Informatyka, studia zaoczne, inż. III rok semestr VI

**Sztuczna Inteligencja i Systemy Ekspertowe 2015/2016**

prowadzący: Mgr inż. Paweł Tarasiuk

Data oddania:\_\_\_\_\_\_\_\_\_\_\_ Ocena:\_\_\_\_\_\_\_\_\_\_\_

Marek Gadzalski 191422

Grzegorz Głąb 191425

**Zadanie 1 - przeszukiwanie przestrzeni stanów**

1. **CEL**

Celem zadania jest napisanie programu, który będzie rozwiązywał łamigłówkę „piętnastkę”, czyli będzie wyznaczał taki ciąg ruchów, które przeprowadzą układankę z układu początkowego do układu wzorcowego. Należy przebadać, jak zachowują się w przypadku tego problemu różne metody przeszukiwania przestrzeni stanów:

Strategia DFS "w głąb"

Strategia BFS "wszerz"

Strategia "A\*"

Dla strategii A\* zaimplementowano dwie metody heurystyczne

Odległość Manhattan i odległość Hamminga

1. **WPROWADZENIE**
   1. **Pojęcia**

**Graf** - abstrakcyjna forma przedstawienia powiazań (krawędzi) miedzy obiektami (wierzchołki)

**Drzewo** - podgrupa grafów. Graf acykliczny. W strukturze powiazań wyróżnia się relacje rodzic-potomek. Dla każdego potomka można określić jego rodzica (wyjątkiem jest korzeń - obiekt stanowiący początek drzewa)

* 1. **Układanka „Piętnastka”**

"Piętnastka", znana również pod angielską nazwą "Fifteen Puzzle", składa się z ramki, w której osadzone jest 15 klocków. Klocki można przesuwać, ponieważ w ramce pozostaje wolne miejsce o wielkości jednego klocka (cała plansza ma wymiar 4x4). Gra polega na takim przesuwaniu klocków, aby z pewnego losowego układu początkowego (przykład poniżej)

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

Aby uzyskać układ wzorcowy odpowiadający poniższemu:

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

* 1. **Rozwiązywalność układanki Piętnastka**

Inwersja jest to stan, gdy klocek układanki jest poprzedzony przez inny klocek w układance o wyższym numerze. Inwersje sprawdzamy od strony lewej do prawej, zaczynając od górnego rzędu. Stąd wynika, iż stan naturalny (rozwiązanie) nie posiada inwersji.

Aby sprawdzić czy układanka jest rozwiązywalna należy policzyć wszystkie inwersje w układance.

Rozwiązywalność stanu układanki o nieparzystej liczbie kolumn (szerokości) można stwierdzić dla stanu gdzie liczba inwersji w układance jest liczbą parzystą.

Gdy układanka ma parzystą liczbę kolumn (szerokość) jej stan jest rozwiązywalny, gdy licząc od dołu wiersz, w którym znajduje się puste miejsce jest wierszem parzystym (kolejność wiersza od dołu) oraz liczba inwersji dla danego stanu jest liczbą nieparzystą.

Taka układanka – o parzystej liczbie kolumn jest rozwiązywalna również, gdy puste miejsce jest w wierszu nieparzystym, licząc od dołu oraz liczba inwersji dla danego stanu jest liczbą parzystą.

Inaczej układanka nie jest rozwiązywalna.

**Uwaga:** Poniższy stan układanki:

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

Jest stanem rozwiązywalnym.

Dla 12 jest 11 inwersji

Dla 9 jest 8 inwersji

Dla 13 jest 10 inwersji

Dla 15 jest 11 inwersji

Dla 11 jest 9 inwersji

Dla 10 jest 8 inwersji

Dla 14 jest 8 inwersji

Dla 7 jest 6 inwersji

Dla 8 jest 6 inwersji

Dla 6 jest 5 inwersji

Dla 2 jest 1 inwersja

Dla 4 są 2 inwersje

Dla 3 jest 1 inwersja

Dla 5 jest 1 inwersja

Razem istnieje 87 inwersji, puste pole jest w wierszu 4.

Zatem, parzysta liczba kolumn (4), nieparzysta liczba inwersji, parzysty numer wiersza potwierdza rozwiązywalność.

**Jednak żadną metodą nie udało się programowi rozwiązać tego stanu układanki!**

* 1. **Rozwiązanie problemu**

Sprowadza się do przeszukania drzewa powstałego w wyniku dozwolonych przejść pomiędzy stanami gry. Korzeniem powstałego drzewa jest stan wejściowy gry. Pozostałe wierzchołki są możliwymi rozwiązaniami.

* 1. **Strategia DFS (**Depth-First Search)

Jest to strategia, która gwarantuje nam znalezienie rozwiązania i polega na przeglądaniu grafu w głąb jednej gałęzi najdłużej jak to możliwe. Po czym następuje wycofanie się do pierwszego możliwego potomka oraz ponowne przeglądanie wzdłuż jego gałęzi. W strategii DFS wybrany wierzchołek należy umieścić na stosie zaznaczyć, jako odwiedzony a następnie przejść do jego następnika.

* 1. **Strategia BFS** (Breadth-First Search)

Jest to strategia, która gwarantuje nam znalezienie rozwiązania i polega na przeglądaniu grafu w szerz. Najpierw przeglądani są bezpośredni potomkowie, następnie dla każdego potomka przeglądani są jego nieodwiedzeni potomkowie. Aby przeszukać graf wszerz (BFS) należy zamiast stosu wykorzystać kolejkę do przechowywania wierzchołków a kolejnych nieodwiedzonych następników szukać od początku macierzy

* 1. **Strategia A\***

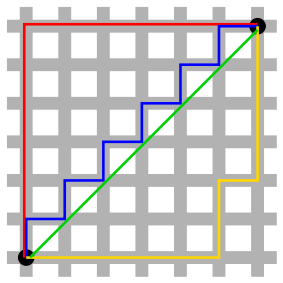
Jest to metoda heurystyczna, to znaczy nie gwarantuje znalezienia najlepszego rozwiązania. Jest to zmodyfikowana wersja algorytmu Best-First. BF polega na przeglądaniu potomka, który jest najbardziej zbliżony do oczekiwanego rozwiązania. Decyduje o tym funkcja heurystyczna (im mniejsza wartość zwrócona przez funkcje, tym potomek bardziej obiecujący). O jakości potomka w algorytmie A\* decyduje suma funkcji heurystycznych przejścia od korzenia do rodzica i wartości funkcji heurystycznej dla potomka.

* 1. **Odległość Manhattan**

Odległość między dwoma punktami w przestrzeni metrycznej. Odległość dwóch punktów w tej metryce to suma wartości bezwzględnych różnic ich współrzędnych.

W przestrzeni  \mathbb R^n metryka ta dana jest wzorem:

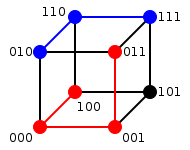
Metryka jest idealizacją sieci ulic biegnących wyłącznie w kierunkach wschód-zachód oraz północ-południe. Każda trasa, jaką można przemieścić się z jednego punktu do drugiego będzie miała długość zgodną z metryką miejską.



Zielona przekątna – odległość względem metryki euklidesowej (\scriptstyle{6\sqrt{2}}, tj. ok. 8,48 [j.](https://pl.wikipedia.org/wiki/Jednostka_miary))  
Pozostałe krzywe – odległość względem metryki miejskiej (12 [j.](https://pl.wikipedia.org/wiki/Jednostka_miary))

* 1. **Odległość Hamminga**

Odległość Hamminga jest miarą odmienności dwóch ciągów o takiej samej długości, wyrażająca liczbę miejsc (pozycji), na których te dwa ciągi się różnią. Innymi słowy jest to najmniejsza liczba zmian (operacji zastępowania elementu innym), jakie pozwalają przeprowadzić jeden ciąg na drugi.



Odległość: 100→011 = 3 (kolor czerwony); 010→111 = 2 (kolor niebieski)

1. **IMPLEMENTACJA**

Program jest zaimplementowany w języku Java – wersja 8.

Pierwszy program jest zaimplementowany w klasie Fifteen i jest konfigurowany przez plik fifteen.properties:

rows=4

cols=4

#initial=0 12 9 13 15 11 10 14 7 8 6 2 4 3 5 1

initial=random

direction=GPDL

fileToSave\_BFS=Solution\_BFS.txt

fileToSave\_DFS=Solution\_DFS.txt

fileToSave\_A\*1=Solution\_aStarManhattan.txt

fileToSave\_A\*2=Solution\_aStarHamming.txt

Plik ten konfiguruje zachowanie programu, rozmiar planszy, stan planszy, pliki, w których zapiszemy sekwencję rozwiązywania oraz kolejność rozpatrywanych kierunków.

Można skonfigurować program parametrami wejściowymi:

-b/--bfs porządek Strategia przeszukiwania wszerz

-d/--dfs porządek Strategia przeszukiwania w głąb

-n/--nn id\_heurystyki Strategia najpierw najlepszy (A\*)

Gdzie porządek jest permutacją zbioru {'L','P','G','D'} określającą porządek przeszukiwania sąsiedztwa bieżącego stanu. Przykładowo napis DGLP oznacza porządek następujący porządek przeszukiwania: dół, góra, lewo, prawo. Jeżeli porządek zaczyna się od 'R' program przyjmuje kolejność losową (czytaj w każdym węźle grafu losujemy kolejność przeszukania).

**Opis wejścia**

W pierwszej linii wejścia znajdują się dwie liczby całkowite **w k**, odpowiednio pionowy (ilość wierszy) i poziomy (ilość kolumn) rozmiar ramki. Każdy z następnych w wierszy standardowego wyjścia zawiera k oddzielonych spacjami liczb całkowitych opisujących element układanki, przy czym wartość 0 oznacza pole puste.

**Opis wyjścia**

Na standardowym wyjściu powinny się pojawić dwie linie. Pierwsza zawiera jedną liczbę całkowitą n określającą długość znalezionego rozwiązania (ilość kroków potrzebnych do rozwiązania łamigłówki). W drugiej zaś linijce jest napis długości n złożony z wielkich liter łacińskich ze zbioru {'L','P', 'G', 'D'} opisującego kolejne ruchy do wykonania.

Jeżeli dla danego układu początkowego nie istnieje rozwiązanie, w pierwszym i jedynym wierszy wyjścia pojawi się liczba -1.

Po uruchomieniu następuje wczytanie konfiguracji i rozpoczęcie działania programu.

Program najpierw sprawdza rozwiązywalność i poprawność stanu układanki.

Jeżeli układanka jest rozwiązywalna to rozpoczynamy rozwiązywanie względem podanych parametrów wejściowych.

Program zapisuje sekwencję stanów w celu wizualizacji rozwiązywania w Aplikacji Przeglądającej:

**Aplikacja przeglądająca**

Zadaniem drugiej aplikacji, jest wizualizowanie procesu rozwiązywania łamigłówki.

Jest to prosty program, przyjmujący, jako parametr ścieżkę do pliku z sekwencją stanów i ją wyświetla.

1. **WYNIKI I WNIOSKI**

W celach badawczych program został skonfigurowany, aby prowadzić badanie w podanej sekwencji:

1 – Sprawdź rozwiązywalność

2 – Strategia A\* z funkcją Manhattan

3 – Strategia A\* z funkcją Hamminga

4 – Strategia BFS

5 – Strategia DFS

Po każdym wykonaniu strategii zakończonym rozwiązaniem układanki wyświetlany jest status w postaci:

Czasu ułożenia, ilości ruchów potrzebnych do rozwiązania, sekwencji ruchów.

**Badanie 1a:**

Puzzle to solve:

5 3

1 2

0 4

Puzzle is solvable!

Direction: GPDL

----------

Manhattan heuristic method

Solved using A\* in 0 ms

Moves to solve: 13

PGLGPDDLGGPDD

----------

Hamming heuristic method

Solved using A\* in 0 ms

Moves to solve: 13

PGLGPDDLGGPDD

----------

Solved using BFS in 0 ms

Moves to solve: 13

PGLGPDDLGGPDD

----------

Solved using DFS in 0 ms

Moves to solve: 81

PGLDPGLDPGGLDDPGLDPGLGPDLDPGLDPGLGPDLDPGLDPGLGPDLDPGLDPGLGPDLDPGLGPDLDPGLDPGGLDDP

**Badanie 1b:**

Puzzle to solve:

5 3

1 2

0 4

Puzzle is solvable!

Direction: R

----------

Manhattan heuristic method

Solved using A\* in 10 ms

Moves to solve: 13

PGLGPDDLGGPDD

----------

Hamming heuristic method

Solved using A\* in 0 ms

Moves to solve: 13

PGLGPDDLGGPDD

----------

Solved using BFS in 0 ms

Moves to solve: 13

PGLGPDDLGGPDD

----------

Solved using DFS in 10 ms

Moves to solve: 17

PGGLDPGLDPDLGGPDD

**Badanie 2a**

Puzzle to solve:

3 2 5

0 4 1

Puzzle is solvable!

Direction: GPDL

----------

Manhattan heuristic method

Solved using A\* in 0 ms

Moves to solve: 14

PGLDPPGLDLGPDP

----------

Hamming heuristic method

Solved using A\* in 0 ms

Moves to solve: 14

PGLDPPGLDLGPDP

----------

Solved using BFS in 0 ms

Moves to solve: 14

PGLDPPGLDLGPDP

----------

Solved using DFS in 10 ms

Moves to solve: 106

PPGLLDPPGLLDPPGLLDPPGLLDPPGLDLGPDPGLLDPPGLDPGLLDPPGLDPGLLDPPGLDPGLLDPPGLDPGLLDPGLDPPGLLDPPGLLDPPGLLDPPGLDP

**Badanie 2b:**

Puzzle to solve:

3 2 5

0 4 1

Puzzle is solvable!

Direction: R

----------

Manhattan heuristic method

Solved using A\* in 10 ms

Moves to solve: 14

PGLDPPGLDLGPDP

----------

Hamming heuristic method

Solved using A\* in 10 ms

Moves to solve: 14

PGLDPPGLDLGPDP

----------

Solved using BFS in 10 ms

Moves to solve: 14

PGLDPPGLDLGPDP

----------

Solved using DFS in 10 ms

Moves to solve: 148

PPGLLDPPGLDPGLLDPGLDPPGLDLGPPDLGLDPGPDLGLDPPGLLDPGPDLLGPPDLGPDLLGPPDLGLDPGLDPGPDLGLDPPGLLDPGPDLGPDLLGPDPGLDLGPPDLLGPDLGPPDLGPDLGLDPGLDPGPDLGPDLLGPPD

**Badanie 3a:**

Puzzle to solve:

3 7 6

1 2 8

0 5 4

Puzzle is solvable!

Direction: GPDL

----------

Manhattan heuristic method

Solved using A\* in 10 ms

Moves to solve: 18

PGGLDPDPGGLDDLGPDP

----------

Hamming heuristic method

Solved using A\* in 40 ms

Moves to solve: 18

PPGLGLDDPGPGLDLDPP

----------

Solved using BFS in 220 ms

Moves to solve: 18

PGGLDPDPGGLDDLGPDP

----------

Solved using DFS in 970 ms

Moves to solve: 38474

PPGLLDPPGLLDPPGLLDPPGLLDPPGLDLGPDLGGPDLDPPGLLDPPGLLDPPGLLDP…

**Badanie 3b:**

Puzzle to solve:

3 7 6

1 2 8

0 5 4

Puzzle is solvable!

Direction: R

----------

Manhattan heuristic method

Solved using A\* in 10 ms

Moves to solve: 18

PGGLDPDPGGLDDLGPDP

----------

Hamming heuristic method

Solved using A\* in 40 ms

Moves to solve: 18

PPGLGLDDPGPGLDLDPP

----------

Solved using BFS in 219 ms

Moves to solve: 18

PPGLGLDPDLGPPGLDDP

----------

Solved using DFS in 600 ms

Moves to solve: 67410

…

**Badanie 4a:**

Puzzle to solve:

5 1 2

6 3 9

8 4 10

0 7 11

Puzzle is solvable!

Direction: GPDL

----------

Manhattan heuristic method

Solved using A\* in 20 ms

Moves to solve: 20

GPPGLLGPPDLLDPDLGPDP

----------

Hamming heuristic method

Solved using A\* in 30 ms

Moves to solve: 20

GPPGLLGPPDLLDDPGLDPP

----------

Solved using BFS in 3554 ms

Moves to solve: 20

GPPGLLGPPDLLDPDLGPDP

----------

Dla DFS działanie program zostało przerwane po 60 minutach.

**Badanie 4b:**

Puzzle to solve:

5 1 2

6 3 9

8 4 10

0 7 11

Puzzle is solvable!

Direction: R

----------

Manhattan heuristic method

Solved using A\* in 30 ms

Moves to solve: 20

GPPGLLGPPDLLDPDLGPDP

----------

Hamming heuristic method

Solved using A\* in 40 ms

Moves to solve: 20

GPPGLLGPPDLLDDPGLDPP

----------

Solved using BFS in 4486 ms

Moves to solve: 20

GPPGLLGPPDLLDDPGLDPP

----------

Dla DFS działanie program zostało przerwane po 60 minutach.

**Wnioski:**

1 – Strategia A\* w połączeniu z metodą Manhattan jest zdecydowanie najlepsza. Osiągnięto najkrótszy czas odnajdywania drogi przy tej metodzie

2 - Następnie pod względem szybkości działania znajdują się strategie: A\* z metodą Hamminga, BFS – wszystkie znajdywały tak samo krótką drogę, ale w dłuższym czasie działania.

3 – Metoda DFS jest zdecydowanie najgorsza do rozwiązywania tej układanki. Nie znajduje optymalnej drogi do rozwiązania i działa dużo dłużej od pozostałych.

4 – Porządek ruchów ma drobny wpływ na działanie algorytmów. Porządek losowy raczej pogarsza wyniki czasowe. Lepszy jest porządek ustalony z góry.