Informatyka, studia dzienne, inż I st. semestr VI

Sztuczna inteligencja i systemy ekspertowe 2017/2018

Prowadzący: dr inż. Krzysztof Lichy Piątek, 11:45

Data oddania: _____ Ocena: _____

Michał Białecki 210141
Marcin Kacprowicz 210211

Zadanie 1: "Piętnastka"

1. Cel

Celem naszego zadania było przygotowanie programu rozwiązującego łamigłówkę, potocznie nazywaną „piętnastką", przy użyciu dowolnego języka programowania i z wykorzystaniem odpowiednich metod. Następnie mieliśmy przeanalizować otrzymane dane.

2. Wprowadzenie

Analizowana przez nas łamigłówka to prosta logiczna układanka, która w swojej domyślnej formie ma 15 zajętych pól i jedno puste. Rozwiązywanie jej polega na zamienianiu pustego pola z sąsiadującymi zajętymi polami. Prawidłowy układ łamigłówki, czyli rozwiązanie, to liczby ułożone w rzędach od 1 do 15, z pustym polem na pozycji 16. Jednym ruchem możemy zamienić miejscami puste pole i przyległe pole sąsiadujące. Wszystkich możliwych ustawień początkowych układanki jest ponad 20 bilionów, aczkolwiek rozwiązywalna jest tylko połowa.

Na potrzeby naszego zadania pole puste oznaczyliśmy cyfrą 0. Aktualny układ pól nazywamy stanem. Stanem sąsiadującym nazywamy stan łamigłówki przesunięty o jedno pole względem obecnego stanu. Rozróżniamy także dwa stany układanki, które pojawiają się w trakcie działania algorytmów i są to: stan odwiedzony – czyli stan, który już znamy, ale nie sprawdziliśmy jeszcze czy jest on rozwiązaniem naszej łamigłówki i stan przetworzony – czyli stan, który znamy i został już on sprawdzony pod względem rozwiązania.

W celu rozwiązania łamigłówki skorzystaliśmy z następujących metod: Przeszukiwanie wszerz (ang. Breadth-first search), Przeszukiwanie w głąb (ang. Depth-first search) i Algorytmu A*.

Przeszukiwanie wszerz, czyli inaczej BFS, polega na przeszukiwani grafu zaczynając od wierzchołka, następnie sprawdzając wszystkie jego dzieci, a dopiero później sprawdzając potoków jego dzieci. W trakcie działania algorytmu musimy zapamiętywać odwiedzone już stany.

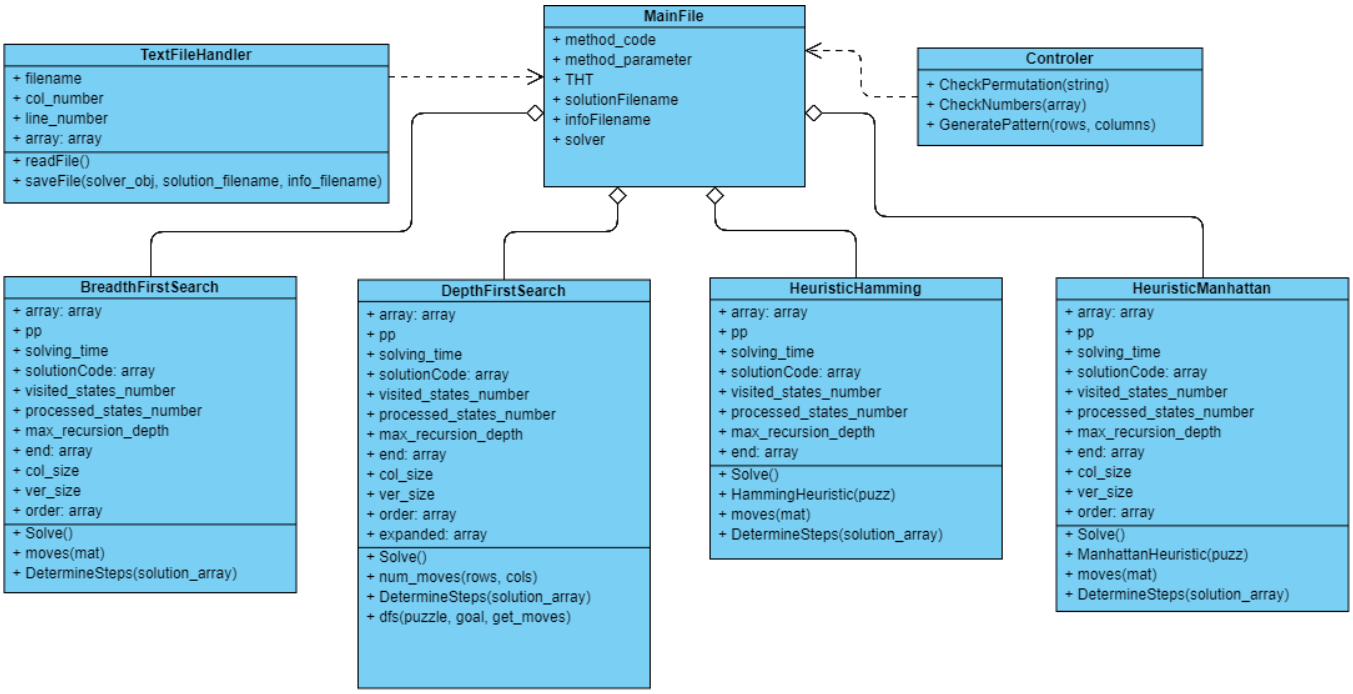
Przeszukiwanie w głąb, czyli inaczej DFS, polega na przeszukiwaniu grafu zaczynając od wierzchołka, następnie sprawdzając wszystkie krawędzie wychodzące z podanego wierzchołka. Po zbadaniu wszystkich krawędzi wychodzących z danego wierzchołka algorytm powraca do wierzchołka, z którego dany wierzchołek został odwiedzony. Po odwiedzeniu każdego wierzchołka należy zapamiętać stan tego wierzchołka.

Algorytm A* to algorytm heurystyczny służący do znajdywania jak najkrótszej ścieżki w grafie. W naszym przypadku wykorzystaliśmy heurystyki Hamminga i Manhattan. Oszacowują one jaki jest koszt dotarcia do rozwiązania, a następnie rozwijają te stany, których koszt jest najmniejszy. Heurystyka Hamminga, inaczej HAMM, polega na obliczeniu wszystkich pól, które nie znajdują się na swoim miejscu. Następnie dodajemy głębokość obecnego stanu. W ten sposób otrzymujemy koszt badanego stanu. Heurystyka Manhattan, inaczej MANH, natomiast polega na obliczeniu przesunięć poziomych i pionowych względem rozwiązania i dodawaniu do tego głębokości obecnego stanu. W ten sposób otrzymujemy koszt stanu dla tej metody.

3. Opis implementacji

Nasz program został napisany w języku Python. Składa się z pliku głównego MainFile a także z 6 klas, gdzie 4 z nich odpowiadają innej metodzie przeszukiwania grafu, czyli BFS, DFS, HAMM i MANH, klasa TextFileHandler służy do operacji na plikach, a Controler do sprawdzania poprawności podanych atrybutów.

MainFile zależnie od użytej metody tworzy instancje klas odpowiedzialnych za poszczególne strategie, a następnie przy użyciu ich metod rozwiązuje układankę. Klasa TextFileHandler jest odpowiedzialna za otworzenie wskazanego pliku, wczytanie z niego danych i ostatecznie do zapisania rozwiązań do poszczególnych plików.



Rysunek 1. Diagram UML klas.

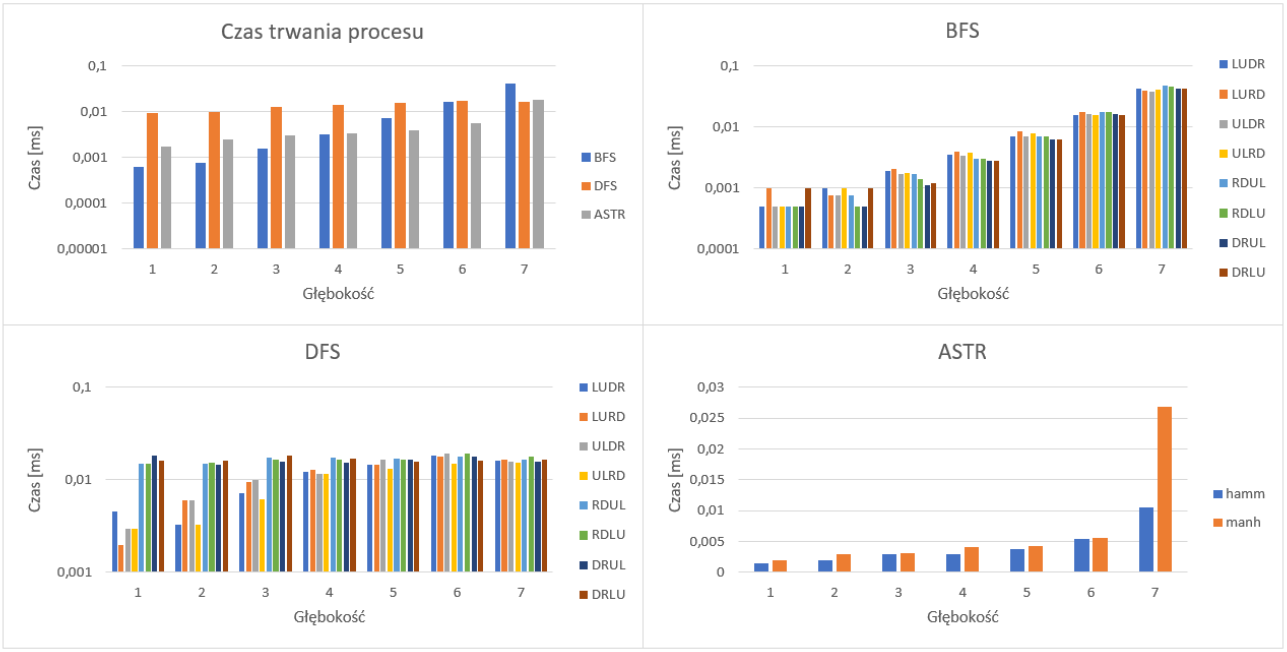
4. Materiały i metody

W celu rozwiązania układanki trzeba podać jako argumenty pliku głównego MainFile.py skrót oznaczający wybraną metodę rozwiązywania łamigłówki, czyli bfs, dfs lub astr – w celu skorzystania z heurystyki. Następnie, w przypadku BFS i DFS, trzeba podać kierunek przeszukiwań np. RLUD. Natomiast dla heurystyki należy podać jaki rodzaj heurystyki nas interesuje: hamm lub manh. Następnym argumentem jest plik z układanką, plik do przechowania rozwiązania i plik do przechowania pozostałych informacji. Plik z pozostałymi informacjami zawiera informacje o długości rozwiązania, ilości odwiedzonych stanów, ilości przetworzonych stanów, maksymalnej osiągniętej głębokości i czas trwania rozwiązywania układanki. Plik z układanką musi być w odpowiednim formacie, tzn. w pierwszej linii musi być podany rozmiar układanki, a w kolejnych liniach elementy układanki. Prawidłowy, przykładowy format:

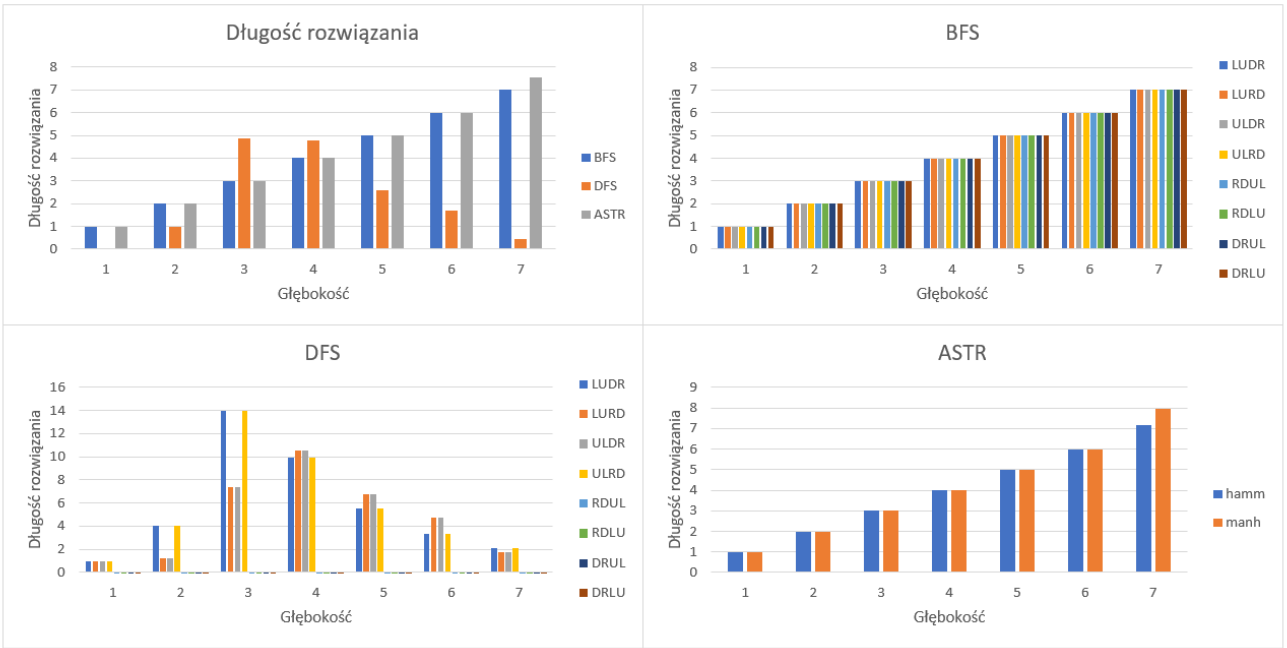
4 4
1 2 3 0
5 6 7 4
9 10 11 8
13 14 15 12

5. Wyniki

Poniżej przedstawiamy wyniki przeprowadzonych przez nas badań 413 układów przy użyciu wszystkich strategii.

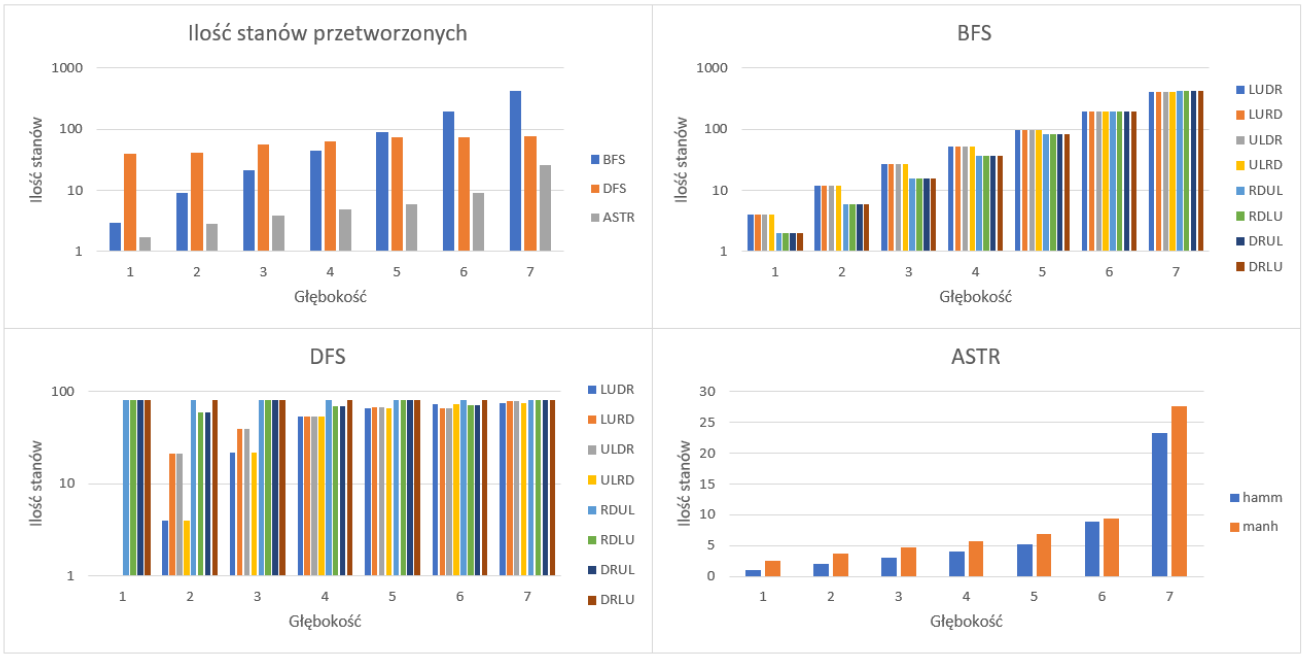


Rysunek 2. Wykresy przedstawiające średni czas trwania procesu poszukiwania rozwiązania dla wszystkich metod i z wyszczególnieniem dla różnych kolejności przeszukiwania.

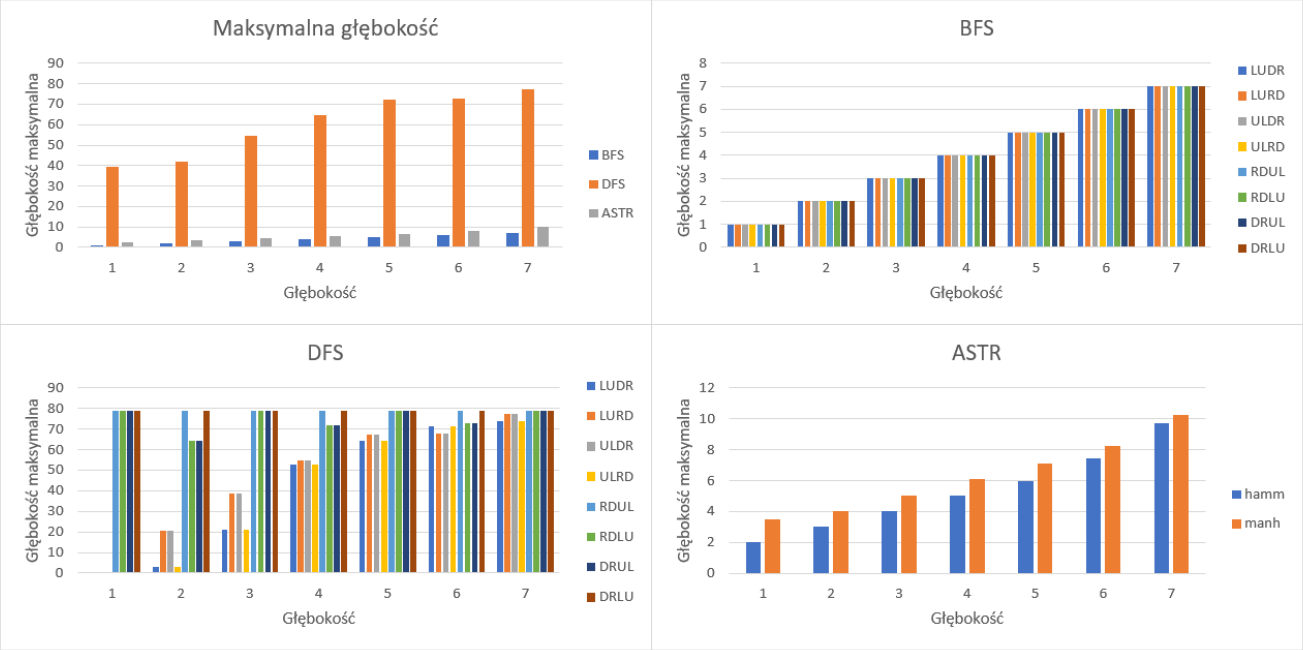


Rysunek 3. Wykresy przedstawiające średnią długość rozwiązania dla wszystkich metod i z wyszczególnieniem dla różnych kolejności przeszukiwania.

Rysunek 4. Wykresy przedstawiające średnią ilość stanów odwiedzonych dla wszystkich metod i z wyszczególnieniem dla różnych kolejności przeszukiwania.



Rysunek 5. Wykresy przedstawiające średnią ilość stanów przetworzonych dla wszystkich metod i z wyszczególnieniem dla różnych kolejności przeszukiwania.



Rysunek 6. Wykresy przedstawiające średnią maksymalną głębkość osiągniętą w trakcie poszukiwania rozwiązywania dla wszystkich metod i z wyszczególnieniem dla różnych kolejności przeszukiwania.

6. Dyskusja

Przedstawione wyniki wskazują, że najmniej razy sprawdzała się metoda DFS, natomiast BFS i A* zależnie od głębokości rozwiązania mają różną skuteczność i nie można wyraźnie określić, która z tych metod jest lepsza.

Metoda BFS przede wszystkim sprawdza się lepiej, gdy głębokość rozwiązania jest mała. Czas trwania procesu przeszukiwania rozwiązania jest wtedy najkrótszy spośród wszystkich metod. Długość rozwiązania zawsze jest równa głębokości na jakiej znajduje się rozwiązanie. Wynika to ze sposobu działania tej metody. BFS przeszukuje wszystkie stany odsuwając się dalej od stanu początkowego, ale nie przechodzi na głębszy poziom dopóki nie sprawdzi wszystkich stanów. Z tego też powodu czas przeszukiwania rośnie proporcjonalnie wraz z głębszym poziomem, gdyż ilość stanów do przetworzenia jest coraz większa. Warto też zauważyć, że przeciwieństwie do metody DFS kolejność przeszukiwania nie miała dużego wpływu. Różnica między stanami odwiedzonymi a przetworzonymi jest niewielka, a ilość stanów rośnie wraz z głębszym poziomem rozwiązań.

Metoda DFS była metodą, która wypadła najgorzej. Czas trwania procesu przeszukiwania jest najdłuższy z pośród wszystkich metod i zarazem przez cały czas oscyluje w podobnym przedziale czasowym, niezależnie od głębokości rozwiązania. Jest to spowodowane sposobem działania tej metody. Algorytm poszukuje rozwiązania w głąb, w przypadku gdy przetworzony

stan nie jest rozwiązaniem zaczyna poszukiwać rozwiązania głębiej, aż gdy dojdzie do dolnej granicy poszukiwania. Dlatego w tej metodzie bardzo ważna jest kolejność przeszukiwań. Gdy przeszukiwanie zaczynało się od przejścia w dół to znacznie szybciej algorytm dochodził do prawidłowego rozwiązania gdyż prawidłowe rozwiązanie znajduję się w prawym dolnym rogu. Często zdarzały się sytuację, gdy algorytm nie mógł odnaleźć rozwiązania w skończonej liczbie kroków. Ilość stanów odwiedzonych i przetworzony jest zbliżona i bardzo szybko rośnie wraz w wzrostem głębokości.

Metoda A* rozwiązywała układankę zdecydowanie szybciej od pozostałych metod gdy głębokość rozwiązania była większa niż 4. Największa przewaga metody A* nad pozostałymi to ilość stanów przetworzonych i odwiedzonych, która jest znacznie mniejsza od innych metod. Jest to spowodowane zastosowaniem kolejki priorytetowej, która najpierw przeszukuje te stany, które mogą prowadzić do rozwiązania. Kolejkę ustala się przy użyciu heurystyk. Różnice miedzy ich skutecznością nie są duże, aczkolwiek heurystyka Hamminga rozwiązuje układankę troszkę szybciej niż heurystyka Manhattan. Gdy mówimy o długości rozwiązania to obie heurystyki zachowują się dokładnie tak samo. Heurystyka hamminga również odwiedza i przetwarza kilka stanów mniej, aczkolwiek różnica miedzy ilością stanów przetworzonych i odwiedzonych jest niewielka. W przeciwieństwie do metod DFS i BFS A* nie przetwarza dużej ilości stanów w poszukiwaniu rozwiązania, tylko stara się z każdym krokiem do niego znacznie zbliżyć i efektem tego jest mała ilość stanów przetworzonych, a także odwiedzonych.

7. Wnioski

Na podstawie naszych badań wynikają następujące wnioski:

- Najszybszą metodą na małych głębokościach jest metoda BFS,
- Metoda BFS odnajduje najkrótsze rozwiązanie,
- W metodzie BFS nie istota jest kolejność przeszukiwania stanów,
- W metodzie DFS bardzo istotna jest kolejność przeszukiwania stanów,
- Proces wyszukiwania rozwiązania jest najdłuższy dla metody DFS,
- Metoda A* odwiedza i przetwarza zdecydowanie mniejszą ilość stanów niż pozostałe metody.

Literatura

[1] T. Oetiker, H. Partl, I. Hyna, E. Schlegl. *Nie za krótkie wprowadzenie do systemu LaTeX2e*, 2007, dostępny online.

[2] *https://pl.wikipedia.org/wiki/Heurystyka*

[3] *http://www.matematyka.wroc.pl/lamiglowki/pietnastka*

[4] *https://en.wikipedia.org/wiki/15_puzzle*

[5] *https://www.ii.uni.wroc.pl/~prz/2011lato/ah/opracowania/szuk_lok.opr.pdf*

[6] *https://pl.wikipedia.org/wiki/Przeszukiwanie_grafu*

[7] *https://pl.wikipedia.org/wiki/Przeszukiwanie_wszerz*

[8] *https://pl.wikipedia.org/wiki/Algorytm_A**

[9] *https://pl.wikipedia.org/wiki/Odleg%C5%82o%C5%9B%C4%87_Hamminga*