Informatyka Stosowana, studia dzienne, inż I st. semestr IV

Sztuczna inteligencja i systemy ekspertowe 2022/2023
Prowadzący: dr inż. Krzysztof Lichy poniedziałek, 15:45

Data oddania: _____ Ocena: _____

Marcin Giska 242390
Przemysław Musiał 242473

Zadanie 1: Piętnastka

1. Cel

Celem części programistycznej jest napisanie programu, który będzie rozwiązywał "Piętnastkę". Cel części badawczej stanowi przebadanie, jak metody przeszukiwania przestrzeni stanów zachowują się w przypadku tego problemu.

2. Wprowadzenie

Łamigłówka „Piętnastka" jest układanką składającą się z ramki i osadzonych w niej 15 elementów, kwadratowej planszy o wymiarach 4x4. Na każdym elemencie znajduje się numer od 1 do 15. Jedno miejsce jest puste, umożliwia ono przesuwanie elementów. Za stan rozwiązany przyjmuje się taki układ elementów, w którym są one ułożone rosnąco od lewej do prawej oraz od góry do dołu. Ułożenie układanki można potraktować jak przeszukiwanie drzewa składającego się z węzłów o wartościach „R", „D", „U" oraz „L" odpowiadających „przesunięciem" pustego pola w prawo, dół, górę oraz lewo.

3. Opis implementacji

Stworzona aplikacja rozwiązująca "Piętnastkę" została napisana w języku Python. Kod składa się z następujących definicji:

— DFS (maksymalna dozwolona głębokość rekursji przyjmuje wartość 20. W sytuacji, gdy program osiągnie taką głębokość i nie znajdzie rozwiązania, powinien wykonać nawrót)

— BFS
— ASTR

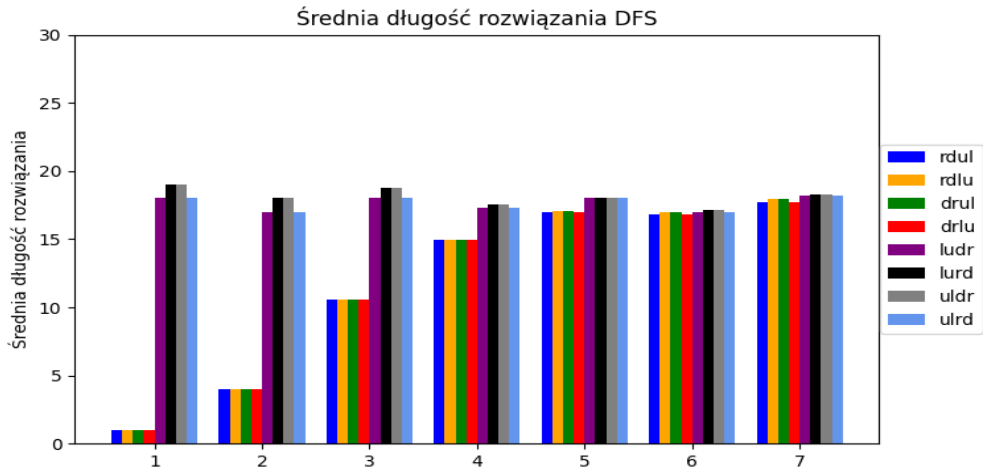
4. Materiały i metody

Do wykonania zadania zostaly wykorzystane skrypty, które są dostępne na stronie przedmiotu. Dzięki tym skryptom zostało storzonych 413 układów. W przypadku strategii ”wszerz” i strategii ”w głąb” użyliśmy 8 następujących porządków przeszukiwania sąsiedztwa:
— prawo-dół-góra-lewo
— prawo-dół-lewo-góra
— dół-prawo-góra-lewo
— dół-prawo-lewo-góra
— lewo-góra-dół-prawo
— lewo-góra-prawo-dół
— góra-lewo-dół-prawo
— góra-lewo-prawo-dół

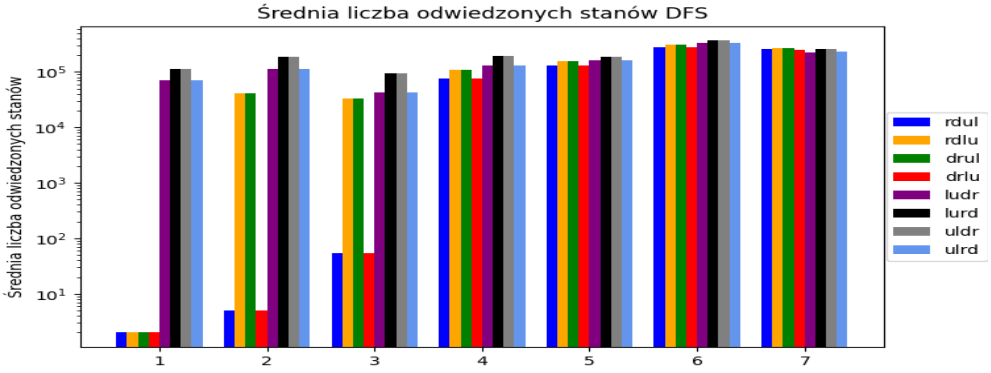
Odpowiednie wykresy zostały wygenerowane za pomocą oddzielnego programu napisanemu w pythonie.

5. Wyniki

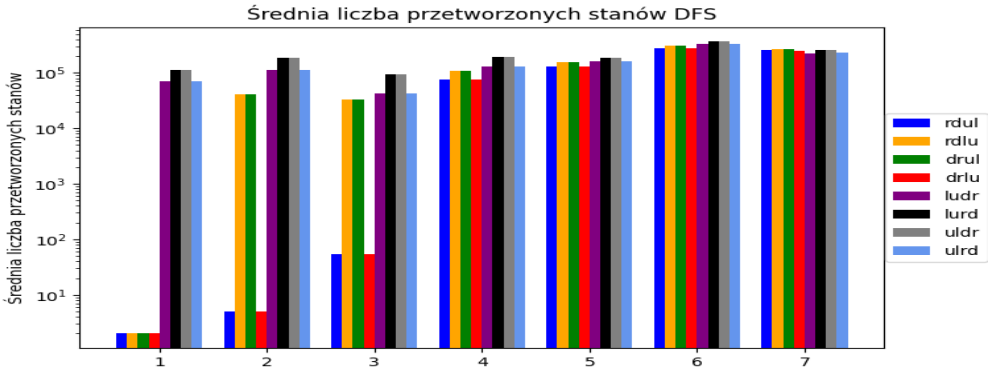
Wykresy zostały podzielone ze względu na strategię przeszukiwania



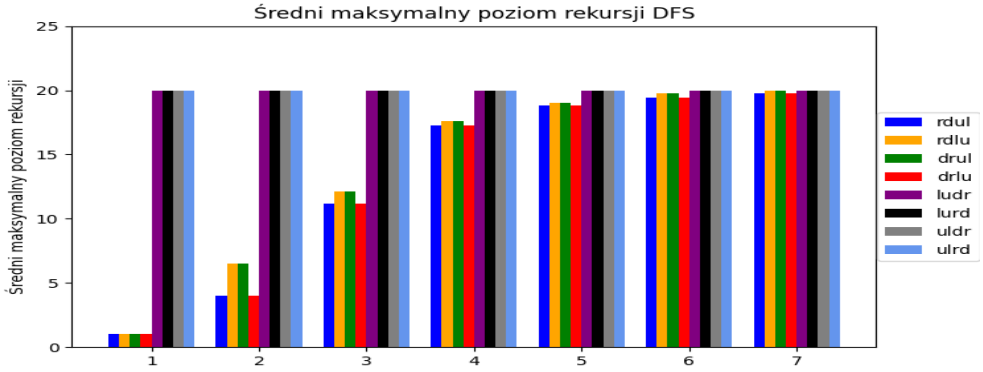
Rysunek 1. Średnia długość rozwiązania



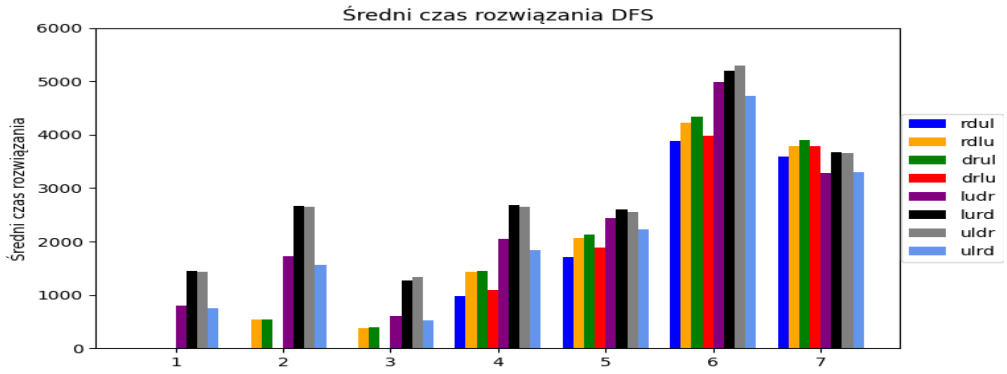
Rysunek 2. Średnia liczba odwiedzonych stanów



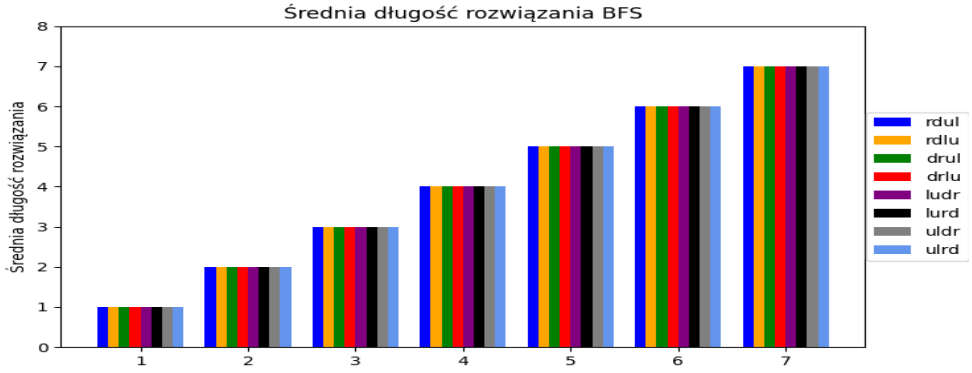
Rysunek 3. Średnia liczba przetworzonych stanów



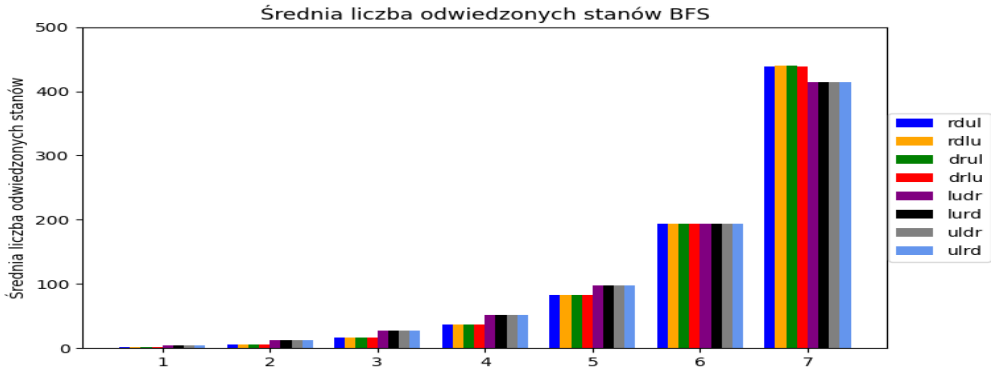
Rysunek 4. Średni maksymalny poziom rekursji



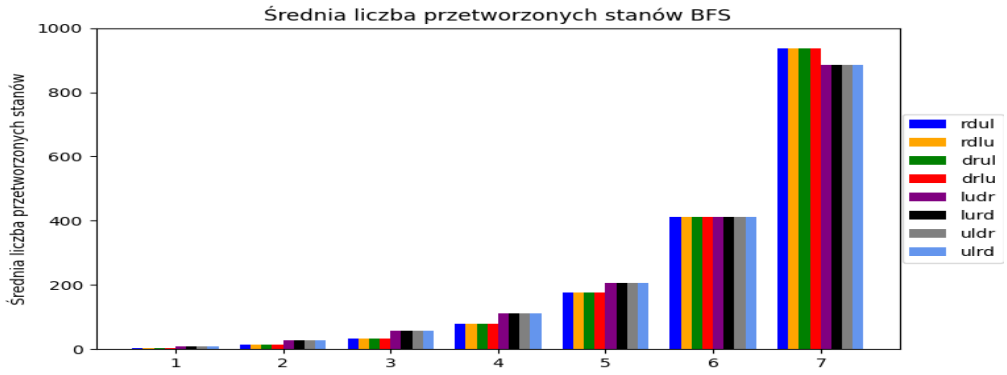
Rysunek 5. Średni czas rozwiązania



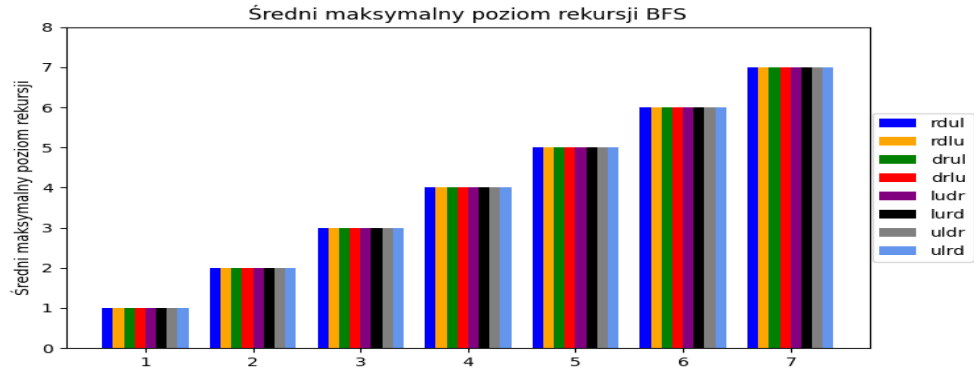
Rysunek 6. Średnia długość rozwiązania



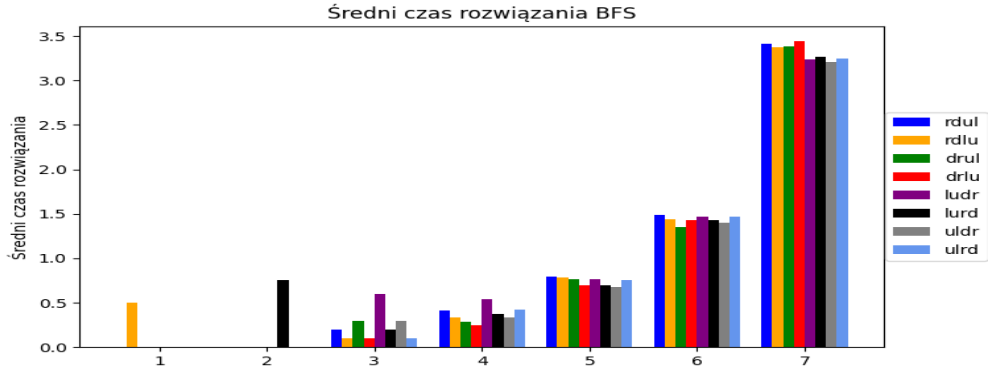
Rysunek 7. Średnia liczba odwiedzonych stanów



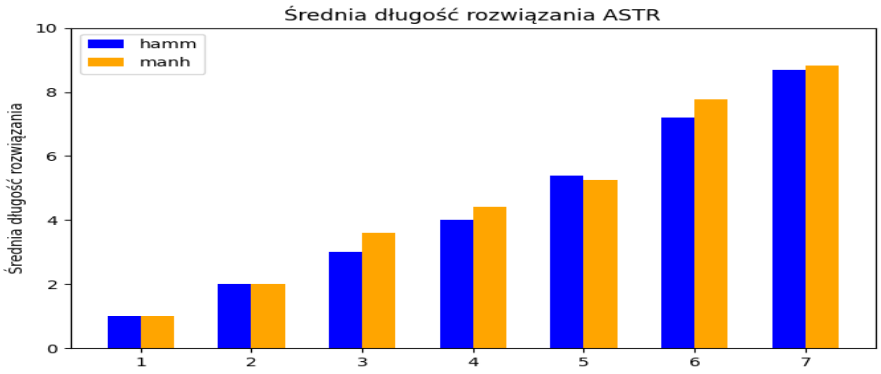
Rysunek 8. Średnia liczba przetworzonych stanów



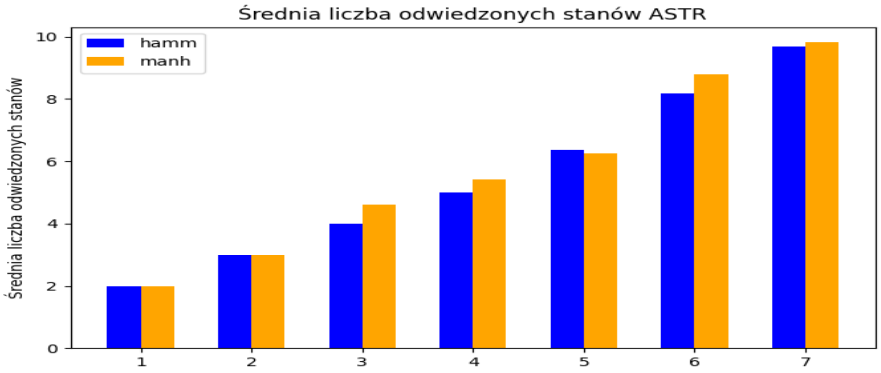
Rysunek 9. Średni maksymalny poziom rekursji



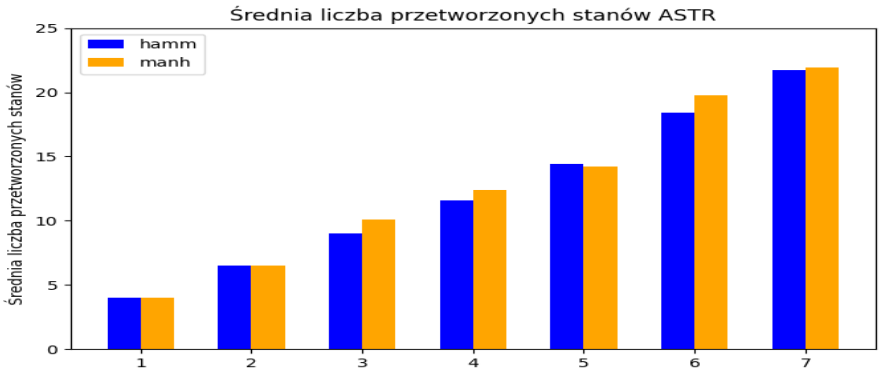
Rysunek 10. Średni czas rozwiązania



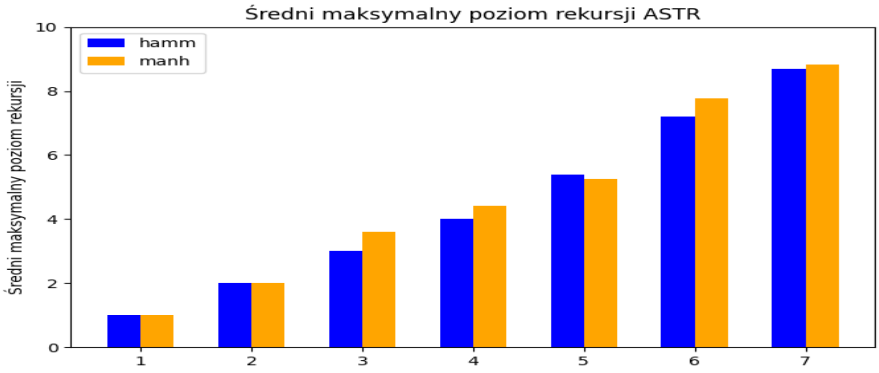
Rysunek 11. Średnia długość rozwiązania



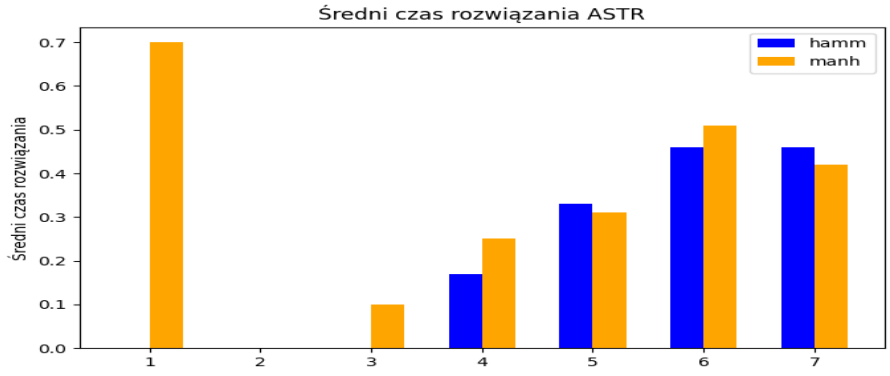
Rysunek 12. Średnia liczba odwiedzonych stanów



Rysunek 13. Średnia liczba przetworzonych stanów



Rysunek 14. Średni maksymalny poziom rekursji



Rysunek 15. Średni czas rozwiązania

6. Dyskusja

6.1. DFS

Pod wszystkimi względami podany algorytm wypada najgorzej. Znalezienie rozwiązania zajmuje mu najdłużej.

6.2. BFS

Algorytm BFS posiada najkrótsze rozwiązania.

6.3. ASTR

W przypadku algorytmu ASTR najlepiej wypada liczba odwiedzonych i przetworzonych sygnałów. Trochę lepiej praktycznie pod każdym względem wypada parametr Hamminga. Nie w każdym przypadku udało się znaleźć rozwiązanie.

7. Wnioski

— Algorytm BFS znajduje najkrótsze rozwiązanie
— Metody heurystyczne nie zawsze potrafią znaleźć rozwiązanie
— Można stwierdzić, że trochę lepsza jest heurystyka Hamminga
— Najgorszym algorytmem jest DFS
— Metody heurystyczne nie zawsze potrafią znaleźć rozwiązanie

Literatura

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