Informatyka, studia dzienne, inż I st. semestr IV

Sztuczna inteligencja i systemy ekspertowe 2024/2025

Prowadzący: dr. inż. Krzysztof Lichy wtorek, 12:15

Data oddania: _____ Ocena: _____

Remigiusz Tomecki 251652
Klim Hudzenko 253833

Zadanie nr. 1: Piętnastka

1. Cel

Celem zadania było stworzenie programu rozwiązującego Piętnastkę (znaną grę logiczną) oraz analiza wyników rozwiązań dla różnych strategii.

2. Wprowadzenie

Gra Piętnastka to plansza 15 kafelków w układzie 4x4 (jedno pole pozostaje puste, co umozliwia ruchy). Polega on na przesuwaniu kafelków w taki sposób, aby kafelki zostały ułożone w kolejności od 1 do 15 (kafelek nr. 1 powinien byc w lewym gornym rogu, natomiast puste pole - kafelek nr. 0 w prawym dolnym rogu.) W celu znajdowania rozwiązań zaimplementowaliśmy trzy strategie wyszukiwania rozwiązań:

- **BFS** (algorytm wszerz): Przegląda wszystkie stany w danym poziomie po czym dopiero przechodzi do nastepnego poziomu.

- **DFS** (algorytm w głąb): Przeglada stany w głąb jednej gałęzi, aż do osiągnięcia maksymalnej głębokości, po czym zawraca.

- **A*** (algorytm heurystyczny): Używa heurystyki do wyboru najbardziej obiecującego stanu. Korzysta z funkcji oceny:

$$f(n) = g(n) + h(n)$$

Gdzie g(n) to koszt dotarcia do stanu, a h(n) to heurystyka

Rozwiązując zadanie użyliśmy dwóch heurystyk:
- **Manhattan** - liczy sumę odległości klocków od ich pozycji docelowych
- **Hamming** - zlicza kafelki będące nie na swoich miejscach

3. Opis implementacji

Program został napisany w języku Python. Implementuje on trzy strategie przeszukiwania stanów planszy Piętnastki: BFS, DFS i A*. Program przyjmuje parametry takie jak: - Strategia rozwiązania - Dla BFS i DFS kolejnosc ruchow, natomiast dla A* heurystyke - Plik z planszą do rozwiązania - Plik wynikowy z rozwiązaniem - Plik wynikowy ze statystykami

Zaimplementowane algorytmy:

- **BFS**: Przegląda wszystkie stany w danym poziomie po czym dopiero przechodzi do nastepnego poziomu. Używa kolejki FIFO, aby sprawdzać stany w odpowiedniej kolejności. Algorytm BFS gwarantuje znalezienie najkrótszego rozwiązania, ponieważ sprawdza wszystkie możliwości na każdym poziomie.

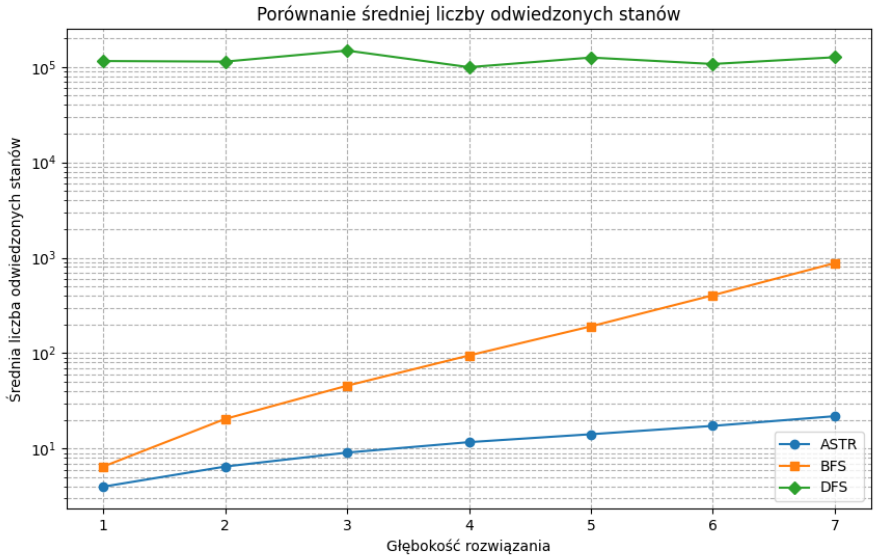
- **DFS**: Przeglada stany w głąb jednej gałęzi, aż do osiągnięcia maksymalnej głębokości, po czym zawraca. Używa stosu LIFO do przechowywania stanów. Nie gwarantuje najkrótszego rozwiązania.

- **A***: Używa heurystyki do wyboru najbardziej obiecującego stanu. Korzystając z funkcji oceny. Używa kolejki priorytetowej, gdzie priorytet to dotychczasowy koszt + oszacowanie do celu. Gwarantuje znalezienie rozwiazania. Dwie heurystyki w naszej implementacji to *Hamming* i *Manhattan*.

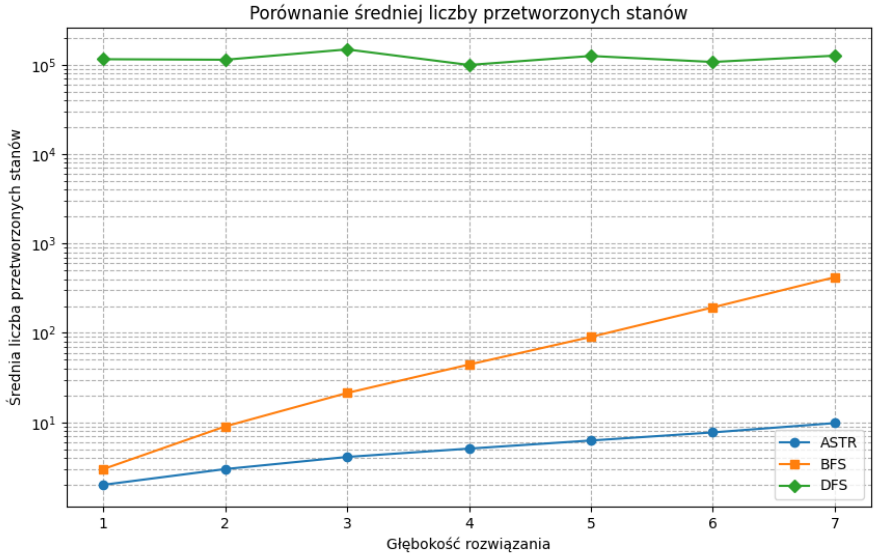
4. Materiały i metody

Za pomocą skryptów dostępnych na platformie Wikamp wygenerowaliśmy 413 układów planszy oraz uruchomiliśmy na nich nasz program. Następnie otrzymane wyniki skonsolidowaliśmy i umieściliśmy na wykresach.

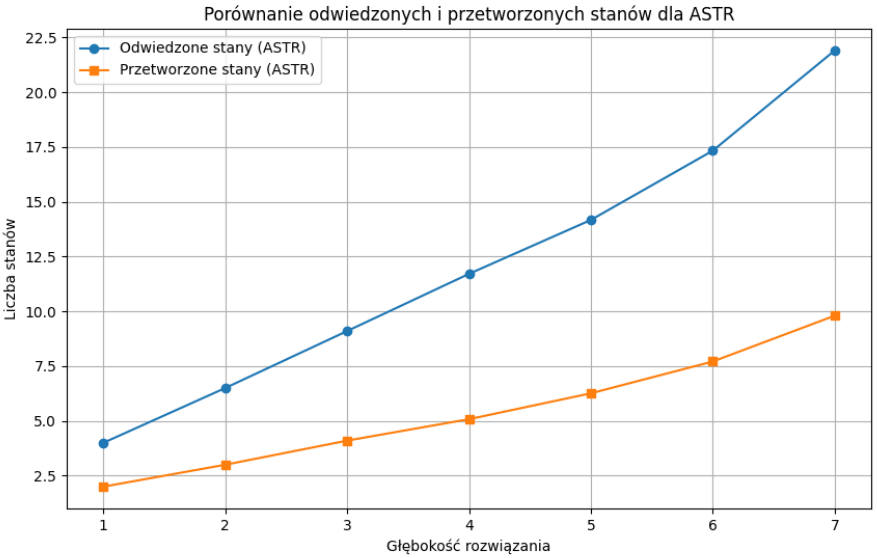
5. Wyniki



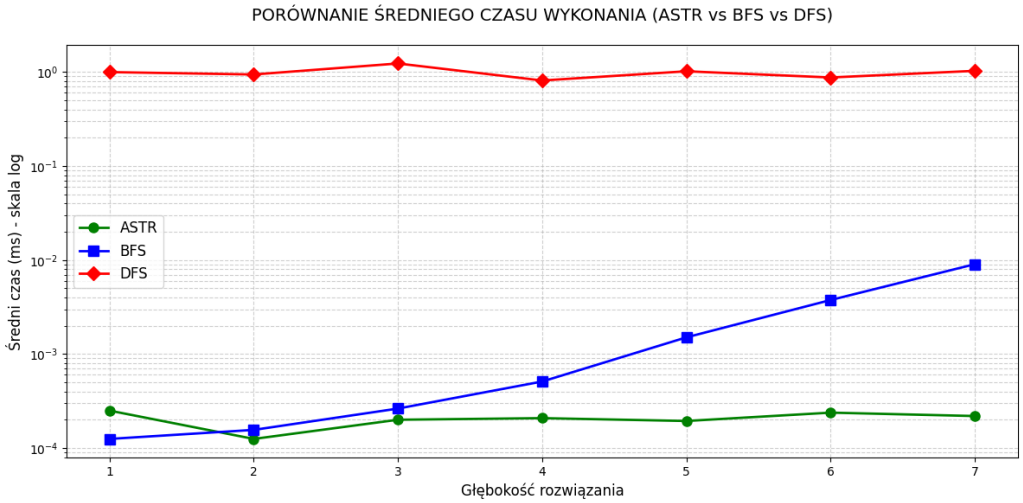
Rysunek 1. Wykres porównujący średnią liczbę odwiedzonych stanów dla poszczególnych strategii



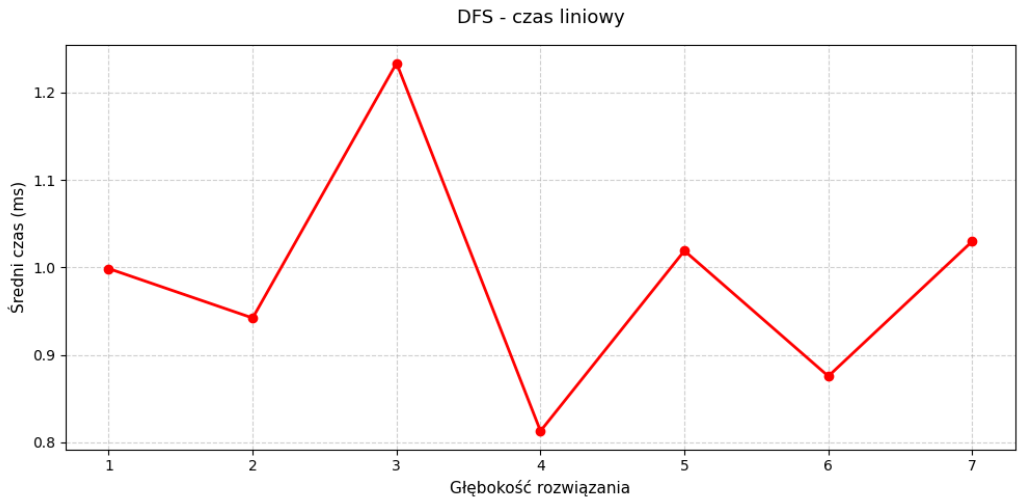
Rysunek 2. Wykres porównujący średnią liczbę przetworzonych stanów dla poszczególnych strategii



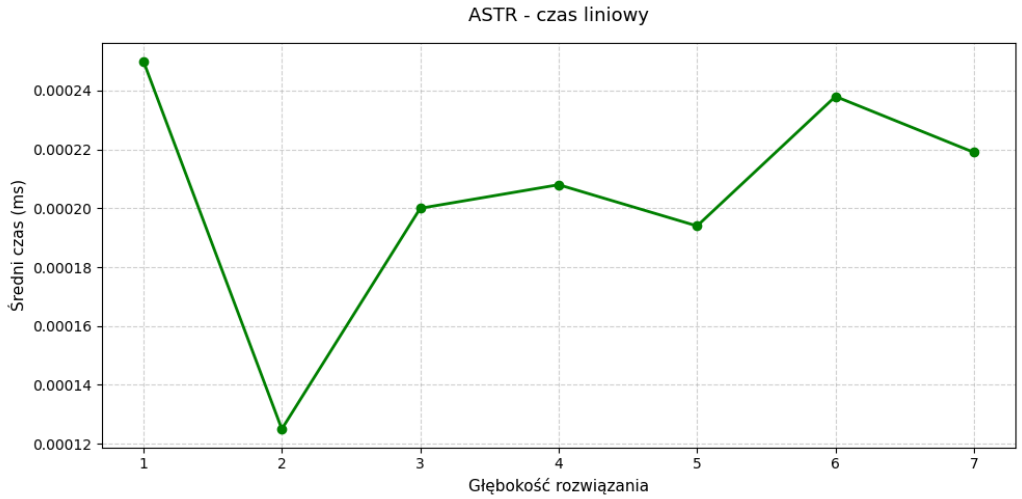
Rysunek 3. Wykres porównujący liczbę odwiedzonych i przetworzonych stanów dla strategii A*



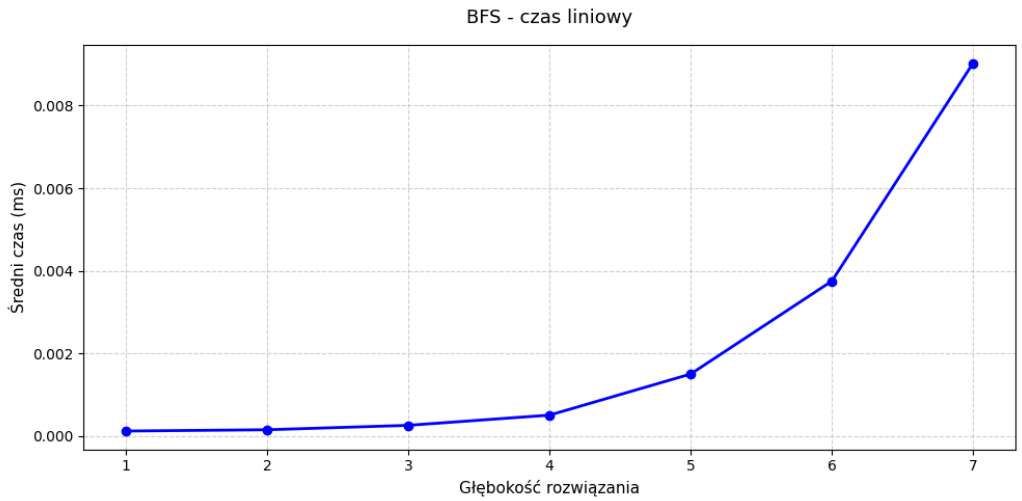
Rysunek 4. Wykres porównujący średni czas wykonywania dla poszczególnych strategii



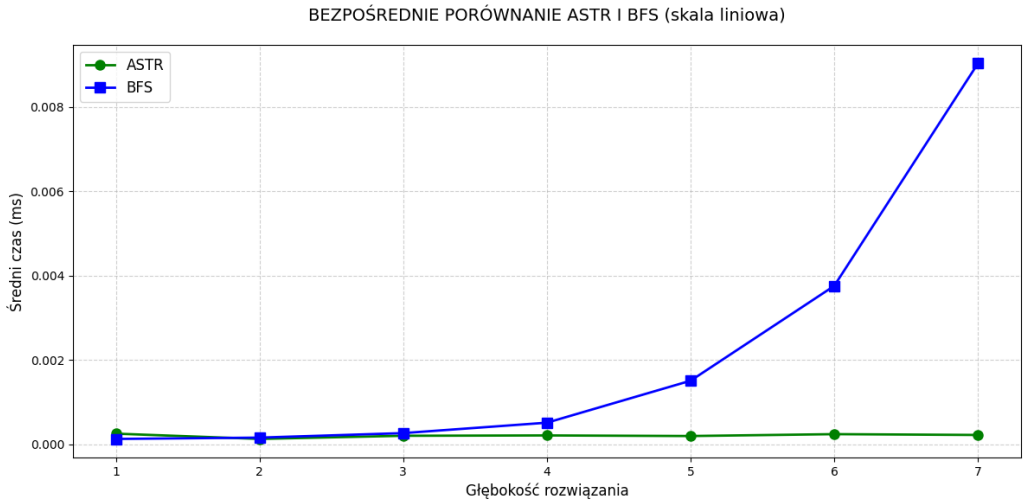
Rysunek 5. Wykres ukazujący średni czas rozwiązywania układanek dla poszczególnych głębokości rozwiązania dla strategii DFS



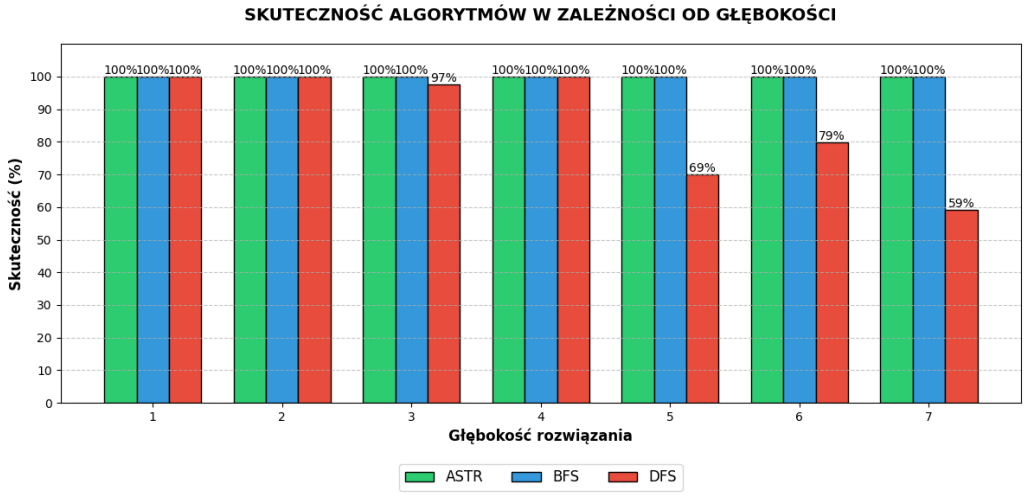
Rysunek 6. Wykres ukazujący średni czas rozwiązywania układanek dla poszczególnych głębokości rozwiązania dla strategii A*



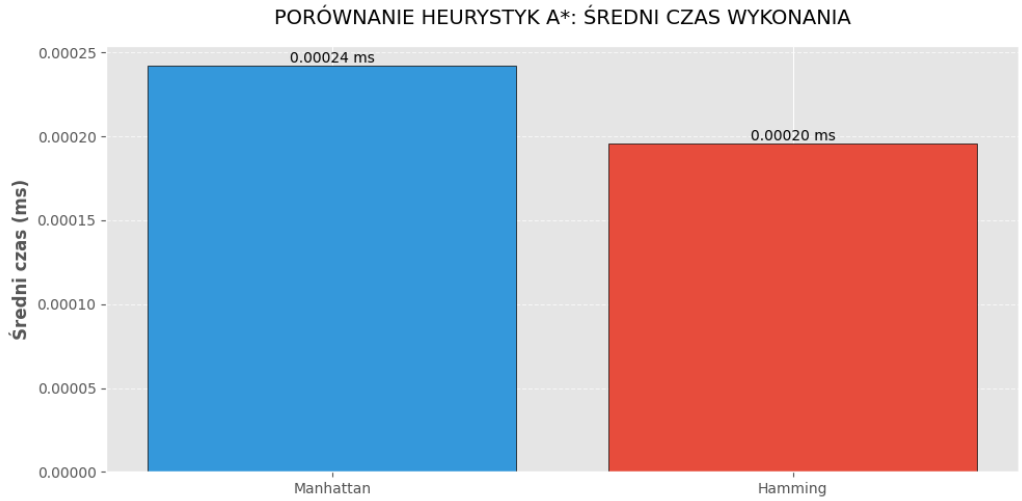
Rysunek 7. Wykres ukazujący średni czas rozwiązywania układanek dla poszczególnych głębokości rozwiązania dla strategii BFS



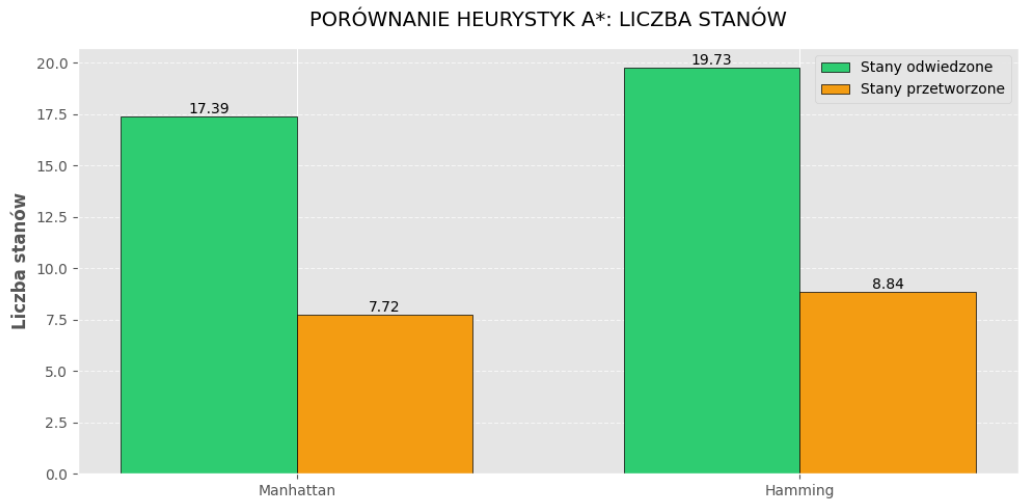
Rysunek 8. Wykres porównujący średni czas rozwiązywania układanek dla poszczególnych głębokości rozwiązania BFS i A*



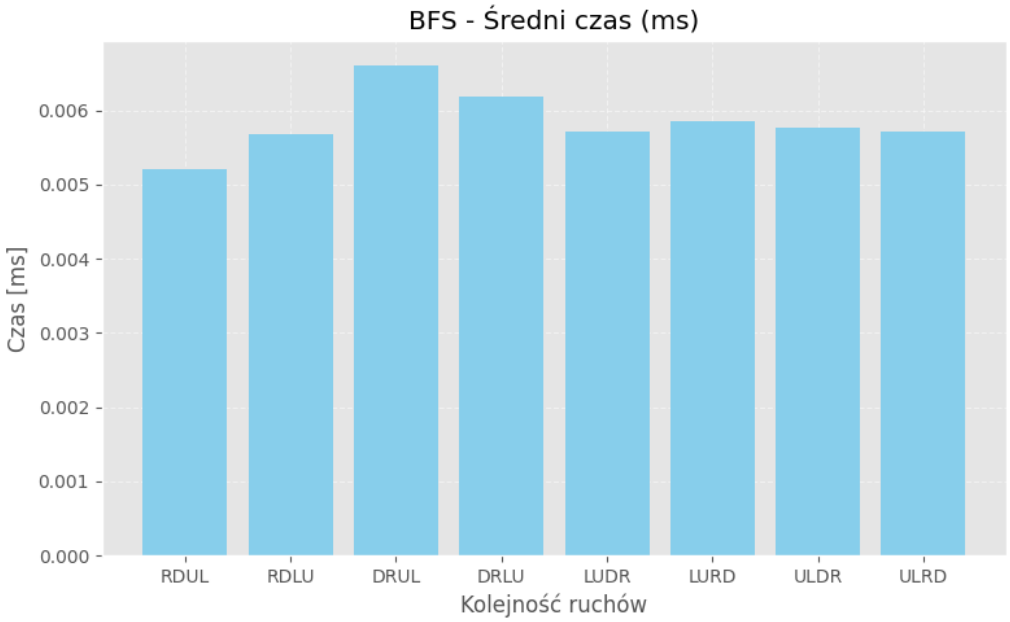
Rysunek 9. Wykres ukazujący skuteczność poszczególnych algorytmów w zależności od głębokości



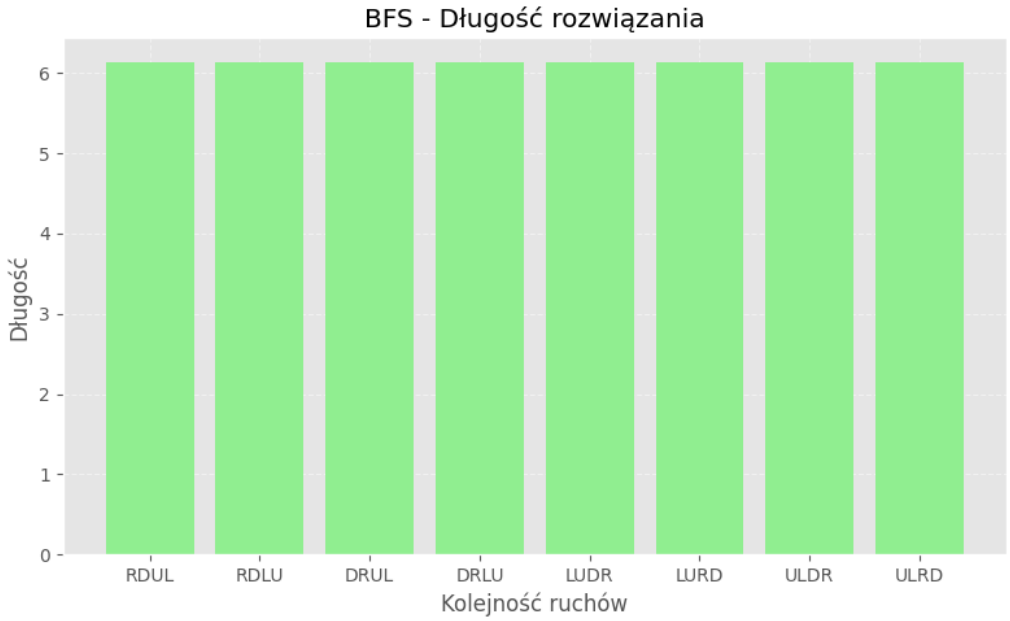
Rysunek 10. Wykres porównujący średni czas wykonywania dla poszczególnych heurystyk strategii A*



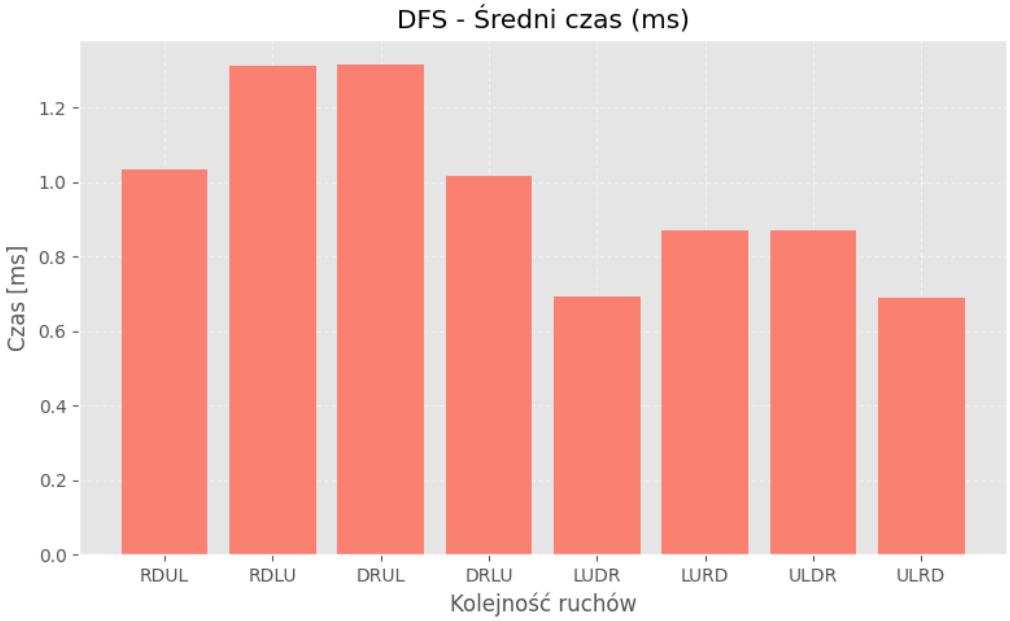
Rysunek 11. Wykres porównujący odwiedzone i przetworzone stany dla poszczególnych heurystyk strategii A*



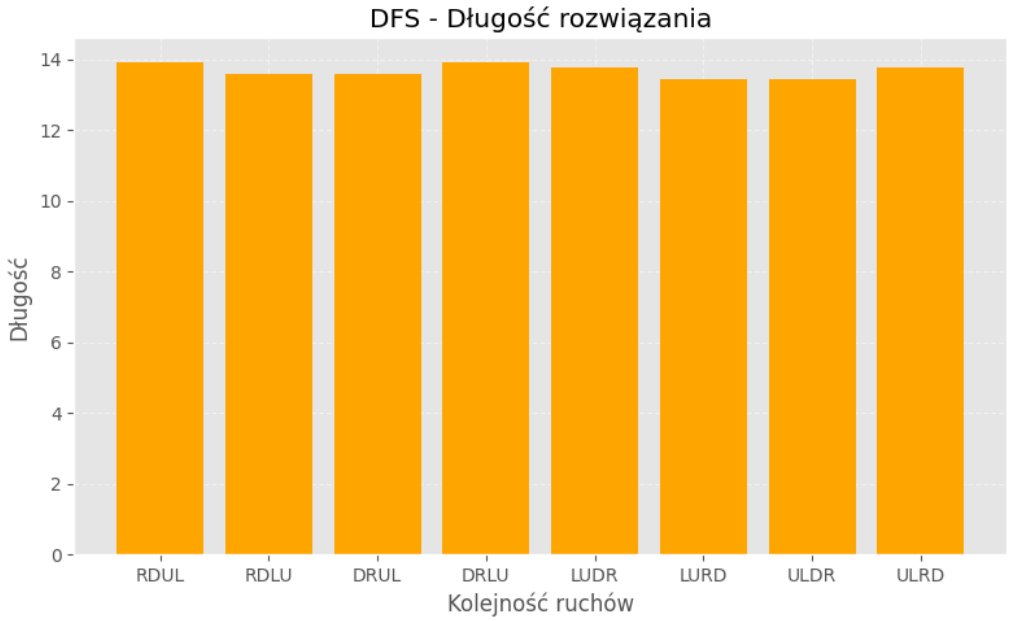
Rysunek 12. Wykres porównujący średni czas wykonywania algorytmu BFS dla poszczególnych kolejności ruchów



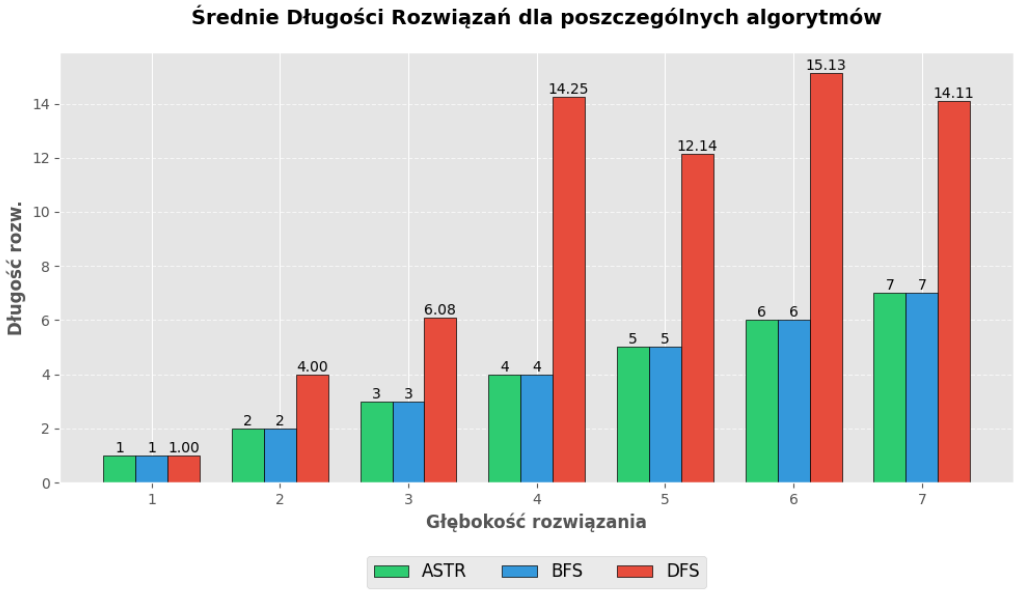
Rysunek 13. Wykres porównujący średni długość rozwiązania algorytmu BFS dla poszczególnych kolejności ruchów



Rysunek 14. Wykres porównujący średni czas wykonywania algorytmu DFS dla poszczególnych kolejności ruchów



Rysunek 15. Wykres porównujący średni długość rozwiązania algorytmu DFS dla poszczególnych kolejności ruchów



Rysunek 16. Wykres porównujący średnią długość rozwiązania dla poszczególnych algorytmów w zależności od głebokości rozwiązania

6. Dyskusja

- **BFS**:

Algorytm najlepiej radził sobie z planszami, które do rozwiązania potrzebowały mało ruchów, jako że jego średni czas wykonywania rósł wykładniczo w stosunku do głębokości rozwiązania (Rys. 7).
Zmiana kolejności wykonywania ruchów nie miała żadnego wpływu na długość rozwiązania (Rys. 13), natomiast miała niewielki wpływ na średni czas wykonywania na korzyść kolejności RDUL (Rys. 12), jednakże dla tak małych głębokości rozwiązań prawdopodobnie była to kwestia przypadku.
Ilość odwiedzonych i przetworzonych stanów rosła logarytmicznie w stosunku do głębokości rozwiązania (Rys. 1 i Rys. 2).

- **DFS**:

Algorytm poradził sobie najgorzej ze wszystkich, działał najdłużej (Rys. 4), dla plansz o większej ilości ruchów nie zawsze udawało mu się zwrócić rozwiązania (Rys. 9). Algorytm dawał też najdłuższe rozwiązania ze wszystkich (Rys. 17).
Średni czas wykonywania algorytmu wydaje się nie mieć związku z głębokością rozwiązania (Rys. 5).
Kolejność wykonywania ruchów miała umiarkowany wpływ na średni czas wykonywania (Rys. 14), lecz niewieki na długości rozwiązania (Rys. 15).
Algorytm odwiedził i przetworzył też najwięcej stanów ze wszystkich strategii (Rys. 1 i Rys. 2).

- **A***:

Algorytm dla głębokości większej od 1 był najszybszy ze wszystkich (Rys. 4).
Algorytm odwiedził (Rys. 1) i przetworzył (Rys. 2) najmniej stanów ze wszystkich, a liczba odwiedzonych stanów była około dwukrotnie większa od liczby przetworzonych stanów dla każdej głębokości rozwiązania (Rys. 3).
Algorytm A* był w stu procentach skuteczny (Rys. 9).
Wybór heurystyk miał minimalny wpływ na średni czas wykonania (Rys. 10) oraz niewielki na odwiedzone i przetworzone stany (Rys. 11), dla heurystyki Hamminga średni czas był krótszy pomimo odwiedzenie i przetworzenia większej liczby stanów.

7. Wnioski

Na podstawie zgromadzonych wyników, zauważamy, że algorytmem który poradził sobie najlepiej ze znajdywaniem rozwiązań Piętnastki był A*, który rozwiązywał plansze najszybciej oraz nie pomylił się ani razu. Najgorzej natomiast wypadł algorytm DFS, który potrzebował najwięcej czasu, jego rozwiązania zazwyczaj potrzebowały znacznie więcej ruchów, a nawet nie zawsze algorytm zwracał rozwiązanie. Algorytm BFS był w stanie znaleźć najbardziej optymalne rozwiązania, jednakże jego średni czas rozwiązywania układanek rósł wykładniczo do liczby ruchów potrzebnych do rozwiązania.

Literatura

[1] Wikipedia: `https://pl.wikipedia.org`
[2] Wikamp: `https://ftims.edu.p.lodz.pl/course/view.php?id=16`
[3] BFS i DFS: `https://eduinf.waw.pl/inf/utils/011_2011/0105.php`
[4] Astr: `https://elektron.elka.pw.edu.pl/~jarabas/ALHE/notatki3.pdf`