Informatyka, studia dzienne, inż I st. semestr IV

Sztuczna inteligencja i systemy ekspertowe 2023/2024

Prowadzący: Dr. Krzysztof Lichy poniedziałek, 12:15

Data oddania: _____ Ocena: _____

Mateusz Giełczyński 247662
Jakub Kubiś 247712

Zadanie 1: Piętnastka

1. Cel

Celem zadania było napisanie programu znajdującego rozwiązanie układanki logicznej zwanej "Pietnastką" oraz zbadanie zasad działania algorytmów przeszukujących grafy.

2. Wprowadzenie

Piętnastka to układanka logiczna składająca się z kwadratowej planszy podzielonej na 16 kwadratów. Zawierających cyfry od 1 do 15 oraz jednym miejscem które zawsze pozostaje puste co pozwala na przesuwanie sąsiednich kafelków, celem piętnastki jest uporządkowanie kafelków tak aby uzyskać układ liczbowy gdzie liczby ułożone są w porządku rosnącym, a ostatnie pole pozostaje puste.

Piętnastkę można zinterpretować jako graf, gdzie danym węzłem jest stan planszy. Natomiast krawędzie łączące węzły reprezentują możliwe ruchy pomiędzy stanami. Korzeniem w takim grafie będzie początkowy nierozwiązany stan układanki.

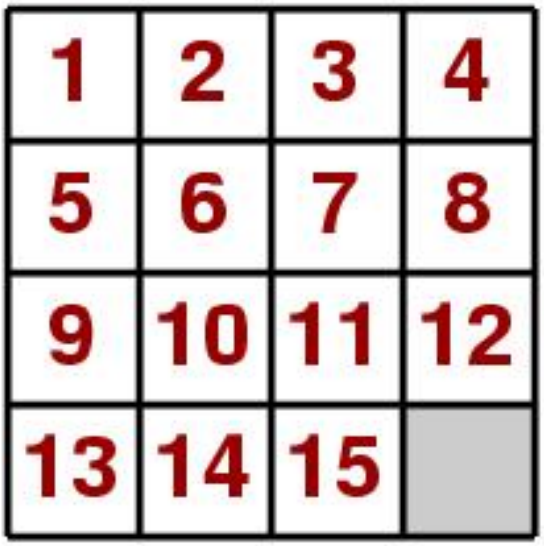
Inepretacja piętnastki jako grafu pozwala nam stosować metody przeszukiwania grafów do znajdywania rozwiązań układnaki. Zaimplementowane przez nas metody to:

— BFS (breadth-first search) - przeszukiwanie wszerz.
Metoda polega na przeszukaniu wszystkich węzłów na jednej głębokości przed zejściem na niższy poziom.

— DFS (depth-first search) - przeszukiwanie wgłąb.

Metoda polega na przeszukiwaniu w głąb, oznacza to że zanim zbada kolejną krawedź pierwszego węzła dopiero gdy zbada wszystkie krawędzie kolejnego węzła.

— A* (A-star).
Metoda polega na wybieraniu najbardziej obiecujacego węzła na podstawie heurystyki.
  — Heurystyka Hamminga
  Heurystyka polega na zliczeniu ilości kafelków które nie znajdują się na odpowiednim miejscu.
  — Heurystyka Manhattana
  Heurystyka polega na zliczeniu sumy ruchów, które potrzebuje każdy z kafelków, żeby znaleźć się na odpowiednim miejscu.



Rysunek 1. Rozwiązana piętnastka

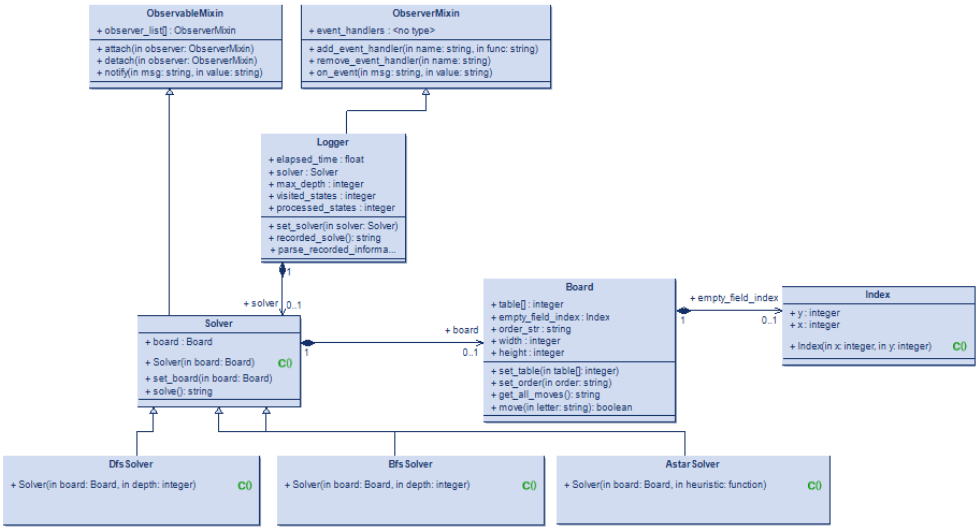
3. Opis implementacji

Program został stworzony w języku Python 3.

Klasa Board odpowiada za przechowywanie stanu planszy, związanych z nią informacji oraz przemieszczanie elementów planszy.

Klasa Solver oraz klasy po niej dziedziczące zawierają logikę odpowiedzalną za rozwiązanie układanki daną metodyką.

Klasa Logger odpowiada za zbieraniu dodatkowych informacji, może być połaczona z klasami rozwiązującymi korzystając ze wzorca projektowego obserwatora zaimplemetowanego poprzez klasy ObserverMixin oraz ObservableMixin.



Rysunek 2. Diagram UML

4. Materiały i metody

Do przeprowadzenia eksperymentów użyto narzędzi udostępnionych na platformie WIKAMP, zostały wykorzystane przy:
— Generowaniu stanów układanek do rozwiązania
— Uruchamianiu wykonanego programu dla wygenerowanych układanek
— Weryfikacji poprawności rozwiązań
— Podsumowaniu wyników
Badania zostały przeprowadzone dla wszystkich metod oraz wszystkich możliwych stanów dla głębokości od 1 do 7. Natomiast dla głębokości 15 zbadano przebiegi dla A*(obie heurystki) oraz dla DFS(Kolejność RDUL) dla 20 losowych stanów.

5. Wyniki

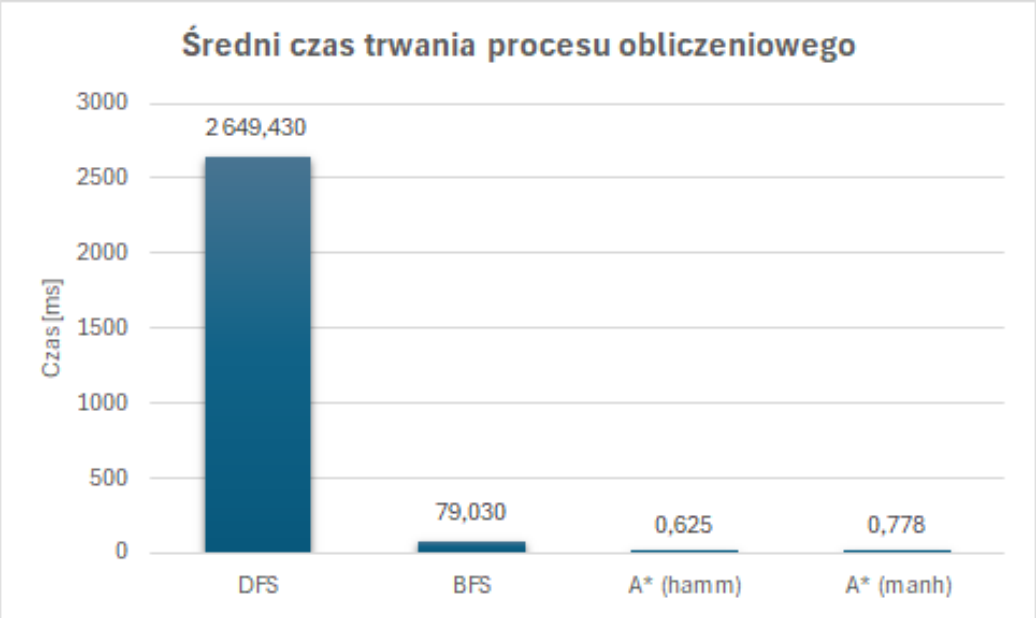
Poniżej przedstawiono tabelę zawierające podsumowanie danych zebranych podczas rozwiązanywania przez program układów dla kolejno:
— Głębokości 1-7
— Głębokości 15
Oraz wygenerowane na podstawie tych tabel wykresy.

Tabela 1. Średnie wyników dla poszczególnych metod z głębokością 1-7.

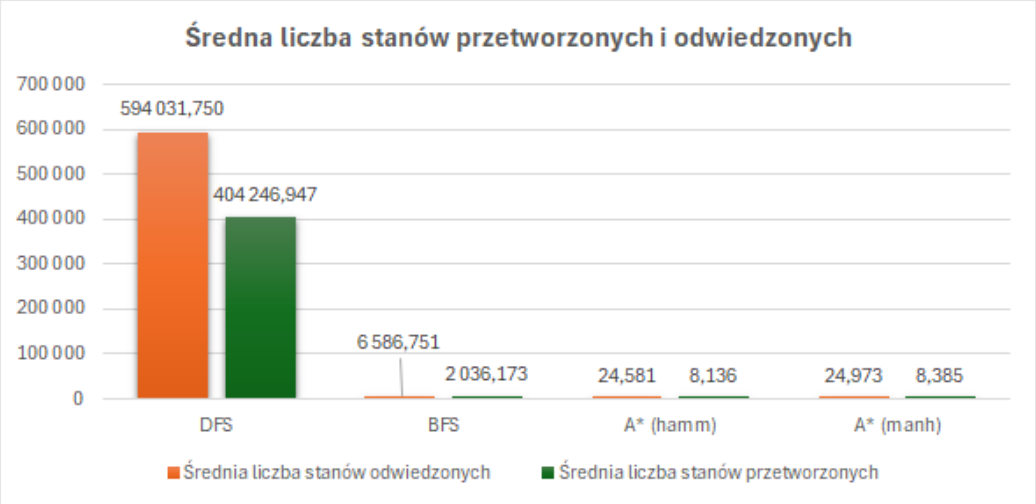
Metoda	Długość rozwiązania	Odwiedzone stany	Przetworzone stany	Maksymalna	Czas[ms]
DFS	17,981	594031,750	404246,947	19,544	2649,430
BFS	6,131	6586,751	2036,173	6,131	79,030
A* (hamm)	6,131	24,581	8,136	6,131	0,625
A* (manh)	6,131	24,973	8,385	6,131	0,778

Tabela 2. Średnie wyników dla poszczególnych metod dla układanek z głębokością 15.

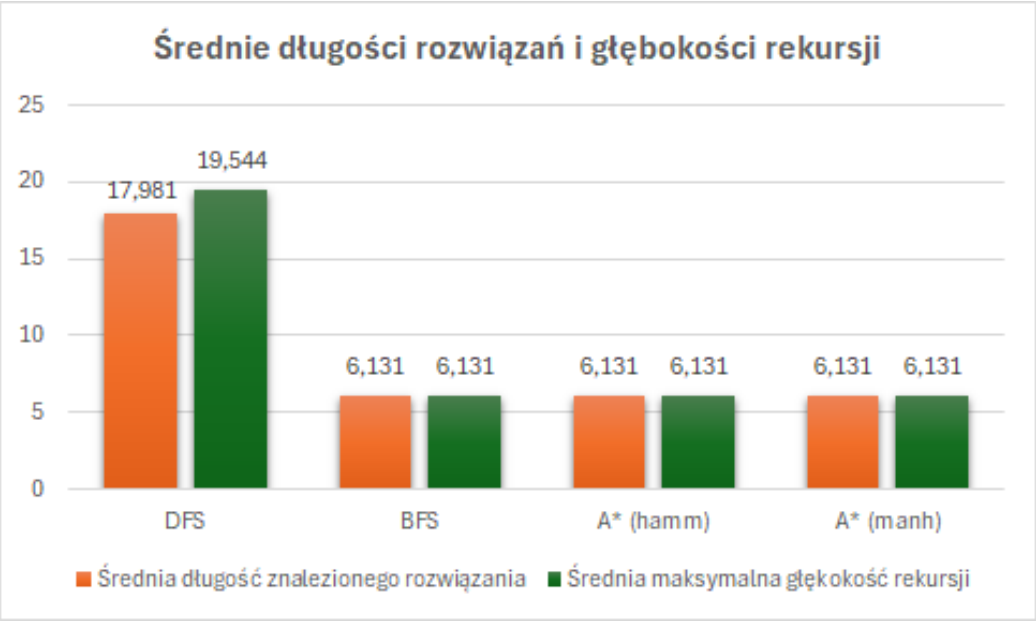
Metoda	Długość rozwiązania	Odwiedzone stany	Przetworzone stany	Maksymalna	Czas
DFS (RDUL)	17,700	5575634,700	3794514,150	20,000	25442,442
A* (hamm)	15,000	530,700	171,050	15,000	10,508
A* (manh)	15,000	238,250	78,850	15,050	6,721



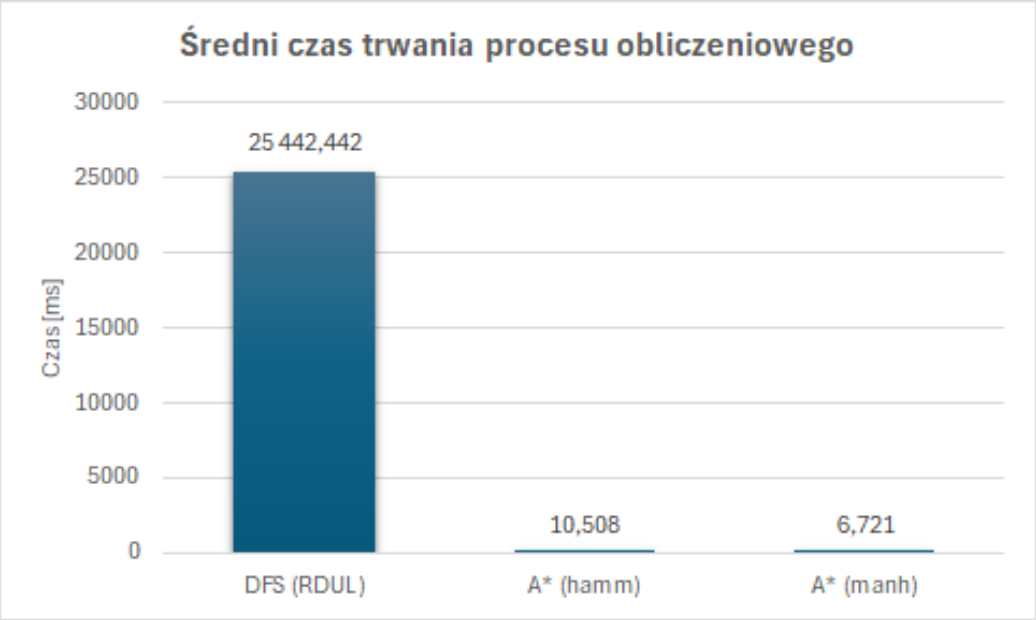
Rysunek 3. Średni czas trwania procesu obliczeniowego dla odległości od rozwiązania równej 1-7.



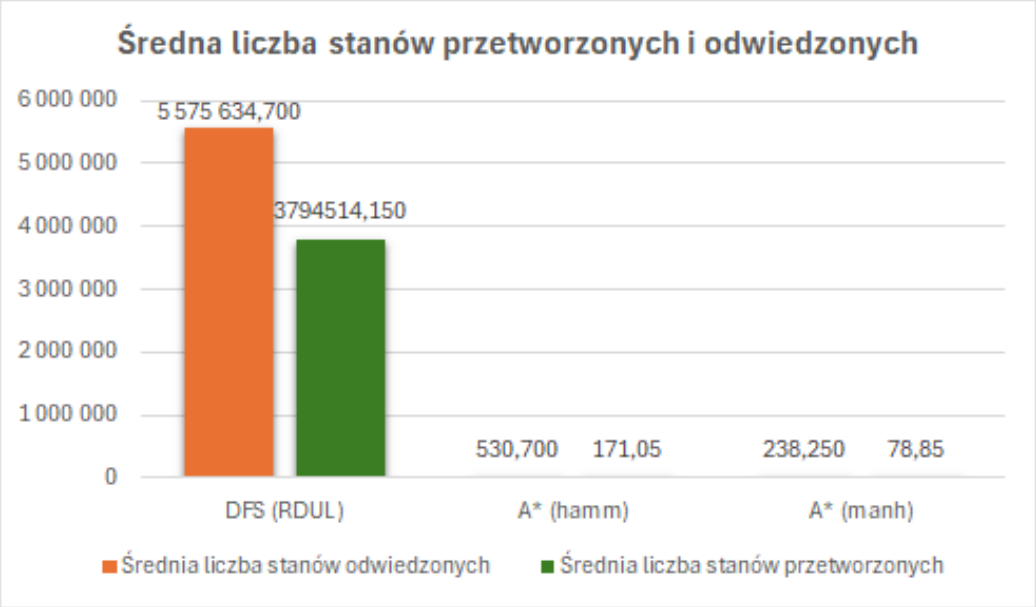
Rysunek 4. Średnia liczba stanów przetworzonych i odwiedzonych dla odległości od rozwiązania równej 1-7.



Rysunek 5. Średnia długość rozwiązań i głębokość rekursji dla odległości od rozwiązania 1-7.



Rysunek 6. Średni czas trwania procesu obliczeniowego dla odległości od rozwiązania równej 15.



Rysunek 7. Średnia liczba stanów przetworzonych i odwiedzonych dla odległości od rozwiązania równej 15.

6. Dyskusja

Na podstawie przeprowadzonych badań można wysunąć następujące wnioski:

— **Dla układów z głębokością rozwiązania 1-7**
  — Najgorsze efekty uzyskano przy pomocy metody DFS. Objawia się to znacznie dłuższym czasem szukania rozwiązania oraz odjandywaniem rozwiązań losowych a niekoniecznie najkrótszych.
  — Metoda BFS zawsze znajduje najkrótsze rozwiazanie, jest to spowodowane sposobem działania metody, która sprawdza wszyskie mozliwe układy na zadanej głębokości przed sprawdzeniem kolejnej.
  — Metoda A* również zawsze odnajduje najkrótsze rozwiązanie, do tego metoda zdecydowanie najszybicej odnajdywała poprawne rozwiązanie. Różnicę pomiędzy wykorzystanymi heurystykami dla układów z głębokością 1-7 jest znikoma.

— **Dla układów z głębokością rozwiązania 15**
  — A* nadal jest najszybszą metodą spośród badanych, odnajdującą najkrótsze rozwiązanie.Dla tej głębokości różnice pomiędzy heurystykami stają się dużo bardziej widoczne, Manhattan jest zdecydowanie bardziej wydajny, osiągając nawet dwukrotnie lepsze wyniki niż Hamming.
  — Metoda DFS również wygenerowała poprawne wyniki.
  — Metoda BFS przez konieczność zapamiętywania wszyskich węzłów na danej głębokości. nie jest w stanie ukończyć rozwiązywania zadania (ograniczony rozmiar pamięci operacyjnej).

7. Wnioski

— A* okazał się najbardziej uniwersalną oraz optmalną metodą przeszukiwania grafów zarówno dla małych jak i duzych odległości układanek od układu wzorcowego. Początkowo znikome różnice pomiędzy heurystykami wzrastają wraz z głębokością rozwiązania układanki. Dla dużych głębokości Manhattan jest znacząco wydajniejszy.
— DFS również był w stanie odnaleźć rozwiązania dla każdego zadanego ułożenia układanki, jednak czas potrzebny na rozwiązanie był znaczie większy, w szczególności, w przypadku, gdy długość najkrótszego rozwiazania była znacznie mniejsza niż zadana głębokość poszukiwań.
— BFS dobrze poradził sobie z przeszukiwaniem układów o małej odległości od układu wzorcowego, jednak dla dużych odległości okazał się zupełnie niezdatny do użytku.

Literatura

[1] *https://personal.math.ubc.ca/ cass/courses/m308-02b/projects/grant/fifteen.html*