Informatyka, studia dzienne, inż I st. semestr VI

Sztuczna inteligencja i systemy ekspertowe 2019/2020

Prowadzący: dr inż. Krzysztof Lichy wtorek, 10:30

Data oddania: _____ Ocena: _____

Radosław Grela 216769
Jakub Wąchała 216914

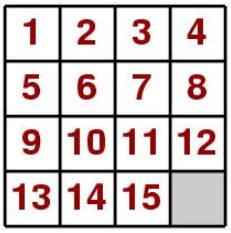
Zadanie 1: Piętnastka

1. Cel

W tym zadaniu należało napisać program, który będzie rozwiązywał układankę pod nazwą *Piętnastka (Fifteen Puzzle)*. Następnie, należało przebadać układy początkowe układanki w odległościach 1-7 od układu wzorcowego (413 układów) korzystając z różnych strategii i przypadków przeszukiwania sąsiedztwa.

2. Wprowadzenie

Piętnastka, potocznie też *Przesuwanka* to prosta gra logiczna, w której zadaniem jest ułożyć 15 ponumerowanych kwadratowych klocków umieszczonych w pudełku 4x4. Pozostałe, 16 miejsce jest puste, co pozwala na poruszanie elementów układanki.[1]



Rysunek 1. Ułożona piętnastka. [1]

W programie przez nas napisanym mamy do wyboru 3 strategie przestrzeni stanów:
— Strategia wszerz BFS (*breadth-first search*)
— Strategia w głąb DFS(*depth-first search*)
— Strategia najpierw najlepszy A* z heurystykami Hamminga i Manhattan
Strategie te są przykładem przeszukiwania drzewa. Jego węzłami są stany, czyli aktualne ułożenia układanki. W części badawczej badaliśmy tylko układanki rozmiarów 4x4, jednak nasz program dopuszcza także niestandardowe rozmiary.

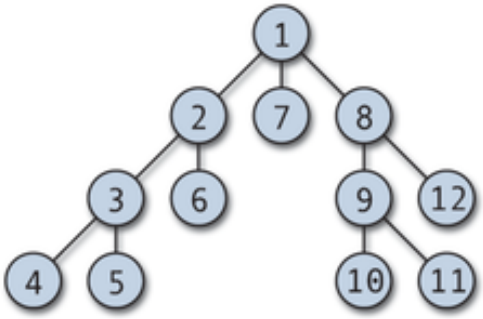
2.1. DFS

Badanie grafu strategią DFS polega na przejściu wszystkich krawędzi wychodzących z podanego wierzchołka. Jest to algorytm rekurencyjny. Kolejność przechodzenia po drzewie przedstawiona jest na poniższym rysunku:

Gdy rozwiązanie zostanie znalezione, wystarczy wrócić rekurencyjnie do rodzica.

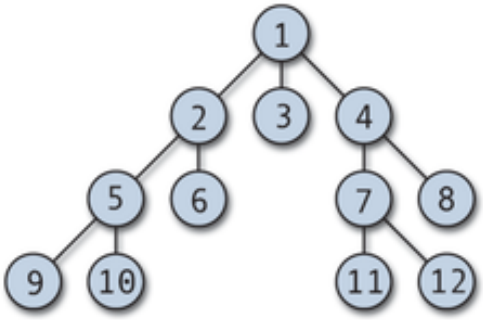
2.2. BFS

Algorytm BFS polega na przejściu przez wszystkich sąsiadów danego wierzchołka. Następnie, należy kontynuować czynność dalej, dopóki nie od-



Rysunek 2. Kolejność przechodzenia w algorytmie DFS. [2]

wiedzimy wszystkich sąsiadów sąsiadów. Kolejność przechodzenia po drzewie jest następująca:



Rysunek 3. Kolejność przechodzenia w algorytmie BFS. [3]

2.3. A*

A* to kolejna strategia przeszukiwania grafu. Bazuje ona na funkcji

$$f(x) = g(x) + h(x) \tag{1}$$

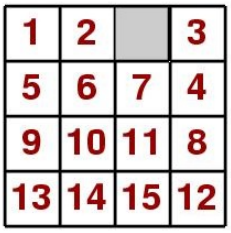
gdzie $g(x)$ oznacza głębokość, a $h(x)$ to wartość odpowiedniej funkcji miary błędu. W naszym programie wykorzystujemy następujące dwie metryki:

2.3.1. Metryka Hamminga

W metryce Hamminga jako wynik funkcji $h(x)$ podawana jest ilość klocków, która nie znajduje się na swojej pozycji. Przykładowo dla układanki na rysunku (4) wynik takiej funkcji będzie równy 4 (elementy 3, 4, 8 i 12 nie znajdują się na swoich pozycjach).

2.3.2. Metryka Manhattan

W tej metryce wynikiem funkcji $h(x)$ jest suma wartości bezwzględnych różnic współrzędnych między punktem, w którym klocek powinien się znaleźć, a punktem, w którym jest obecnie. Dla rysunku (4) wynik to 8 (4 + 1 + 1 + 1 + 1).



Rysunek 4. Przykładowa błędnie ułożona układanka.

3. Opis implementacji

Napisany przez nas program jest aplikacją konsolową napisaną w języku Java. Jako parametry programu należy podać 5 argumentów:

1. strategia (dfs, bfs, astr)
2. porządek (jeżeli jest to strategia DFS lub BFS) lub heurystyka (dla metody A*).
3. ścieżka do pliku z zadaną układanką do ułożenia
4. ścieżka do pliku, w którym zostanie zapisane rozwiązanie
5. ścieżka do pliku, w którym zostaną zapisane dodatkowe informacje dot. przeprowadzonego procesu

Poniżej przedstawiamy krótki opis, jak zostały zaimplementowane poszczególne strategie.

3.1. DFS

Do przechowywania stanów nie używamy specjalnej struktury danych, lecz za pomocą rekurencji program wie, który węzeł musi odwiedzić jako następny. Zatem przechowywanie stanów jest możliwe dzięki odkładaniu wywołań funkcji na stosie.

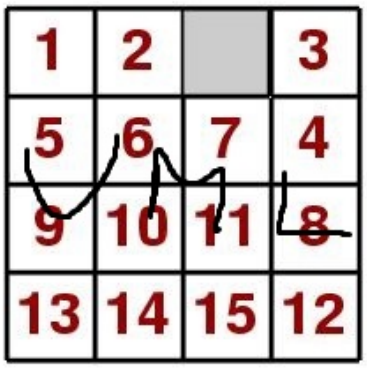
3.2. BFS

elo here do napisania

3.3. A*

W przypadku algorytmów A* dane są przetrzymywane na liście (ArrayList), która spełnia zadania kolejki priorytetowej. W momencie dodania nowego elementu na liście, jest ona sortowana zgodnie z wynikiem funkcji (1). Na początku listy znajdują się ”najgorsze” elementy - te, których wartość funkcji jest największa, a na końcu elementy, które mają wartość tej funkcji najmniejszą.

Na poniższym rysunku przedstawiony jest diagram UML naszego programu.



Rysunek 5. Diagram UML.

4. Materiały i metody

Do przeprowadzenia badań nad utworzoną przez nas aplikacją użyliśmy kilku programów znajdujących się na stronie przedmiotu na Wikampie.

1. Program do utworzenia 413 różnych układów piętnastki z głębokością 1-7
2. Skrypt uruchamiający naszą aplikację z każdym algorytmem oraz ewentualną heurystyką i kolejnością
3. Skrypt Podsumowujący uzyskane wyniki z poprzedniego skryptu

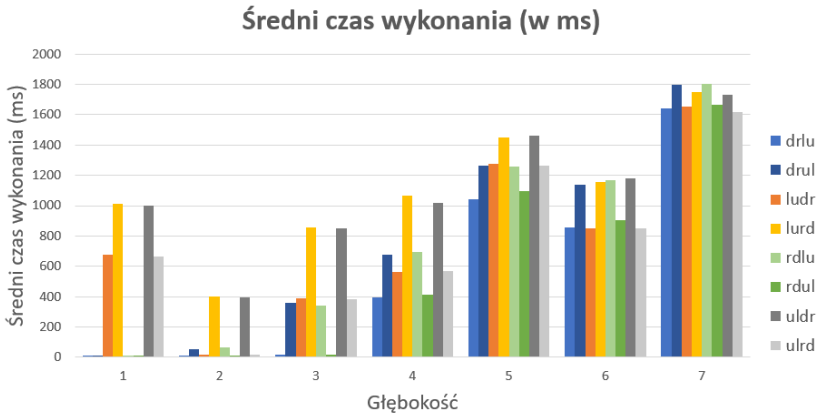
Otrzymane wyniki przedstawiliśmy na wykresach w następnej sekcji.

5. Wyniki

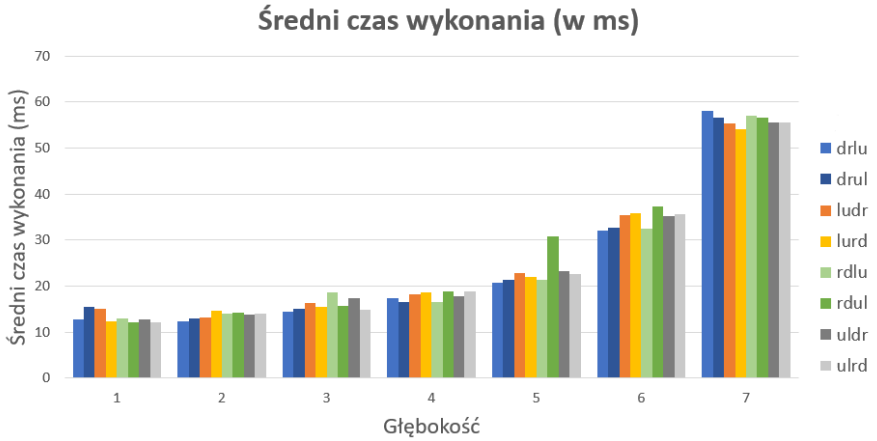
Na poniższych wykresach przedstawiliśmy średnie wartości:

— Czasu wykonywania
— Maksymalnej głębokości
— Ilości przetworzonych węzłów
— Ilości odwiedzonych węzłów
— Długości rozwiązania

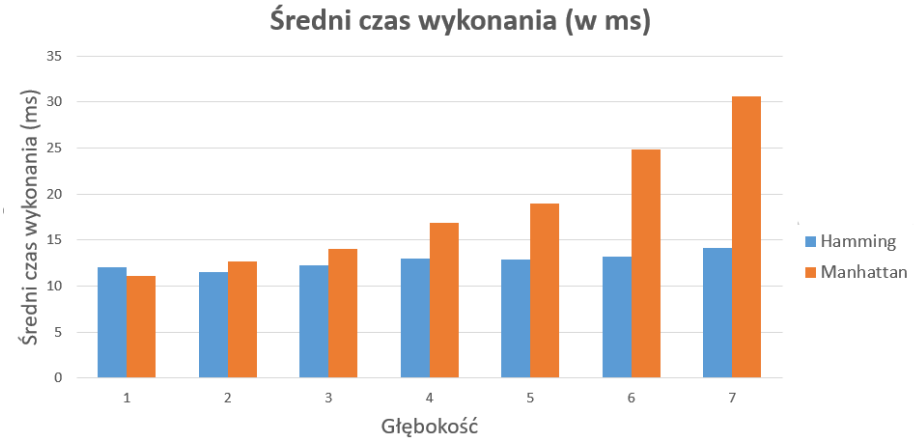
Każdy wykres został wykonany dla każdej strategii oraz ich heurystyk.



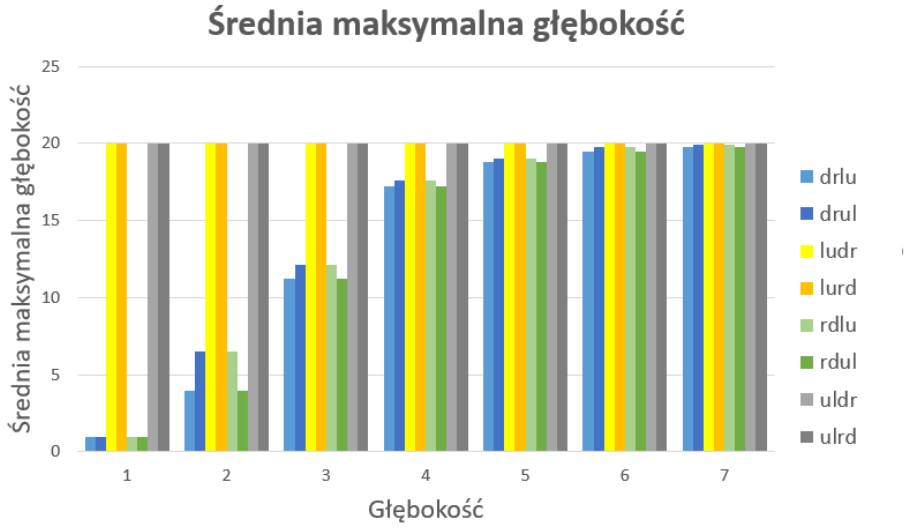
Rysunek 6. Średni czas wykonania dla algorytmu DFS



Rysunek 7. Średni czas wykonania dla algorytmu BFS



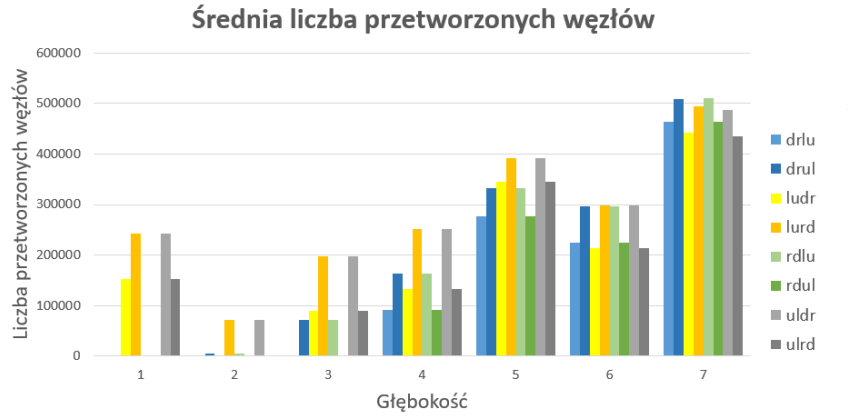
Rysunek 8. Średni czas wykonania dla algorytmu A*



Rysunek 9. Średnia maksymalna głębokość dla algorytmu DFS

Rysunek 10. Średnia maksymalna głębokość dla algorytmu BFS

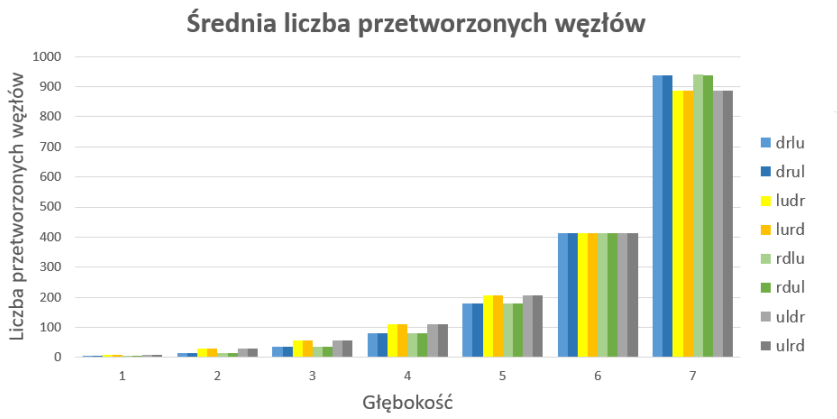
Rysunek 11. Średnia maksymalna głębokość dla algorytmu A*



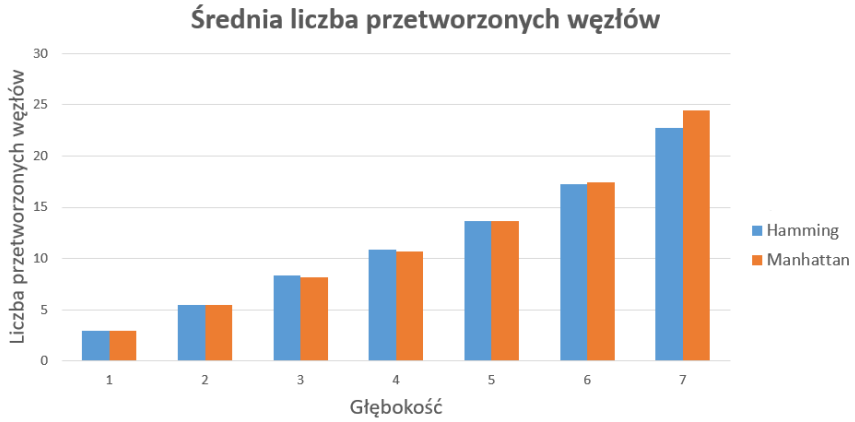
Rysunek 12. Średnia ilość przetworzonych węzłów dla algorytmu DFS

6. Dyskusja

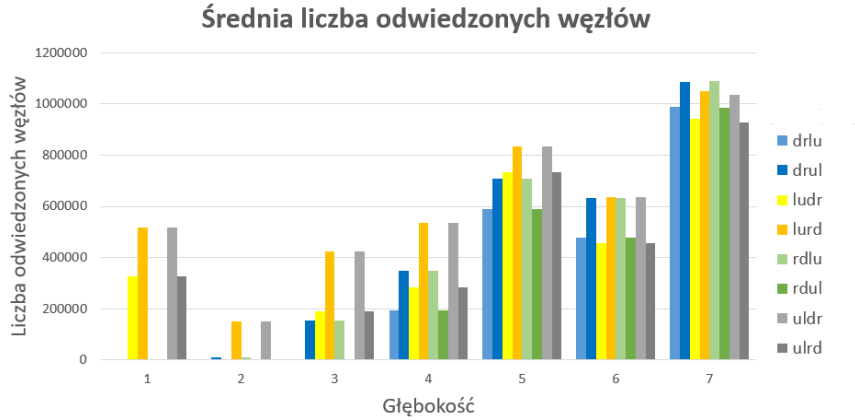
Sekcja ta powinna zawierać dokładną interpretację uzyskanych wyników eksperymentów wraz ze szczegółowymi wnioskami z nich płynącymi. Najcenniejsze są, rzecz jasna, wnioski o charakterze uniwersalnym, które mogą być istotne przy innych, podobnych zadaniach. Należy również omówić i wyjaśnić wszystkie napotkane problemy (jeśli takie były). Każdy wniosek powinien mieć poparcie we wcześniej przeprowadzonych eksperymentach (odwołania



Rysunek 13. Średnia ilość przetworzonych węzłów dla algorytmu BFS



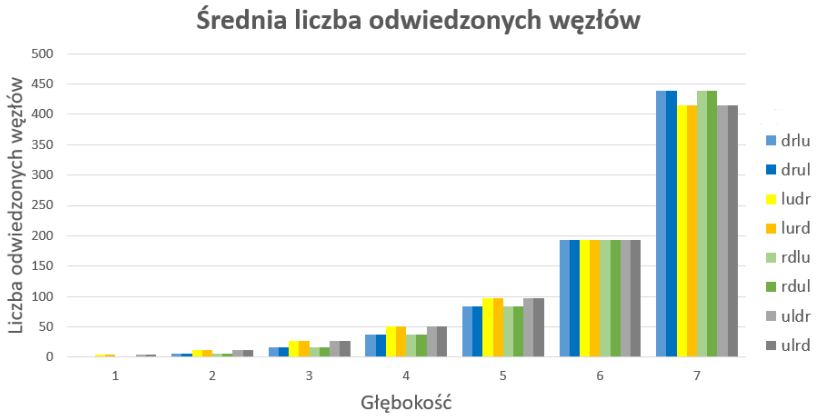
Rysunek 14. Średnia ilość przetworzonych węzłów dla algorytmu A*



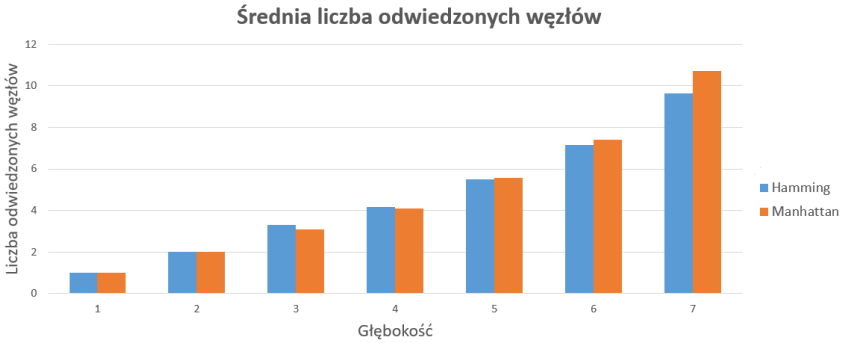
Rysunek 15. Średnia ilość odwiedzonych węzłów dla algorytmu DFS

do konkretnych wyników). Jest to jedna z najważniejszych sekcji tego sprawozdania, gdyż prezentuje poziom zrozumienia badanego problemu.

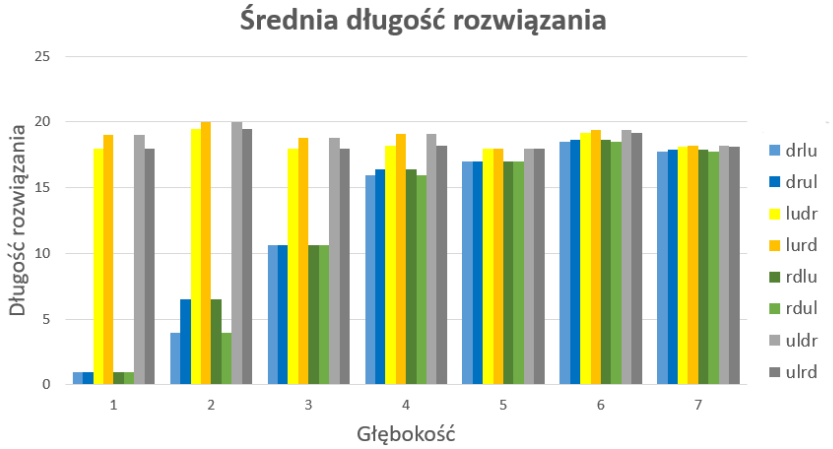
7. Wnioski



Rysunek 16. Średnia ilość odwiedzonych węzłów dla algorytmu BFS

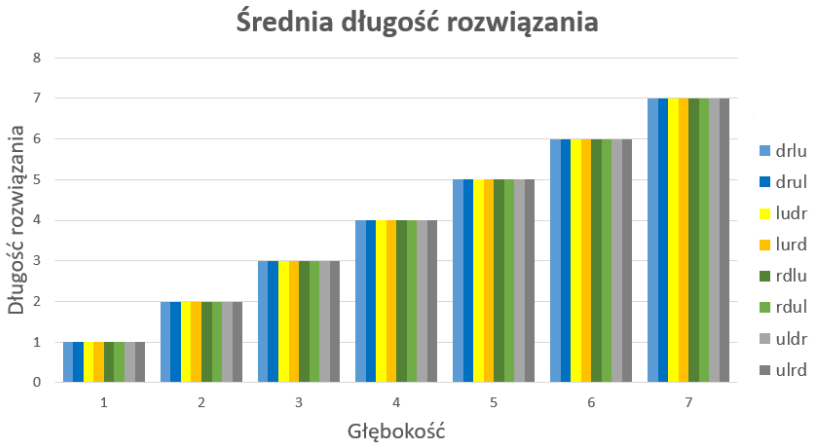


Rysunek 17. Średnia ilość odwiedzonych węzłów dla algorytmu A*

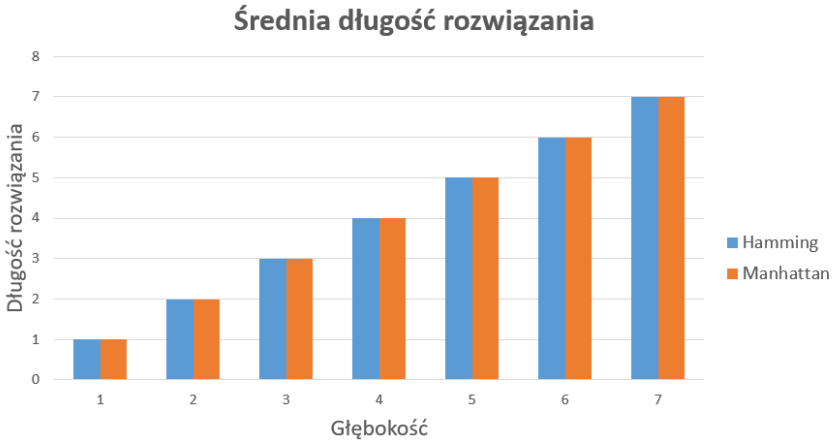


Rysunek 18. Średnia długość rozwiązania dla algorytmu DFS

W tej, przedostatniej, sekcji należy zamieścić podsumowanie najważniejszych wniosków z sekcji poprzedniej. Najlepiej jest je po prostu wypunktować. Znów, tak jak poprzednio, najistotniejsze są wnioski o charakterze uniwersalnym.



Rysunek 19. Średnia długość rozwiązania dla algorytmu BFS



Rysunek 20. Średnia długość rozwiązania dla algorytmu A*

Literatura

[1] *http://www.math.ubc.ca/ cass/courses/m308-02b/projects/grant/fifteen.html* [dostęp 17.03.2020]
[2] *https://pl.wikipedia.org/wiki/Przeszukiwanie_w_głąb* [dostęp 17.03.2020]
[3] *https://pl.wikipedia.org/wiki/Przeszukiwanie_wszerz* [dostęp 17.03.2020]
[4] *https://en.wikipedia.org/wiki/A*_search_algorithm* [dostęp 17.03.2020]