Gymnázium, Praha 6, Arabská 14

Obor programování

ROČNÍKOVÝ PROJEKT

Daniela Pilková, 2.E

Patnáctka

Duben 2022

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, Arabská14 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 ..... dne ..... Daniela Pilková .....

Název práce: Patnáctka

Autor: Daniela Pilková

Zadání:

Patnáctka – logická hra s čísly. Máme pole o rozměrech 4*4. Toto pole obsahuje náhodně rozhozená čísla od 1 do 15 a jedno volné políčko. Pokaždé, když klikneme na číslo sousedící s volným políčkem, toto číslo si vymění pozici s volným políčkem. Cílem hry je čísla seřadit.

Anotace:

Cílem projektu bylo vytvořit hru patnáctka. Celá hra je pojatá klasickým způsobem a to tak, že hráč se snaží náhodně rozházená čísla uspořádat do správného pořadí. Největším problémem u této hry je řešitelnost. Některé kombinace rozházených čísel totiž nejsou řešitelné konečným počtem tahů. Moje práce se tedy bude zabývat hlavně touto částí.

Obsah

1. Úvod 2

1.1 Hra Patnáctka a její pravidla 2

1.2 Historie Patnáctky 3

2. Řešitelnost Patnáctky 4

2.1 Co je inverze 5

2.2 Jak řešitelnost funguje 7

3. Další části programu 10

3.1 Metoda move 10

3.2 Metoda solution 11

4. Grafika 12

5. Nápady na zlepšení 12

6. Použité technologie 13

6.1 Nástroje 13

7. Závěr 13

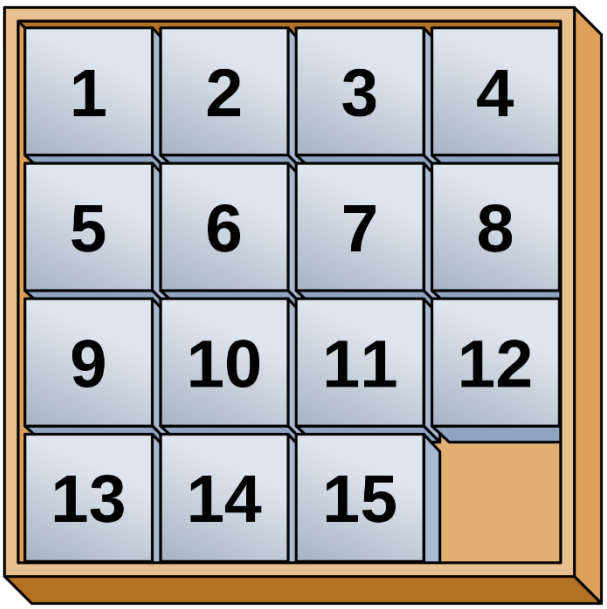
8. Seznam obrázků 13

9. Použité zdroje 14

1.Úvod

1.1 Hra Patnáctka a její pravidla

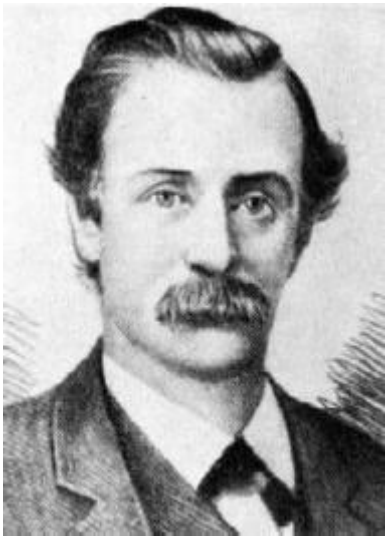
Patnáctka se skládá z 15 čtverců očíslovaných od 1 do 15, které jsou umístěny v poli o velikosti 4x4 tak, že jedna pozice z 16 je prázdná. Cílem je přemístit čtverce z daného libovolného počátečního uspořádání do správného pořadí, tedy od 1 do 15. U některých počátečních uspořádání je toto přeskupení možné, ale u jiných není. Teorie hlavolamu říká, že existují dvě skupiny počátečních uspořádání. Pro počáteční uspořádání v první skupině by mohl být hlavolam nakonec vyřešen, zatímco pro počáteční uspořádání v druhé skupině je neřešitelný. Rozdíl mezi těmito dvěma je v tom, že počáteční uspořádání v první skupině lze získat náhodným posouváním čtverců ze správného pořadí. Počáteční uspořádání neřešitelné skupiny se získají, když se navíc fyzicky zvednou dva sousední čtverce a jejich pozice se vymění (např. čtverce 14 a 15). Řešitelnost patnáctky dále popíšu v druhé kapitole. Kromě základní verze 4×4 existují také verze o jiné velikosti, např. jednodušší verze 3×3, která je někdy označována jako Lišák.



Obr.1 Jak vypadá hra patnáctka

1.2 Historie Patnáctky

Patnáctka, známá též jako Loydova patnáctka, je posuvný čtvercový hlavolam běžně, ale nesprávně připisovaný Samu Loydovi. Výzkum Slocuma a Sonnevelda však odhalil, že Sam Loyd hlavolam nevynalezl a neměl nic společného s jeho propagací nebo popularizací. Hlavolamové šílenství, které bylo vytvořeno Patnáctkou začalo v lednu 1880 ve Spojených státech, v dubnu v Evropě a skončilo v červenci 1880. Loyd poprvé začal tvrdit, že hlavolam vymyslel v roce 1891, že vynalezl a pokračoval až do své smrti v dvacetileté kampani, aby si neprávem připsal zásluhy za hlavolam. Skutečným vynálezcem byl Noyes Chapman, poštmistr z Canastoty v New Yorku, a v březnu 1880 požádal o patent.



Obr.2 Sam Loyd - americký šachista, šachový skladatel a tvůrce různých hlavolamů a matematických hříček

2. Řešitelnost Patnáctky

Máme pole o velikosti 4×4 s 15 čtverci, přičemž každý čtverec má čísla od 1 do 15 a jeden volný čtverec. Cílem je seřadit čtverce tak, aby čísla na nich byly ve správném pořadí za pomocí volného čtverce. Čtverce s čísly můžeme posouvat čtyřmi různými směry (vlevo, vpravo, nahoru a dolů) a zaměňovat je pouze s volný čtvercem.

2	1	3	4
5	6	7	8
9	10	11	12
13	14	15	X

Obr.3 Počáteční uspořádání

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	X

Obr.4 Správné uspořádání

V obrázku 2 a 3 X označuje místo, kde lze čtverce posunout a konečné uspořádání zůstává vždy stejné. Obecně platí, že pro dané pole šířky N můžeme zjistit, zda hlavolam N*N–1 je řešitelný nebo není, a to podle následujících pravidel:

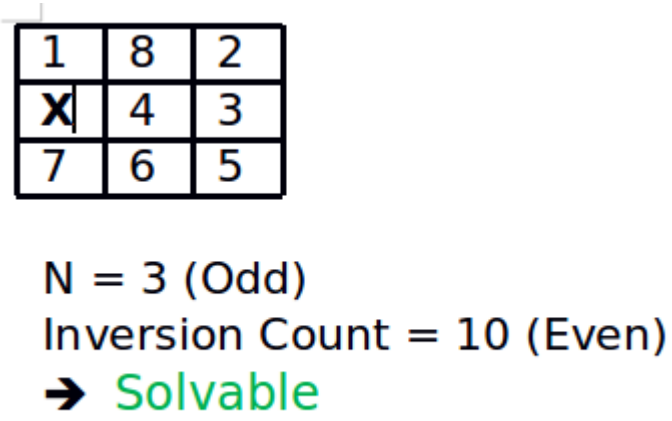
1. Pokud je N liché, je hlavolam řešitelný, pokud je počet záměn sudý v počátečním uspořádání.
2. Pokud je N sudé, je hlavolam řešitelný, pokud:
    - je volný čtverec na sudém řádku (počítáno zdola) a počet záměn je lichý.
    - je volný čtverec na lichém řádku (počítáno zdola) a počet záměn je sudý.
3. Pro všechny ostatní případy je hlavolam neřešitelný.

2.1 Co je inverze

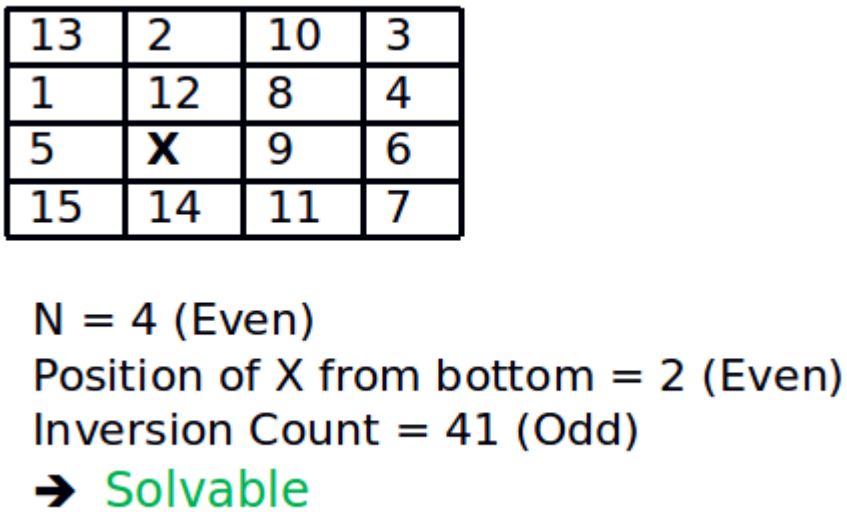
- Inverze = záměna

Když si představíme, že čísla ve čtvercích jsou vypsány jenom v jedné řadě (1D pole) namísto toho, aby byly rozhozené v N-řadách (2D pole), dvojice čísel (a,b) tvoří inverzi, pokud se první v řadě objeví b, přičemž a > b. Pro výše uvedený příklad, uvažujme, že čísla jsou zapsané v řadě takhle: 2 1 3 4 5 6 7 8 9 10 11 12 13 14 15 X
Výše uvedené pole tvoří pouze jednu inverzi, a to (2, 1).

Další příklady:



Obr.5



Obr.6

6	13	7	10
8	9	11	**X**
15	2	12	5
14	3	1	4

N = 4 (Even)
Position of X from bottom = 3 (Odd)
Inversion Count = 62 (Even)
➔ Solvable

Obr.7

3	9	1	15
14	11	4	6
13	X	10	12
2	7	8	5

N = 4 (Even)
Position of X from bottom = 2 (Even)
Inversion Count = 56 (Even)
➔ Not Solvable

Obr.8

2.2 Jak řešitelnost funguje

```
static int getInvCount(int[] arr) {
        int inv_count = 0;
        for (int i = 0; i < 15; i++) {
            for (int j = i + 1; j < 16; j++) {
                if (arr[j] > 0 && arr[i] > 0 && arr[i] > arr[j])
                    inv_count++;
            }
        }
        return inv_count;
    }
    static int findXPosition(int[][] puzzle) {
        for (int i = 4 - 1; i >= 0; i--)
            for (int j = 4 - 1; j >= 0; j--)
                if (puzzle[i][j] == 0)
                    return 4 - i;
        return 0;
    }
    static boolean isSolvable(int[][] puzzle) {
        int[] flat = new int[16];
        int c = 0;
        for (int x = 0; x < 4; x++)
            for (int y = 0; y < 4; y++) {
                flat[c] = puzzle[x][y];
                c++;
            }
        int invCount = getInvCount(flat);
        int pos = findXPosition(puzzle);
        if ((pos & 1) == 1)
            return !((invCount & 1) == 1);
        else
            return (invCount & 1) == 1;
    }
}
```

Fakt 1: Pro pole liché šířky všechny povolené pohyby zachovávají polaritu (sudou nebo lichou) počtu záměn.

Důkaz pro fakt 1:

- Přesouváním čtverce v řádku (doleva nebo doprava) se nezmění počet záměn, a proto ani polarita.
- Přesouváním čtverce v sloupci (nahoru a dolů) se může změnit počet záměn. Čtverec se posouvá kolem sudého počtu jiných čtverců (N-1). Takže tento posun buď zvýší nebo sníží počet záměn o 2, nebo jej ponechá.

Fakt 2: Pro pole sudé je následující neměnné: počet inverzí je sudý a volný čtverec se nachází na lichém řádku**.**

12	1	10	2
7	11	4	14
5		9	15
8	13	6	3

49 inversions blank on even row from bottom

12	1	10	2
7		4	14
5	11	9	15
8	13	6	3

48 inversions blank on odd row from bottom

Obr. 9

Příklad: Uvažujme o výše uvedeném posunu. Počet záměn v poli, které se nachází nahoře, je 49 a volný čtverec je v sudé řádce ze spodu. Takže hodnota neměnné je „false == false“, což je pravda. Počet záměn v poli, které se nachází dole, je 48, protože 11 ztratila dvě inverze, ale zase 14 jednu

inverzi získala. Volný čtverec je na liché řádce od spoda. Takže hodnota neměnné je „true == true“, což je pořád pravda.

Důkaz pro fakt 2:

- Posouváním čtverce v řádce (doleva nebo doprava) se nezmění počet záměn ani řádka, na které se nachází volný čtverec.
- Posouváním čtverce v sloupci (nahoru nebo dolů) se změní počet záměn. Čtverec se pohybuje kolem lichého počtu jiných čtverců (N-1). Takže počet záměn se mění lichým počtem opakování. Také řádek volného čtverce se změní z lichého na sudý nebo ze sudého na lichý. Takže obě poloviny neměnných se změní. To znamená, že jejich hodnota je zachována.

Kombinací Faktu 1 a Faktu 2 dostaneme **Fakt 3:**

- Pokud je šířka pole lichá, pak každé řešitelné počáteční uspořádání má sudý počet záměn.
- Pokud je šířka pole sudá, pak každé řešitelné počáteční uspořádání má:
    - sudý počet inverzí, pokud je volný čtverec na lichém řádku (počítáno ze spodu).
    - Lichý počet inverzí, pokud je volný čtverec na sudém řádku (počítáno ze spodu).

Důkaz pro fakt 3:

- Vyřešený stav má tyto vlastnosti.
- Tyto vlastnosti jsou zachovány každým povoleným posunem.
- Jakékoli řešitelné počáteční uspořádání je možné získat z vyřešeného stavu nějakou sekvencí povolených posunů.

3. Další části programu

3.1 Metoda move

```
void Move(JButton button1, JButton button2) {
        String shuffle = button2.getText();
        if (shuffle == "") {
            button2.setText(button1.getText());
            button1.setText("");
        }
    }
```

Tato metoda slouží k prohazování čísel s volným čtvercem. Vezme text jednoho čtverce (resp. číslo) a nastaví ho na místo volného čtverce a naopak namísto čtverce, kde bylo původně číslo nastaví prázdný text. Metoda je dále volána u každého z tlačítek.

Příklad:

```
private void buttonOneActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Move(buttonOne, buttonTwo);
        Move(buttonOne, buttonFive);
        Solution();
        counter++;
}
```

3.2 Metoda solution

```
public void Solution() {
        String solution = buttonOne.getText();
        String solution1 = buttonTwo.getText();
        String solution2 = buttonThree.getText();
        String solution3 = buttonFour.getText();
        String solution4 = buttonFive.getText();
        String solution5 = buttonSix.getText();
        String solution6 = buttonSeven.getText();
        String solution7 = buttonEight.getText();
        String solution8 = buttonNine.getText();
        String solution9 = buttonTen.getText();
        String solution10 = buttonEleven.getText();
        String solution11 = buttonTwelve.getText();
        String solution12 = buttonThirteen.getText();
        String solution13 = buttonFourteen.getText();
        String solution14 = buttonFifteen.getText();


        if (solution == "1" && solution1 == "2" && solution2 == "3" &&
solution3 == "4" && solution4 == "5" && solution5 == "6" && solution6 == "7"
                && solution7 == "8" && solution8 == "9" && solution9 == "10"
&& solution10 == "11" && solution11 == "12" && solution12 == "13"
                && solution13 == "14" && solution14 == "15") {
            JOptionPane.showMessageDialog(this, "Vyhrál jsi!", "Patnáctka",
JOptionPane.INFORMATION_MESSAGE);
        }
        kliky.setText("      " + Integer.toString(counter));
    }
```

Tato metoda ověřuje, jestli jsou již čísla ve správném pořadí, a když ano vyskočí správa o tom, že daný člověk vyhrál. Zároveň tato metoda obsahuje počítadlo, které počítá počet posunů.

4. Grafika

Obr. 10

Grafika je jednoduchá, vytvořená pomocí třídy JavaSwing. V levé části obrázku můžeme vidět hrací plochu. Jedná se o 16 tlačítek očíslovaných od 1 do patnácti a jedno políčko volné. Ve střední části se nachází počitadlo, které počítá počet posunů. V pravé části můžete nalézt tři tlačítka. Po stisknutí tlačítka řešení se čísla automaticky uspořádají do správného pořadí. Když stisknete tlačítko reset čísla se automaticky rozhází a hráč začíná znova. Když stisknete tlačítko Exit, vyskočí na vás okno s dotazem jestli chcete hru opustit. V pravém horním rohu můžete najít ještě jedno tlačítko, a to pravidla hry.

5. Nápady na zlepšení

Část, kterou je potřeba nejvíce vylepšit je grafika. Rozhodně by bylo potřeba tomu věnovat více času. Dále mě během řešení této úlohy napadla podobná hra na stejném principu, akorát s obrázky. Na každý čtverec by připadla část obrázku a cílem by bylo poskládat kompletní obrázek.

6. Použité technologie

K tvorbě programu jsem použila třídu JavaSwing. Dále jsem použila třídu Random k proházení hodnot pro počáteční uspořádání.

6.1 Nástroje

K vývoji mé aplikace jsem použila IDE Netbeans 8.2 a JDK 8.

obr. 10 NetBeans

obr. 11 JDK

7. Závěr

Závěrem bych chtěla vyhodnotit svou práci. Práce se mi víceméně povedla podle představ, ale mohla být více propracovaná. Nápady na zlepšení jsou uvedené výše.

8. Seznam obrázků

Obr.1 Patnáctka. In: *Https://www.google.com* [online]. [cit. 2022-04-28]. Dostupné z: https://www.google.com/search?q=patnactka&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiuhYrP7qX3AhWMnqQKHShQBJoQ_AUoAXoECAIQAw&biw=1366&bih=625&dpr=1#imgrc=RAXDsHD--f8PtM

Obr.2 Sam Loyd. In: *Https://www.goodreads.com* [online]. [cit. 2022-04-28]. Dostupné z: https://www.goodreads.com/author/show/560282.Sam_Loyd

Obr.3 a 4 Počáteční a správné uspořádání. In: *Https://www.geeksforgeeks.org* [online]. [cit. 2022-04-28]. Dostupné z: https://www.geeksforgeeks.org/check-instance-15-puzzle-solvable/

Obr.5-9 Řešitelnost. In: *Https://www.geeksforgeeks.org* [online]. [cit. 2022-04-28]. Dostupné z: https://www.geeksforgeeks.org/check-instance-15-puzzle-solvable/

Obr.10

9. Zdroje

Patnáctka. *Https://mathworld.wolfram.com* [online]. [cit. 2022-04-28]. Dostupné z: https://mathworld.wolfram.com/15Puzzle.html

Pravidla. *Https://cs.wikipedia.org* [online]. [cit. 2022-04-28]. Dostupné z: https://cs.wikipedia.org/wiki/Patn%C3%A1ctka

Historie. *Http://www.cut-the-knot.org* [online]. [cit. 2022-04-28]. Dostupné z: http://www.cut-the-knot.org/pythagoras/fifteen.shtml

Sam Loyd. *Http://mesosyn.com* [online]. [cit. 2022-04-28]. Dostupné z: http://mesosyn.com/mental1-8.html

Řešitelnost. *Https://www.geeksforgeeks.org* [online]. [cit. 2022-04-28]. Dostupné z: https://www.geeksforgeeks.org/check-instance-15-puzzle-solvable/