Informatyka, studia dzienne, inż I st. semestr V

Sztuczna inteligencja i systemy ekspertowe **2018/2019**

Prowadzący: Przemysław Nowak Czwartek, 12:00

Data oddania: _____ Ocena: _____

Patryk Lisik 210254
Adam Sadowski 210310

Zadanie 1: Piętnastka

1. Cel

Implementacja kilku algorytmów przeszukiwania przestrzeni stanów oraz rozwiązanie i analiza wyników zadanych stanów układanki "piętnastka".

2. Wprowadzenie

2.1. Piętnastka

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

Tablica 1: Rozwiązana piętnastka

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

Tablica 2: Układ wymagający jednego ruchu

Piętnastka jest grą logiczną wymyśloną pod koniec XIX wieku. Jej nazwa pochodzi od piętnastu ponumerowanych 1–15, które trzeba ułożyć tak jak w tablicy 1.

Stan końcowego musi musi zostać osiągnięty jedynie poprzez ruchy pustego pola w obrębie macierzy 4×4. Jeśli puste pole znajduje sie przy krawędzi niektóre ruchy mogą nie być dozwolone. W dalszej części ruchy będą oznaczane.

— L – (ang. left) w lewo
— R – (ang. right) w prawo
— D – (ang. down) w dół
— U – (ang. up) w górę

2.1.1. Złożonosć i rozwiązywalnosć

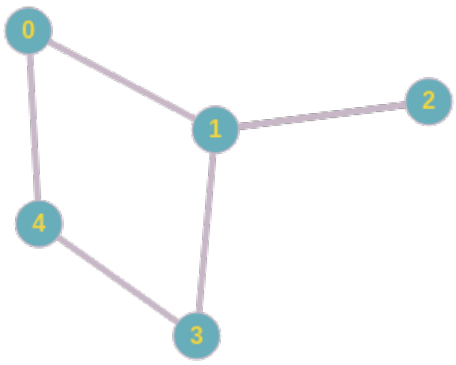
Ilość wszystkich stanów układanki można łatwo obliczyć znakując permutację bez powtórzeń zbioru elementów układanki($\{$"*puste_pole*", $1, 2 \ldots, 15\}$). Stany układanki można podzielić ze względu na ich parzystość. Nie można przejść z układu do nieparzystego, czego oczywistą implikacją jest nierozwiązywalność połowy układów[4]. Podsumowując ilość rozwiązywalnych stanów to $\frac{16!}{2} \approx 1.04 \cdot 10^{13}$

2.2. Grafy

Grafem nazywamy strukturę danych modelującą zbiór wierzchołków i połączeń między nimi.

2.2.1. Reprezenatacja

Najbardziej oczywistą formą reprezentacji grafów sa rysunki podobne do ilustracji 1. Nie jest to jednak optymalny sposób przedstawiania grafu w pamięci komputera.



Rysunek 1: Przykładowy graf

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	0
2	0	1	0	0	0
3	0	1	0	0	1
4	1	0	0	1	0

Tablica 3: Przykładowa macierz sąsiedztwa

Macierz sąsiedztwa to sposób reprezentacji grafu w którym krawędzie reprezentowane są przez wartość 1 w komórce której pozycja odpowiada numerom wierzchołków. Na rysunku 1 widzimy połączenie 1–2. Jest ono reprezentowane w tablicy 3 poprzez 1 na pozycji 12 i 21. Redundancja informacji spowodowana jest tym, że graf na rysunku 1 jest nieskierowany(nie można wyróżnić kierunku połączenia). Wady tej metody reprezentacji ujawniają się w przypadku grafów o małej liczbie połączeń. Ilość elementów w macierzy jest proporcjonalna do kwadratu ilości wierzchołków, gdy ilość połączeń jest znikoma duża część przechowywanych informacji to 0.

0	1	4	
1	2	0	3
2	1		
3	1	4	
4	3	0	

Tablica 4: Przykładowa lista sąsiedztwa

Lista sąsiedztwa to sposób reprezentacji grafu przez wektor wektorów, gdzie indeksami pierwszej z nich(pogrubione cyfry w tablicy 4) oznaczany numer wierzchołka. Pod każdym z indeksów znajduje się lista połączeń każdego wierzchołka. Zaletą tego rozwiązania jest stały czas dostępu do listy połączeń danego wierzchołka.

2.2.2. Metody przeszukiwania

DFS Depth first search – przeszukiwanie w głąb
Opiera się na przeszukiwaniu od korzenia wzdłuż każdej gałęzi tak głęboko, jak to możliwe zanim zacznie się cofać

BFS Breath first search – przeszukiwanie w szerz
Opiera się na przeciwnej strategii do DFS i najpierw przeszukuje najpłycej położone węzły, zanim zejdzie głębiej. Z pośród rozważanych algorytmów jest to jedyny gwarantujący znalezienie najbliższego rozwiązania.

A* A star/gwiazdka – heurystyczne przeszukiwanie
Opiera się na zupełnie innym podejściu. W każdej iteracji wyznaczany jest koszt stanów, zgodnie z używaną metryką, i algorytm przeszukuje drzewo w pierwszej kolejności po węzłach o najniższym koszcie.
Wykorzystane metryki to:
Manhattan Dla każdego punktu liczona jest odległość od punktu docelowego

$$d = \sum_{i=1}^{n} |x_i - x'_i| + |y_i - y'_i|$$

Hamminga Koszt zwiększa się wraz z każdym elementem będącym na złej pozycji

2.2.3. Pozostałe pojęcia

Cykl
Droga

3. Opis implementacji

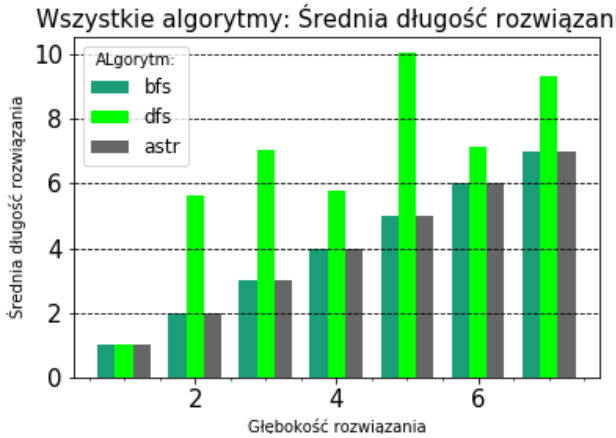
Implementację wykonano w języku Python3. Całość implementacji poza parserem parametrów i funkcją main() zamyka się w jednej klasie o długości ok. 200 linii, dlatego zrezygnowaliśmy z zamieszczania diagramu UML.

4. Materiały i metody

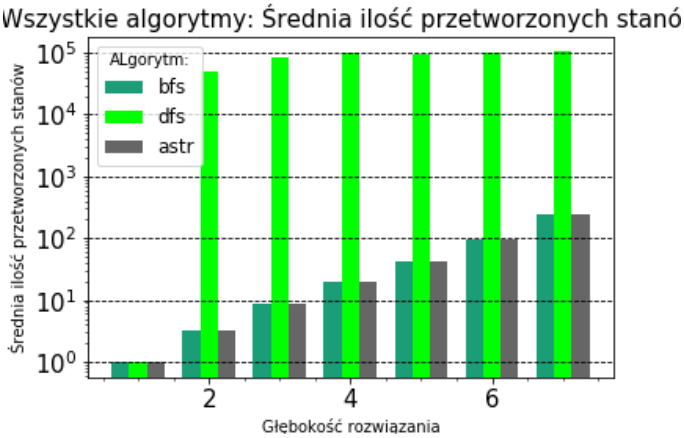
1. Przygotowanie danych
   — Pobranie z platformy WIKAMP generatora układanek 4×4
   — Wygenerowanie przypadków testowych
2. Uruchomienie przygotowanego programu
   — Pobranie z platformy WIKAMP skryptu uruchamiającego.
   — Modyfikacja polecenia uruchamiającego w skrypcie.
   — Uruchomienie skryptu
3. Analiza wyników i generowanie wykresów przy pomocy Jupyter Notobook'a
4. Profit

Celem zachowania wiarygodności pomiarów czasów program uruchomiono w środowisku tekstowym, jako process o największym priorytecie. Podczas przetwarzania nie doszło do użycia pamięci swap ani throttlingu procesora.

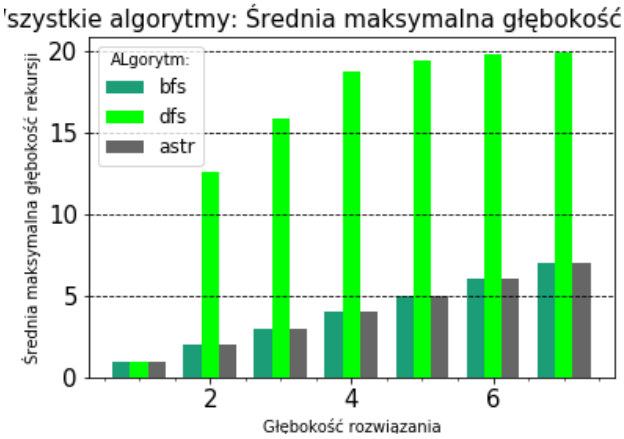
5. Wyniki



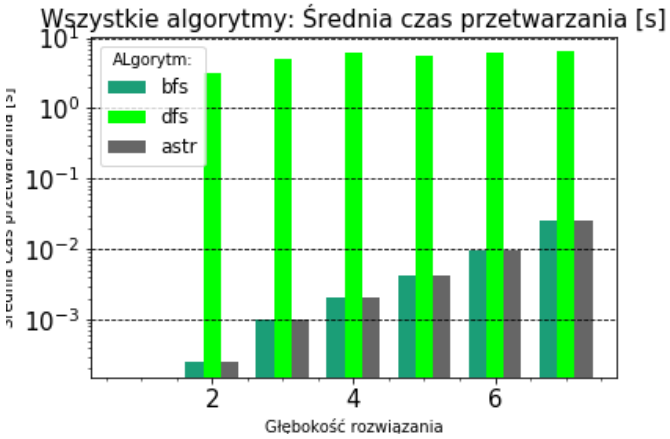
(a) Średnia długość rozwiązania



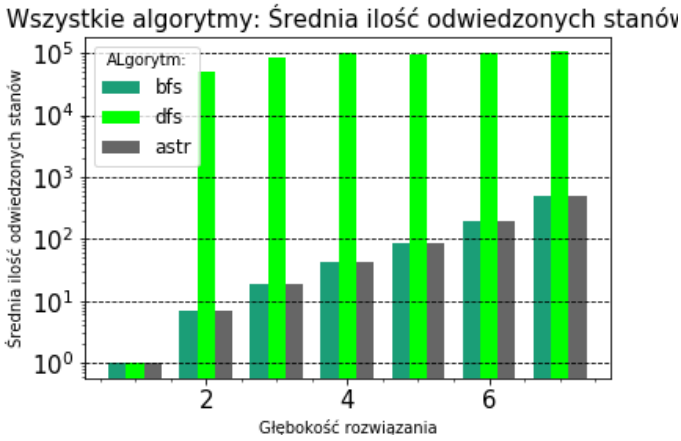
(b) Średnia ilość przetworzonych stanów



(c) Średnia maksymalna głębokość rekursji

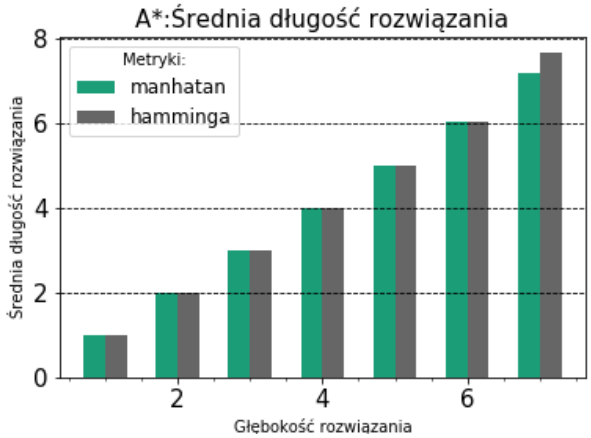


(d) Średni czas przetwarzania

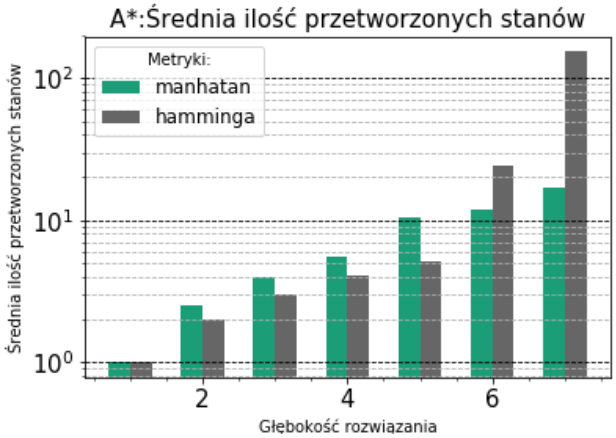


(e) Średnia ilość odwiedzonych stanów

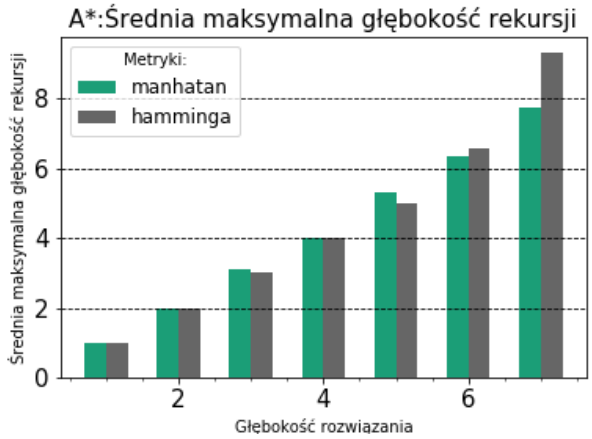
Rysunek 2: Porównanie wszystkich metod przeszukiwania



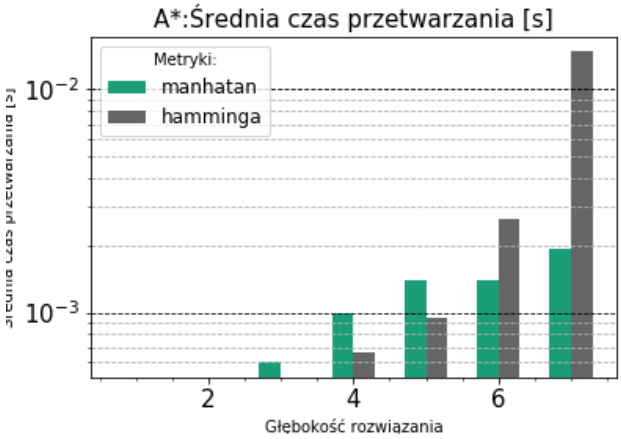
(a) Średnia długość rozwiązania



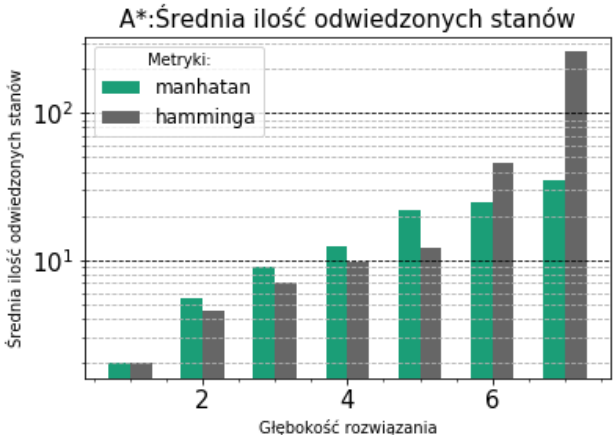
(b) Średnia ilość przetworzonych stanów



(c) Średnia maksymalna głębokość rekursji

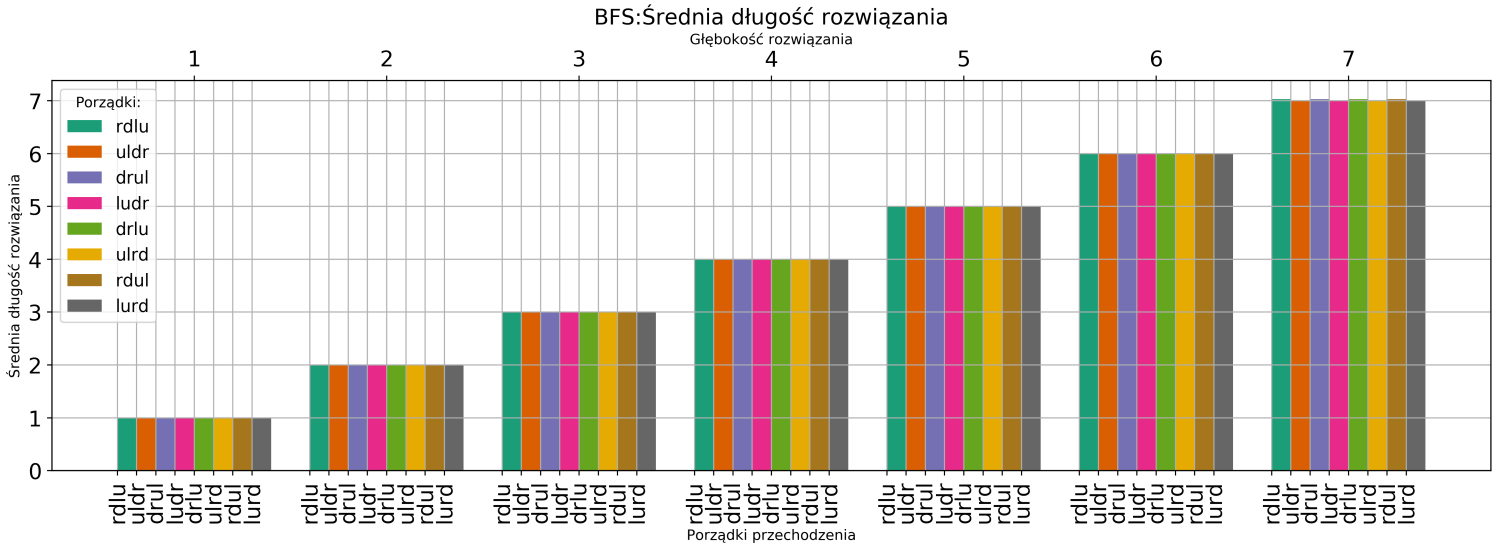


(d) Średni czas przetwarzania

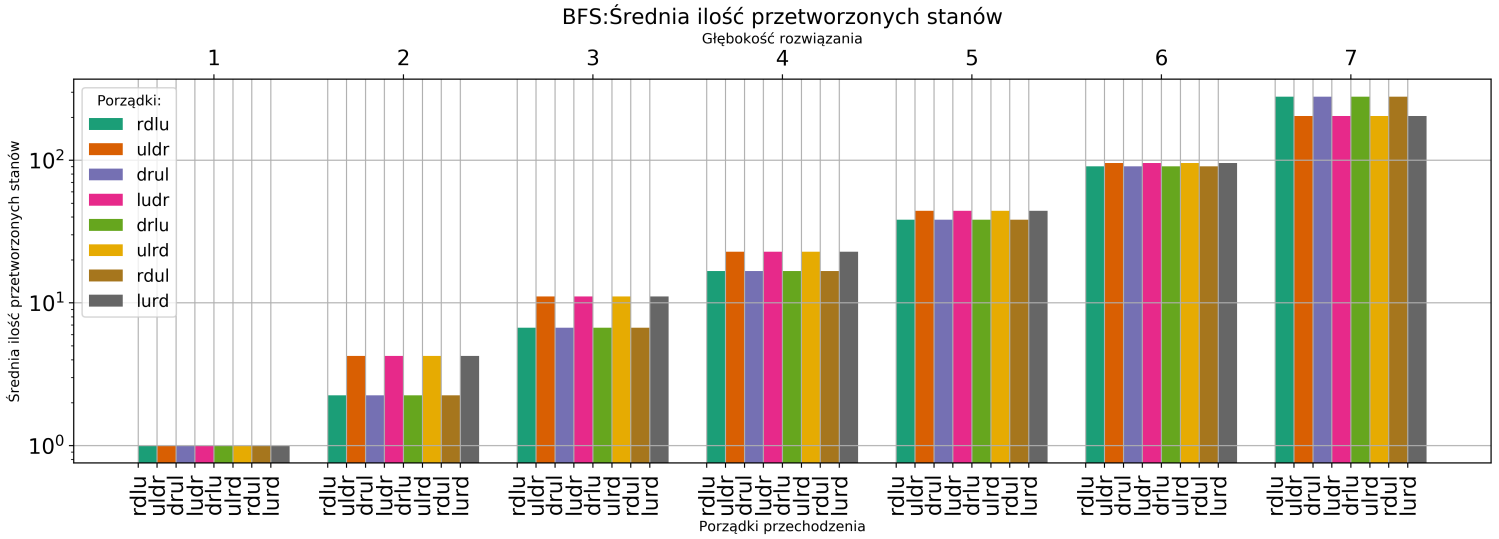


(e) Średnia ilość odwiedzonych stanów

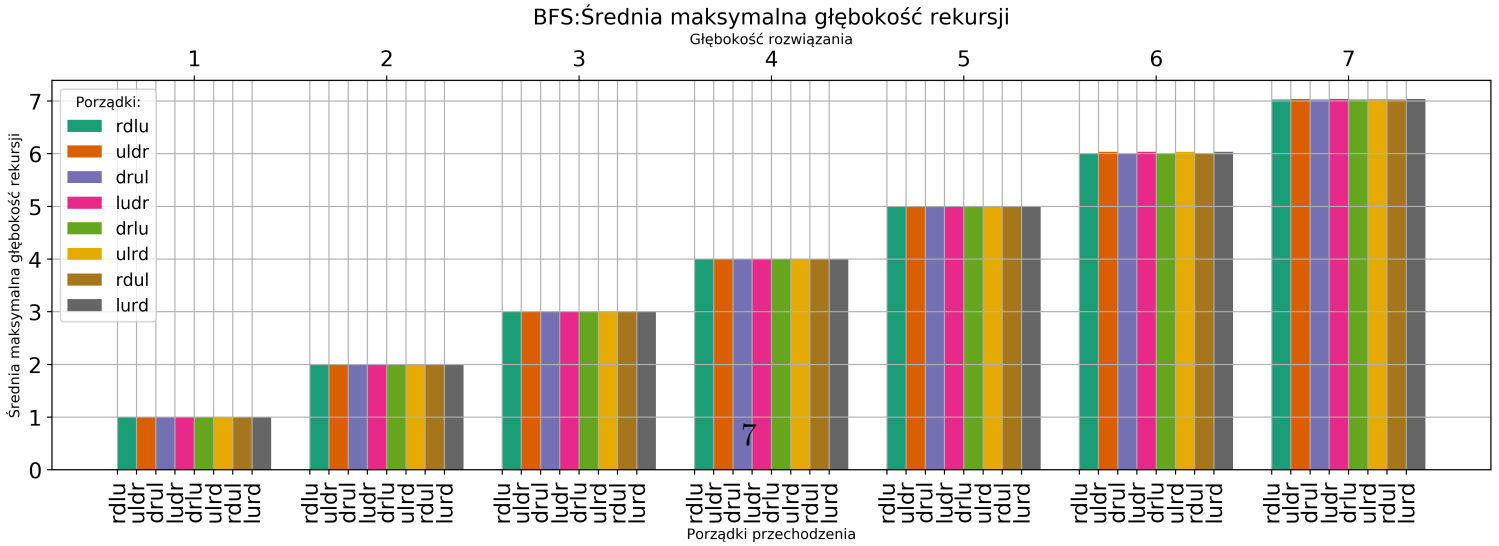
Rysunek 3: Porównanie wszystkich metod przeszukiwania



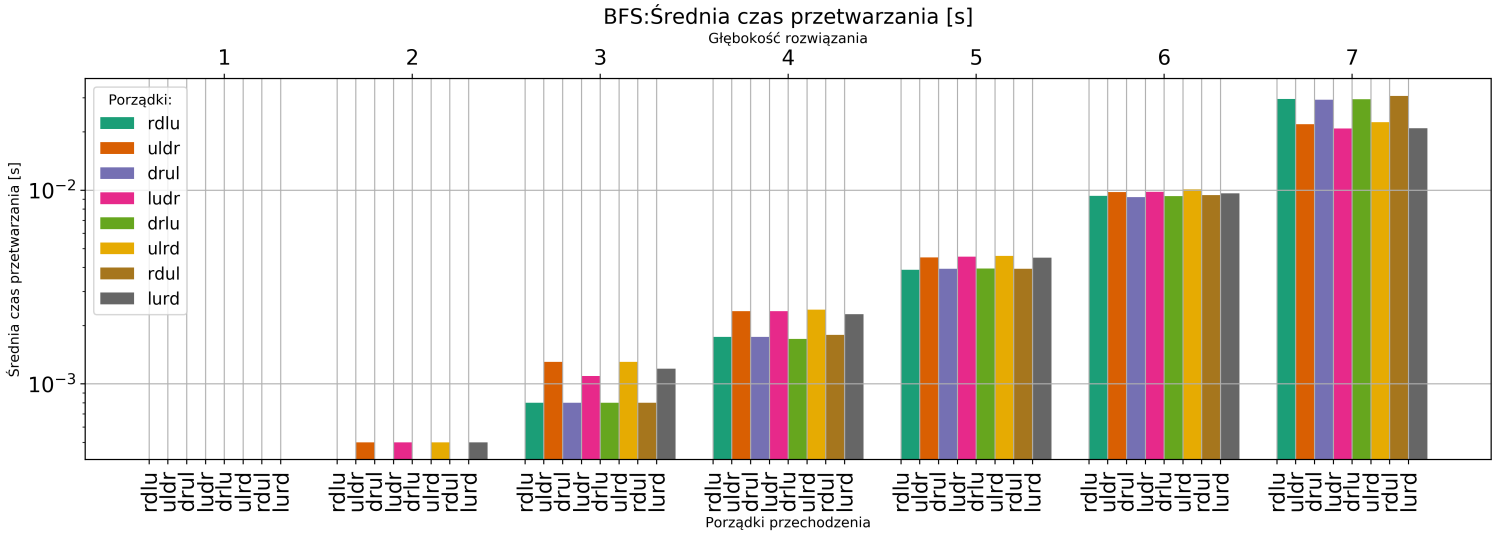
Rysunek 4: BFS – Średnia długość rozwiązania



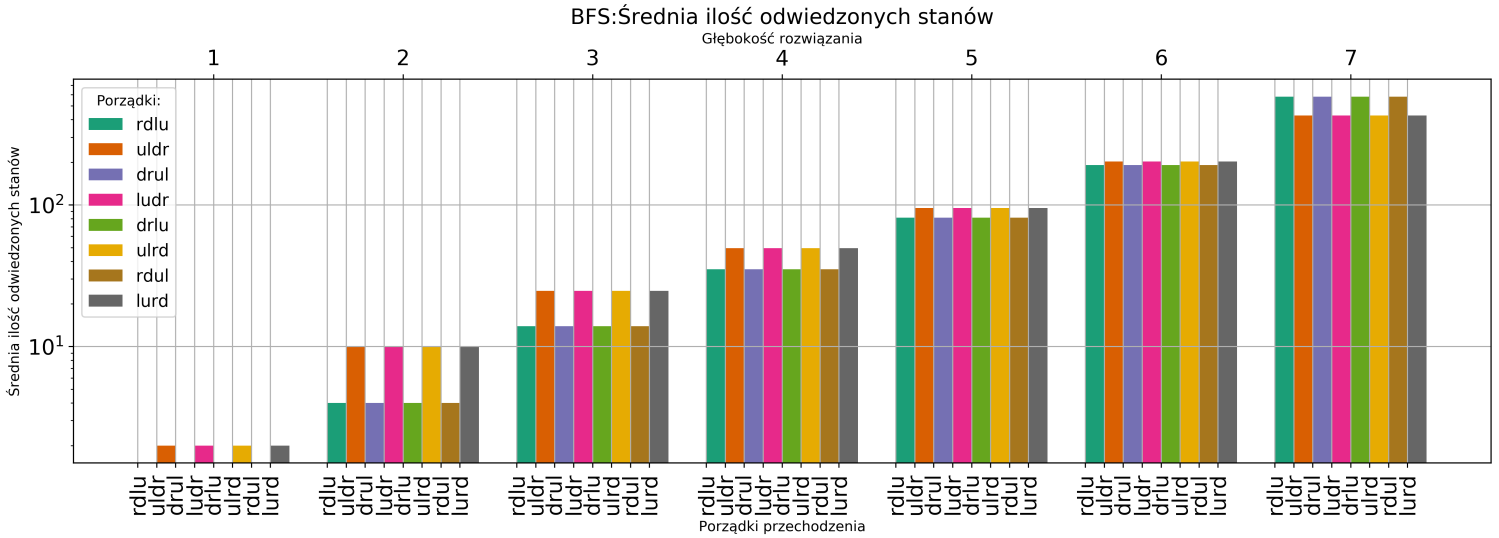
Rysunek 5: BFS – Średnia ilość przetworzonych stanów



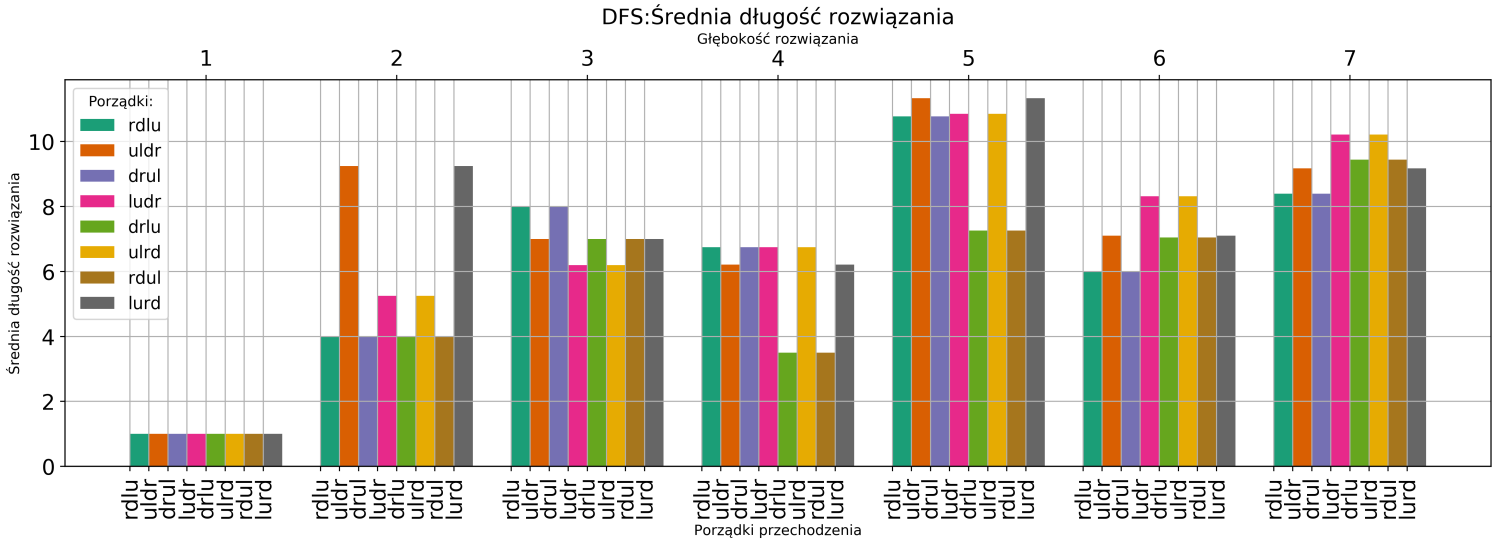
Rysunek 6: BFS – Średnia maksymalna głębokość rekursji



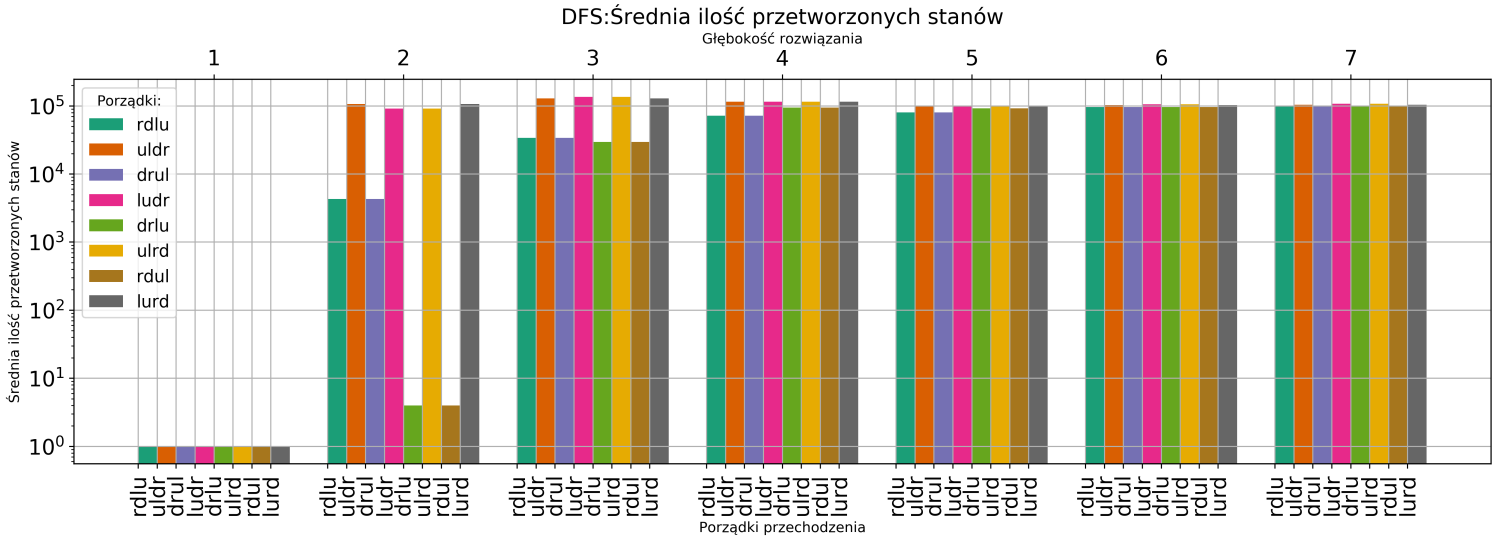
Rysunek 7: BFS – Średni czas przetwarzania



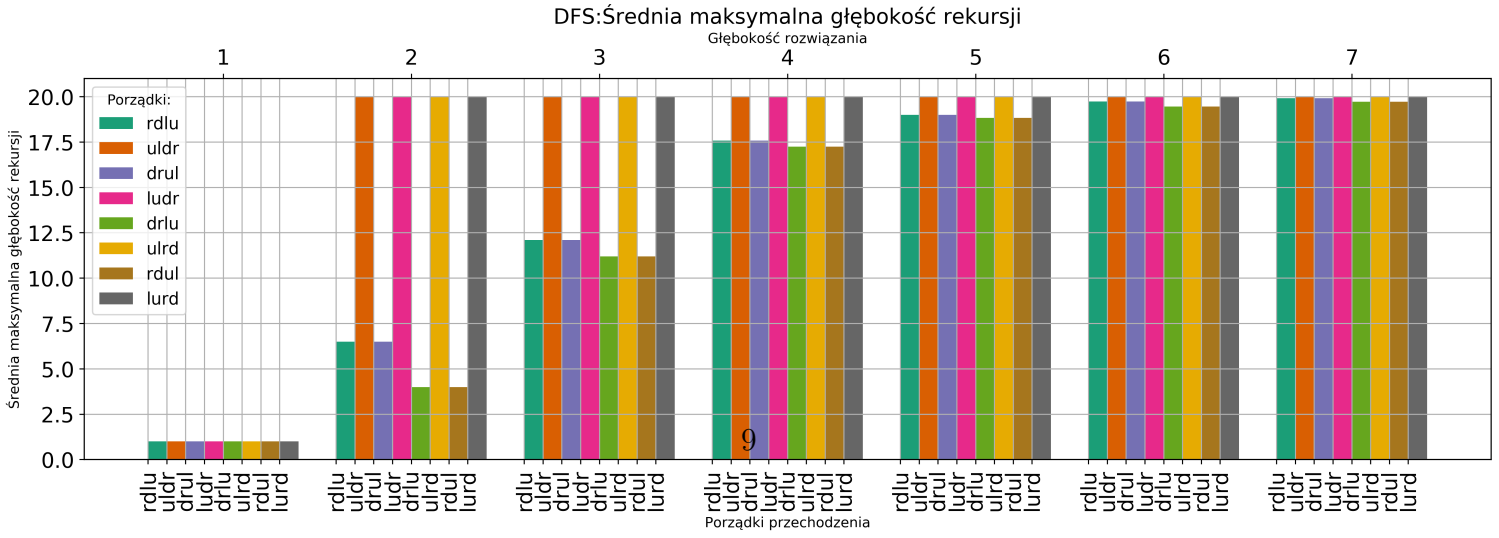
Rysunek 8: BFS – Średnia ilość odwiedzonych stanów



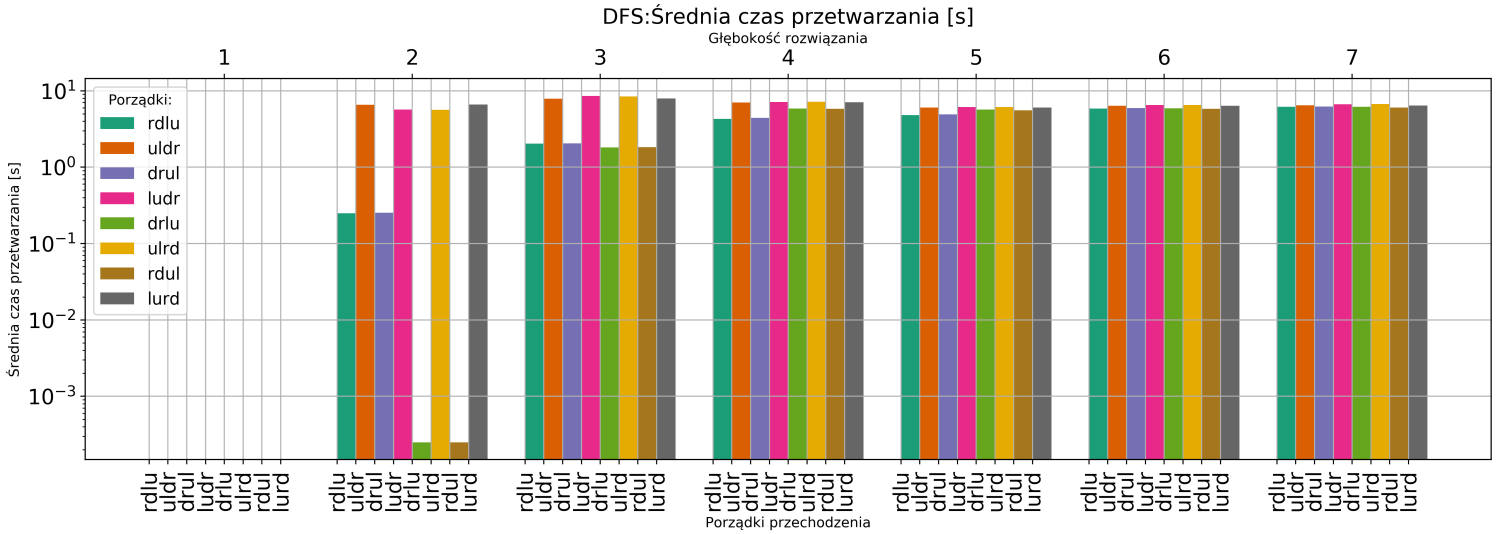
Rysunek 9: DFS – Średnia długość rozwiązania



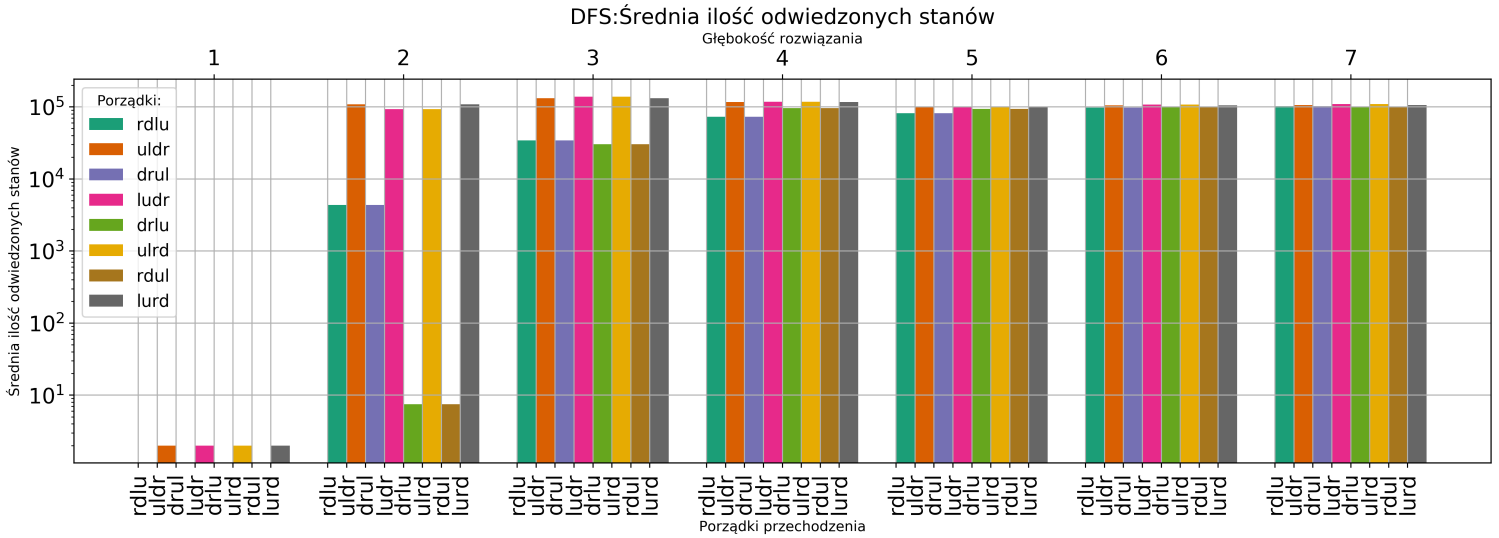
Rysunek 10: DFS – Średnia ilość przetworzonych stanów



Rysunek 11: DFS – Średnia maksymalna głębokość rekursji



Rysunek 12: DFS – Średni czas przetwarzania



Rysunek 13: DFS – ŚŚrednia ilość odwiedzonych stanów

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

7. Wnioski

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.

Literatura

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

[2] *15 Puzzle* `http://mathworld.wolfram.com/15Puzzle.html`

[3] *The Fifteen Puzzle can be solved in 43 "moves"* `http://cubezzz.duckdns.org/drupal/?q=node/view/223`

[4] *The Fifteen Puzzle* `http://www.math.ubc.ca/~cass/courses/m308-02b/projects/grant/fifteen.html`

Na końcu należy obowiązkowo podać cytowaną w sprawozdaniu literaturę, z której grupa korzystała w trakcie prac nad zadaniem.