnou tendenci[1]. Bludiště má vždy aspoň jedno řešení. Algoritmus předpokládá mřížku buněk. Každá buňka má všechny 4 stěny. Postupně se podle daných pravidel mažou stěny buněk, až se vytvoří dané bludiště. Kroky algoritmu jsou následující:

1. každá buňka má všechny 4 stěny
2. vyber jakoukoliv buňku v bludišti a označ ji jako navštívenou
   A. vyber náhodného souseda této buňky
   B. přesuň se na novou buňku
   C. pokud je již navštívená → krok A
   D. jinak: vymaž stěnu mezí předchozí a touhle buňkou
   E. označ ji jako navštívenou → krok A
3. opakujeme dokud nejsou navštívené všechny buňky v bludišti

Jak vidíte Aldous–Broderův algoritmus je jednoduché implementovat, zato je ale velmi neefektivní. Pokud se chcete dozvědět více, o tom jak algoritmus funguje doporučuji http://weblog.jamisbuck.org/2011/1/17/maze-generation-aldous-broder-algorithm

[1] Bárta, D. (n.d.). Aldous Broder. Retrieved May 1, 2022, from http://www.generatorbludist.cz/html/aldous_broder.html

5. Třída Cell

Třída Cell popisuje danou buňku v bludišti. Třída Cell.java vypadá následovně:

```
public class Cell {
    public static final int W = 20;//šířka jednoho čtverečku v px

    private int x, y;//souradnice dane bunky

    private boolean visited = false;

                                //top, right, left, down
    public boolean[] walls = {true, true, true, true};
    public Cell(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public boolean isVisited() {
        return visited;
    }

    public void removeWalls(Cell next) {
          //… metoda je popsána v kapitole Metoda removeWalls
    }

    public void draw(Graphics g){
         //… metoda je popsána v kapitole Metoda draw
    }
}
```

Jak můžete vidět, každá buňka zná svoje souřadnice x a y (bude samozřejmě použita v 2D poli), zda byla již navštívena a všechny své stěny. Dále třída obsahuje pomocné metody jako getX() a getY(), které jen vrátí danou hodnotu – např. souřadnici x; a konstruktor třídy.

6. Funkce MazeGen

Funkce MazeGen implementuje Aldous–Broderův algoritmus. Nepřijímá žádné argumenty a vrací 2D pole s datovým typem Cell.

```
public static Cell[][] mazeGen() {
    Cell[][] maze = new Cell[RADKY+1][SLOUPCE+1];
    for (int x = 0; x <= RADKY; x++) { //vygeneruj maze
        for (int y = 0; y <= SLOUPCE; y++) {
            maze[x][y] = new Cell(x, y);
        }
    }

    Cell current;
    Cell next;

    //gerovani nahode sourdnice pro prvni bunku
    Random rand = new Random();
    int upperbound = RADKY-1;//-1 protoze Random() zacina od 0

    //generování náhodného čísla
    int random_radek = rand.nextInt(upperbound);
    int random_sloupec = rand.nextInt(upperbound);

    current = maze[random_radek][random_sloupec]; //nahodna bunka
    current.setVisited(true);
    next = maze[random_radek][random_sloupec];

    int remaning = (RADKY+1)*(SLOUPCE+1)-1; //pocet zbyvajicich bunek
    int[]policko;
    int x = current.getX();
    int y = current.getY();
    int Nx = 0;
    int Ny = 0;

    while(remaning > 0){ //dokud neprojdeme celý maze
        x = current.getX();
        y = current.getY();
        policko = vyberRandSouseda(x, y); //vraci souradnice noveho souseda
        Nx = policko[0]; //nove souradnice od bunky na x, y
        Ny = policko[1];
        if( (Nx >= 0) && (Ny >= 0) && (Nx <= RADKY) && (Ny <= SLOUPCE)){
           //souradnice se vejdou do maze
            next = maze[Nx][Ny];
            if(next.isVisited() == false){
                current.removeWalls(next);
                remaning = remaning - 1;
                next.setVisited(true);
            }
            current = next;
            System.out.println(remaning);
        }
    }
    maze[0][0].walls[0] = false;//start a konec
    maze[RADKY][SLOUPCE].walls[3] = false;

    return maze;
}
```

Na začátku se naplní 2D pole *maze* objekty typu Cell. Poté se vyberou náhodná čísla x a y menší než hranice pole (aby jsme nevyskočili z pole). Do buňky *current* se poté kopíruje buňka na souřadnicích *x* a *y*. Naše buňka se označí jako navštívená a vypočítá se počet zbývajících buněk – šířka × výška bludiště − 1. Nastaví se základní proměnné a skočí se do cyklu while. Ten se opakuje dokud neprojdeme celé bludiště – nenavštívíme každou buňku.

První si uložíme souřadnice naší současné (*current*) buňky. Ty potom použijeme, abychom našli náhodného souseda. Souřadnice náhodného souseda jsou uloženy v poli *policko[]*. Poté zkontrolujeme, že nevystupujeme z bludiště. Pokud ne, přesuneme se na novou buňku (náhodný soused od *current* buňky). Pokud nova buňka *next* ještě nebyla navštívena, vymažeme stěnu mezi *current* a *next* a snížíme počet ještě nenavštívených buněk o 1. Poté se současná buňka *current* nastaví na buňku *next* a cyklus se opakuje dokud neprojdeme celé bludiště. Až se tak stane vytvoříme start a cíl bludiště a vrátíme ho.

7. Metoda RemoveWalls

Metoda removeWalls řeší jeden z hlavních problémů – jak vymazat odpovídající stěnu u buňek. Potřebujeme nějak rozeznat, u které buňky budeme mazat stěny pouze za pomoci souřadnic daných buňek. Metoda removeWalls() má následující prototyp – void removeWalls(Cell next).

Voláme ji pomocí dané buňky *current* a dostává jako argument buňku *next*. Známe tedy souřadnice obou buňek. K popisu této metody se vztahuje obrázek 1 a 2.

Řekněme, že začneme na buňce na souřadnicích 3,2 (zeleně) – tato buňka je už navštívená. Poté jsme náhodně vybrali jako buňku *next* buňku, která se nachází na souřadnicích 3,3. Jelikož ještě nebyla navštívena, zavolá se metoda removeWalls(). Metoda sama zjistí, kam jsme se posunuli vzhledem k buňce *current* (3,2) a to následujícím způsobem.

Odečteme hodnotu souřadnice x buňky next od hodnoty souřadnice x buňky current. V tomto případě je to 0 (3 − 3). To samé uděláme i se souřadnicemi *y*. V našem případě 2 − 3 = −1, tedy jdeme doprava. Vymažeme tedy odpovídající stěny u buněk, kterých se to týká. U buňky current pravá stěna a u buňky next levá stěna.

2D pole maze o velikosti 5x5

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

obrázek 1

výsledek zavolání metody

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

obrázek 2

Kód dané metody vypadá následovně.

```
public void removeWalls(Cell next) {
    int i = this.x - next.x;
    if(i == 1){ //jdu nahoru
        walls[0] = false; //horni stena pryc
        next.walls[3] = false; //dolni strana pryc
    }
    else if(i == -1){ //jdu dolu
        walls[3] = false; //spodni strana pryc
        next.walls[0] = false; //horni strana pryc
    }

    int j = this.y - next.y;
    if(j == -1){ // jdu doprava
        walls[1] = false; //prava stena pryc
        next.walls[2] = false; //leva stena pryc
    }
    else if(j == 1){//jdu doleva
        walls[2] = false; //leva strana pryc
        next.walls[1] = false; //prava strana pryc
    }
}
```

8. Funkce VyberRandSouseda

Funkce VyberRandSouseda má tento prototyp – int[] vyberRandSouseda(int x, int y).

Jako argument dostává souřadnice buňky a vrací pole o dvou prvcích – souřadnice x a y nového souseda.

```
public static int[] vyberRandSouseda(int x, int y) {
    Random ran = new Random();
    int smer = ran.nextInt(4);
    int Nx = x;
    int Ny = y;

    if(smer == 0){//nahoru
        Nx = x - 1;
    }
    else if(smer == 1){ //doprava
        Ny = y + 1;
    }
    else if(smer == 2) {//dolu
        Nx = x + 1;
    }
    else if(smer == 3){//doleva
        Ny = y - 1;
    }
    int[]pole = {Nx, Ny};
    return pole;
}
```

Funkce si vygeneruje náhodné číslo v rozmezí 1 až 4. Každé číslo označuje jeden směr. Funkce nastavuje svoje nové x a y (*Nx, Ny)* podle našich daných souřadnic, protože se vždy souřadnice změní pouze o 1.

Po tom co se vybere směr, se souřadnice upraví podle toho, kterým směrem se potřebujeme posunout. Např. jsem na souřadnicích 1,2 a hledám nového souseda. Náhodně se vylosovalo číslo 3. Jdu tedy doleva. Nové souřadnice tedy budou 1,1. Vratíme v poli *[1, 1]*.

9. Metoda draw

Metoda draw byla převzata z git repositáře – https://github.com/jaalsh/java-maze-algorithms. V tento moment už algoritmus pro generaci bludiště doběhl. Každá buňka v poli *maze* zná své stěny. Pokud má buňka danou stěnu, nakreslí se daná čára na obrazovku.

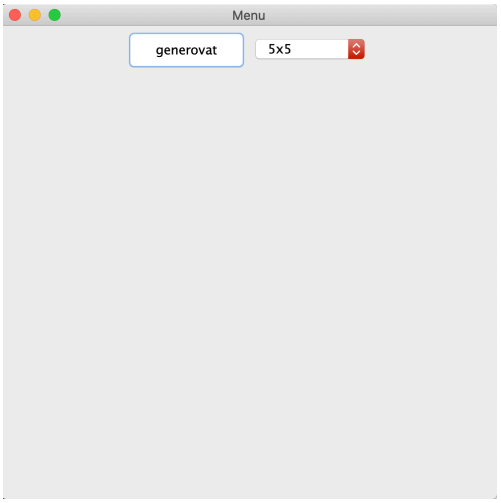
```
public void draw(Graphics g){
    int x2 = this.y * W;
    int y2 = this.x * W;

    g.setColor(Color.BLACK);
    if (walls[0]) {//horní
        g.drawLine(x2, y2, x2+W, y2);
    }
    if (walls[1]) {//pravá
        g.drawLine(x2+W, y2, x2+W, y2+W);
    }
    if (walls[2]) { //levá
       g.drawLine(x2, y2+W, x2, y2);
    }
    if (walls[3]) { //dolní
        g.drawLine(x2+W, y2+W, x2, y2+W);
    }
}
```

10. Grafické rozhraní

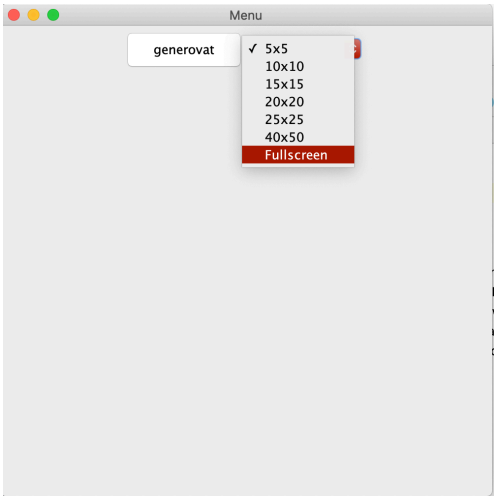
Grafické rozhraní používá knihovny Swing a AWT. V okně *Menu* si uživatel může vybrat velikost bludiště, poté klikne na generovat. Vykreslení menu s tlačítky realizuje třída HlavniMenu. Menu si můžete prohlédnout na obrázku číslo 3 a 4. Kód k grafickému rozhraní zde nebudu uvádět, jelikož neobsahuje žádně nezvyklé či zajímavé prvky. Všechen kód, ale naleznete v Git repositáři: https://github.com/gyarab/2021-1e-hurt-bludiste

Menu aplikace



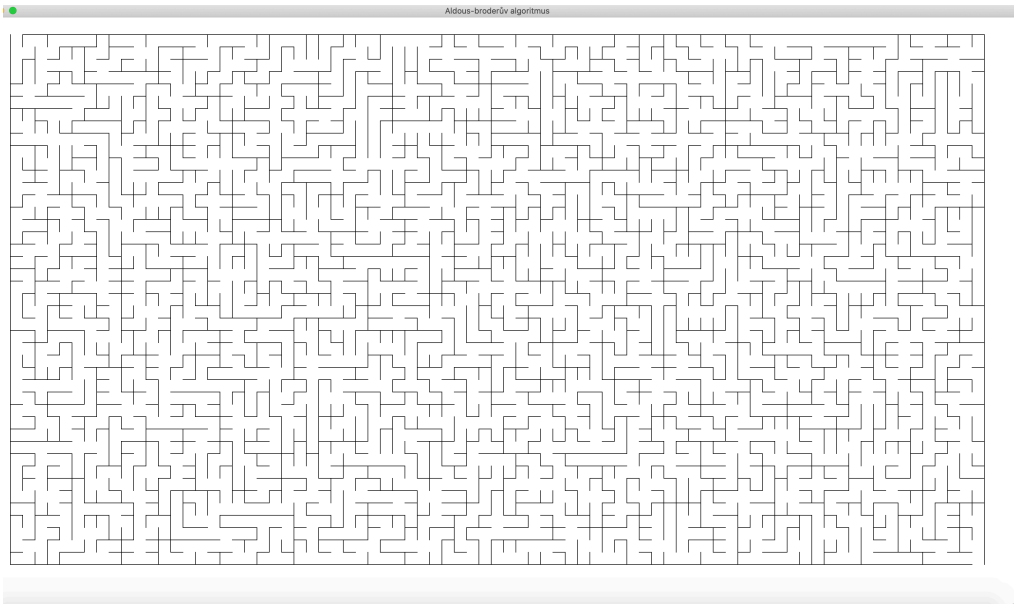
Obrázek 3

Výběr velikosti bludiště



Obrázek 4

Vygenerované bludiště ve fullscreen (42 × 72)



Obrázek 5

Bludiště se vykresluje na nové okno.

Závěr

Jelikož jsem s grafickým zpracováním projektu nepočítal, tak jsem se svým ročníkovým projektem velmi spokojen. Program by se dal rozšířit např. o – posupné vykreslování – budeme moct sledovat jak se bludiště postupně vytváří pomocí animací, možnost použít jiné algoritmy – Prohledávání do hloubky, Primův, Kruskalův…, možnost uložit bludiště (např. formát PNG).

Při vytváření tohoto projektu jsem se dozvěděl něco o tvorbě grafického rozhraní i o Jave samotné. Jsem rád, že jsem ročníkový projekt úspěšně dokončil a dokonce jsem ho mírně vylepšil oproti zadání.

Obsah