Data oddania: _____ Ocena: _____

Wiktor Bechciński 229840
Kamil Budzyn 229850

Zadanie 1: Piętnastka

1. Cel

Celem zadania była implementacja oraz przebadanie jak zachowują się strategie przeszukiwania przestrzeni stanów przy rozwiązywaniu łamigłówki znanej jako "Piętnastka".

2. Wprowadzenie

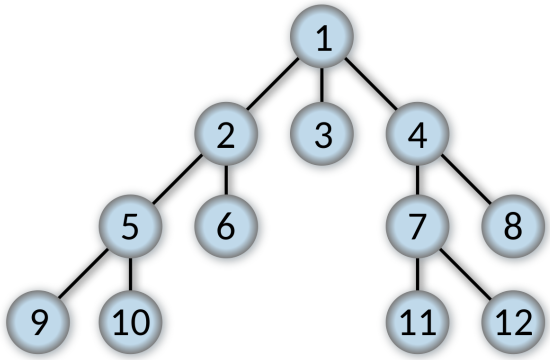
Piętnastka to klasyczna łamigłówka, w której należy ułożyć liczby w kolejności od 1 do 15 na planszy o rozmiarach 4x4. Jedno miejsce na planszy jest puste i umożliwia zamienianie go z sąsiadującymi polami tak aby uzyskać poprawny układ:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 0 \end{bmatrix}$$

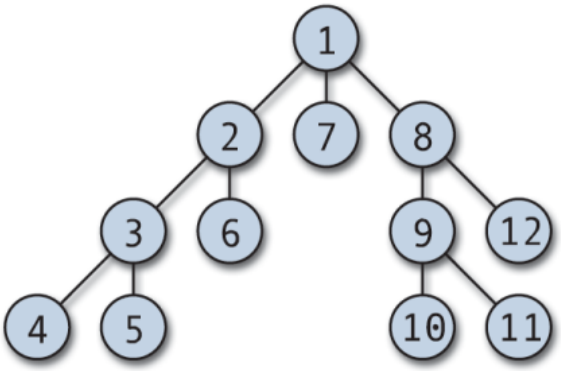
Rozwiązywanie układanki odbywa się przy użyciu trzech strategii przeszukiwania stanów:

1. Strategia BFS (Breadth-First-Search):

Jest to algorytm przeszukiwania grafu 'wszerz', polega na przeszukiwaniu kolejnych poziomów grafu aż do znalezienia węzła docelowego. Metoda najpierw pobiera stan początkowy, sprawdza czy spełnia on warunek stopu następnie przechodzi do wyznaczenia wszystkich możliwych stanów potomnych zgodnie z podaną kolejnością. Każdy kolejny stan zostaje sprawdzony czy został już rozpatrzony, jeśli nie to dodawany jest do kolejki z której pobierane są kolejne stany do przetworzenia. Algorytm wykonuje się do momemntu gdy nie będzie już nowych stanów.



2. Strategia DFS (Depth-First-Search):
Jest to algorytm przeszukiwania grafu 'wgłąb', polega na przeszukiwaniu grafu wzdłuż jednej z gałęzi w zadanym kierunku. Gdy metoda dojdzie do końca danej gałęzi to wraca do rodzica i przechodzi do kolejnego dziecka. W przypadku strategi DFS wprowadziliśmy również maksymalną głębokość rekursji. Jeśli do danego stanu dotrzemy z kosztem równym przyjętej wartości maksymalnej to nie jest on poddawany dalszemu przetwarzaniu i wykonywany jest nawrót.



3. Strategia A* (A-Star):
Jest to algorytm przeszukiwania grafu w kolejności od najlepszej możliwości. Wybierana jest ścieżka pomiędzy wierzchołkami początkowym i końcowym. Wierzchołki przetworzone trafiają na listę zamkniętą, a te do przetworzenia znajdują się na liście otwartej. Stany są posortowane według sumy " odległości od rozwiązania "(w oparciu o jedną z Heurystyk) i kosztu dotarcia do nich - te o najniższej sumie pobierane są do przetwarzania w pierwszej kolejności. Wykorzystano dwie Heurystyki obliczania w/w odległości:

1. Heurystyka Hamminga - obliczona odległość jest sumą ilości pól nie znajdujących się na właściwym miejscu względem stanu docelowego.

2. Heurystyka Manhattan - obliczona odległość jest sumą odległości wszystkich pól od ich właściwych miejsc względem stanu docelowego, z zastosowaniem metryki Manhattan. Odległość dla pojedynczego pola obliczana jest według wzoru:

$$d(a,b) = |x_a - x_b| + |y_a - y_b| \tag{1}$$

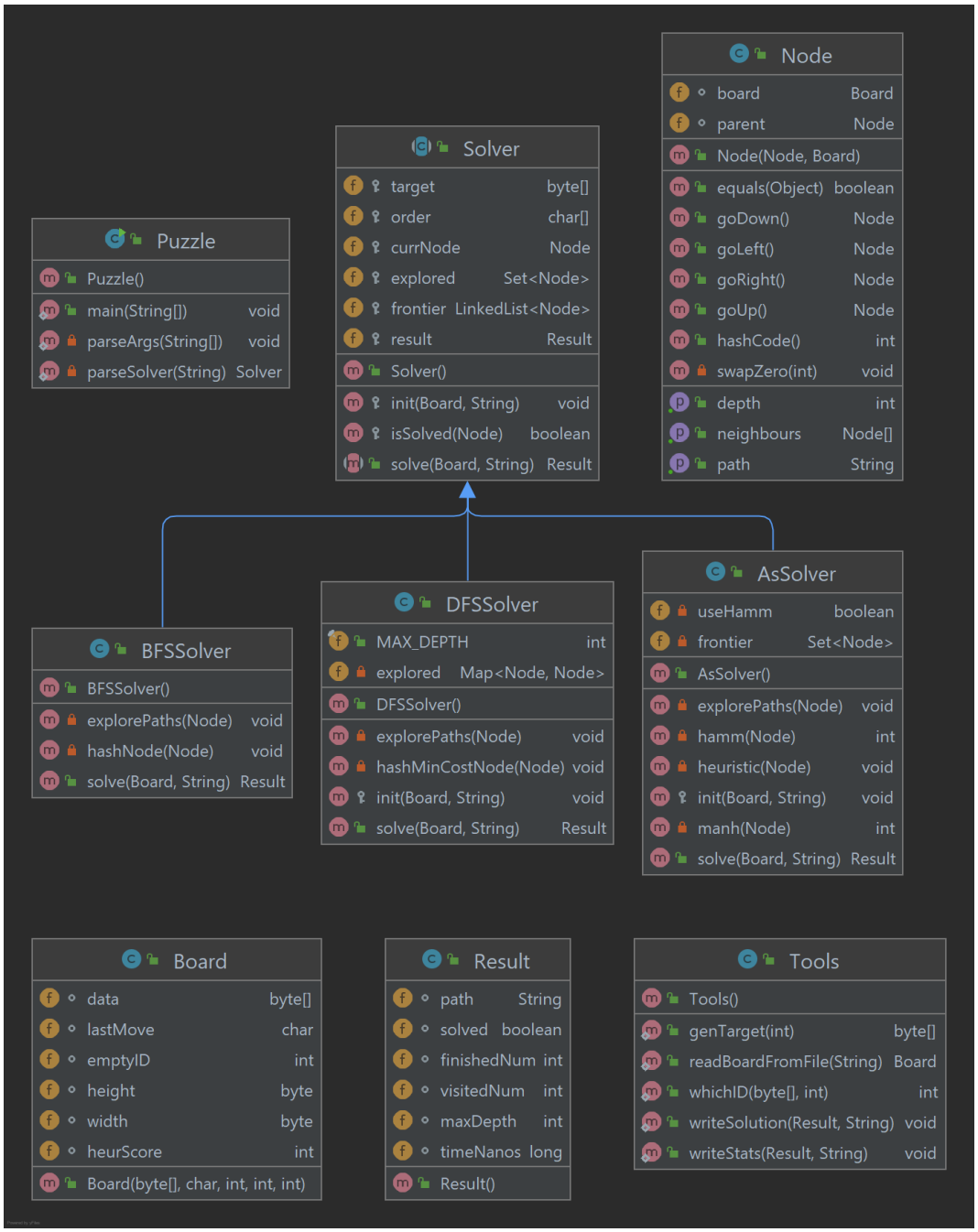
d - obliczana odległość
x_a, y_a - współrzędne pola przetwarzanego stanu
x_b, y_b - współrzędne pola stanu docelowego

Jest to zatem suma wartości bezwzględnych różnic współrzędnych pola stanu przetwarzanego i jego odpowiednika w stanie docelowym.

3. Opis implementacji

Program został napisany w języku Java zgodnie z wymaganiami funkcyjnymi. Jego konfiguracja odbywa się poprzez parametry wywołania gdzie podawane są kolejno: strategia przeszukiwania, parametr strategii przeszukiwania, nazwę pliku wejściowego, nazwę pliku wyjściowego z rozwiązaniem i nazwę pliku ze statystykami.

— Puzzle - klasa główna programu, odbiera parametry wywołania programu oraz uruchamia odpowiednią strategię wraz z wszystkimi parametrami podanymi przez użytkownika
— Solver - abstrakcyjna klasa wspólna dla klas z implementacją strategii
— AsSolver - klasa rozszerzająca abstrakcyjną klasę, zawiera implementację strategii A*
— BFSSolver - klasa rozszerzająca abstrakcyjną klasę, zawiera implementację strategii przeszukiwania wszerz (BFS)
— DFSSolver - klasa rozszerzająca abstrakcyjną klasę, zawiera implementację strategii przeszukiwania wgłąb (DFS)
— Tools - klasa zawierająca funkcje do obsługi wejścia - wyjścia oraz funkcje pomocnicze
— Node - klasa odpowiadająca za ”poruszanie” się po planszy



4. Materiały i metody

W części badawczej przeprowadziliśmy eksperymenty polegające na próbie znalezienia rozwiązania wszystkich układów początkowych w odległościach 1-7 od układu docelowego. Przeprowadziliśmy 18 eksperymentów i w ramach każdego z nich zbadaliśmy 413 układów:

— Strategia BFS oraz DFS, obie dla 8 przyjętych porządków operatorów (RDUL, RDLU, DRUL, DRLU, LUDR, LURD, ULDR, ULRD)

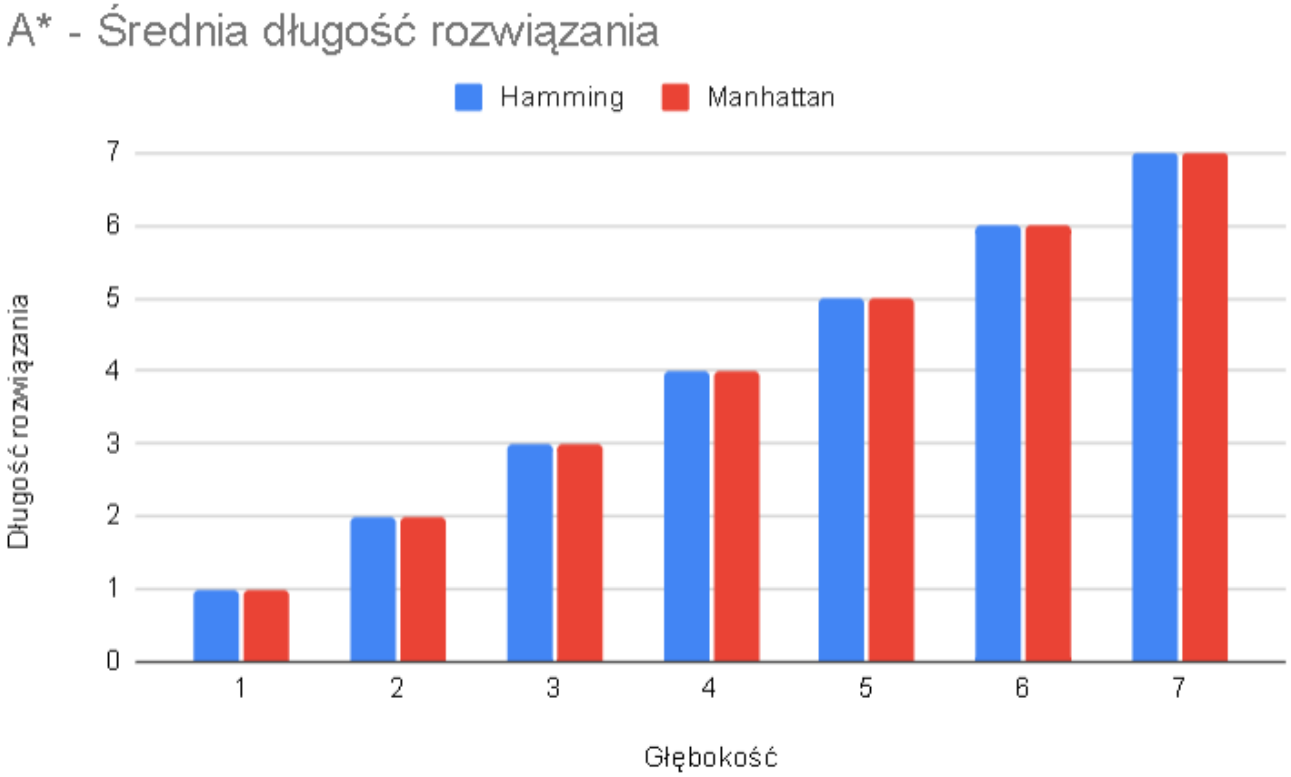
— Strategia A* dla 2 przyjętych Heurystyk (Hamminga oraz Manhattan) - 2 eksperymenty

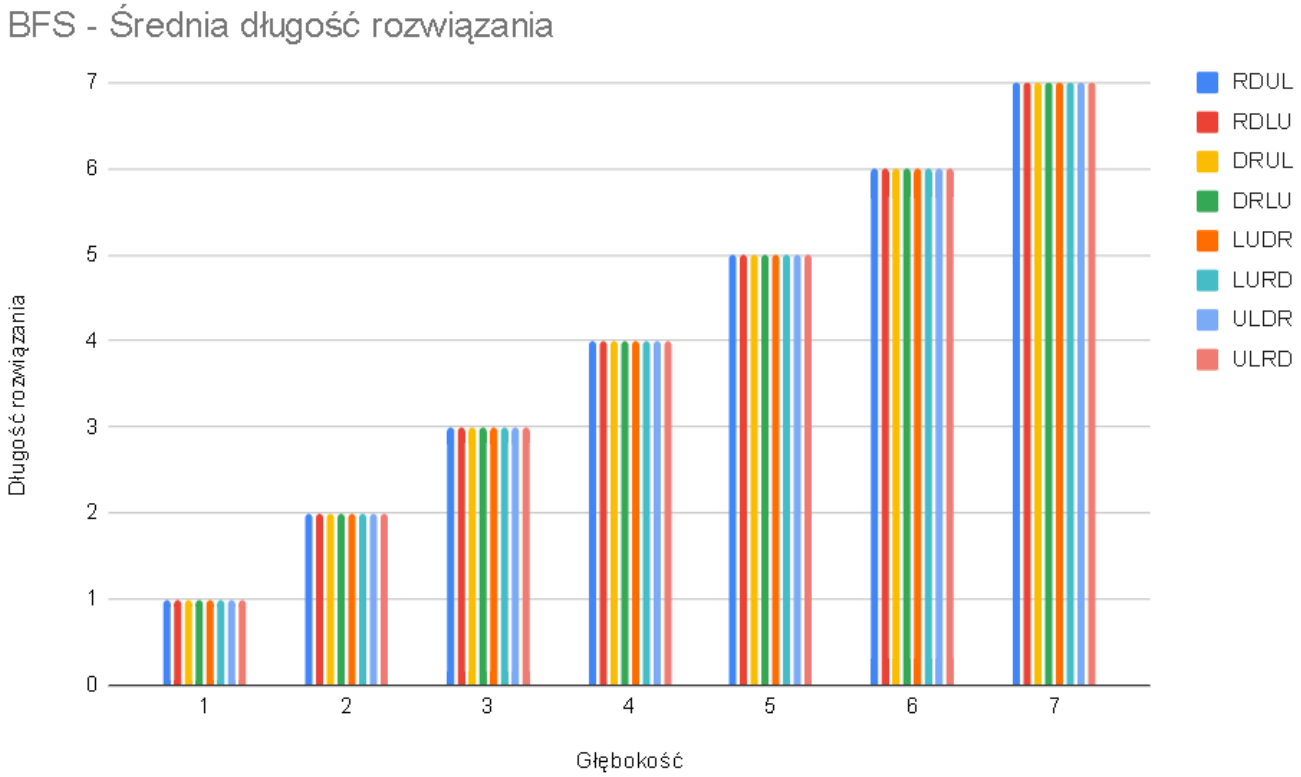
W strategii DFS przyjęta maksymalna głębokość rekursji to 20. Otrzymaliśmy łącznie 7434 przypadki testowe. Do uzyskania wyników dla wszystkich przypadków zostały wykorzystane skrypty, służące do uruchomienia programu w trybie wsadowym, udostępnione na platformie WIKAMP. Dla każdego przypadku testowego został wygenerowany plik ze statystykami zawierający:

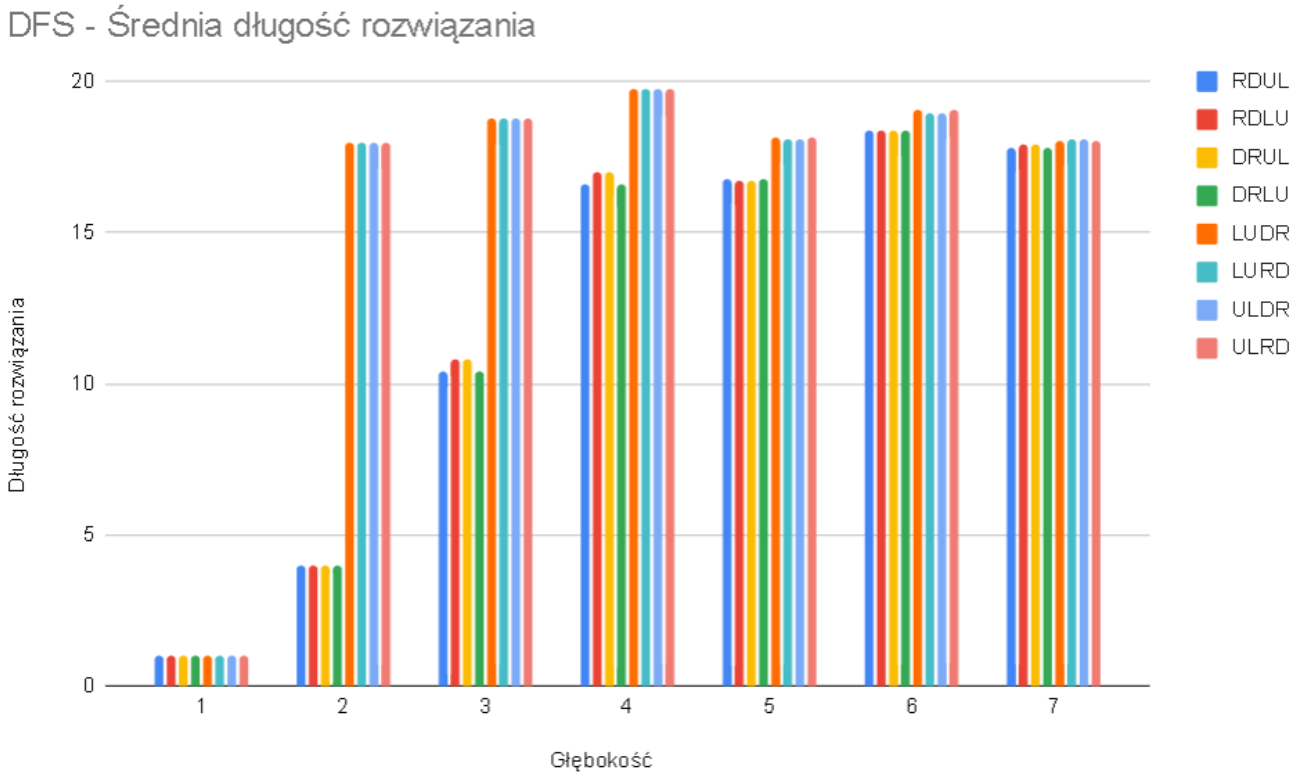
1. długość otrzymanego rozwiązania
2. liczbę stanów odwiedzonych
3. liczbę stanów przetworzonych
4. maksymalną osiągniętą głębokość rekursji
5. czas trwania procesu obliczeniowego (ms)

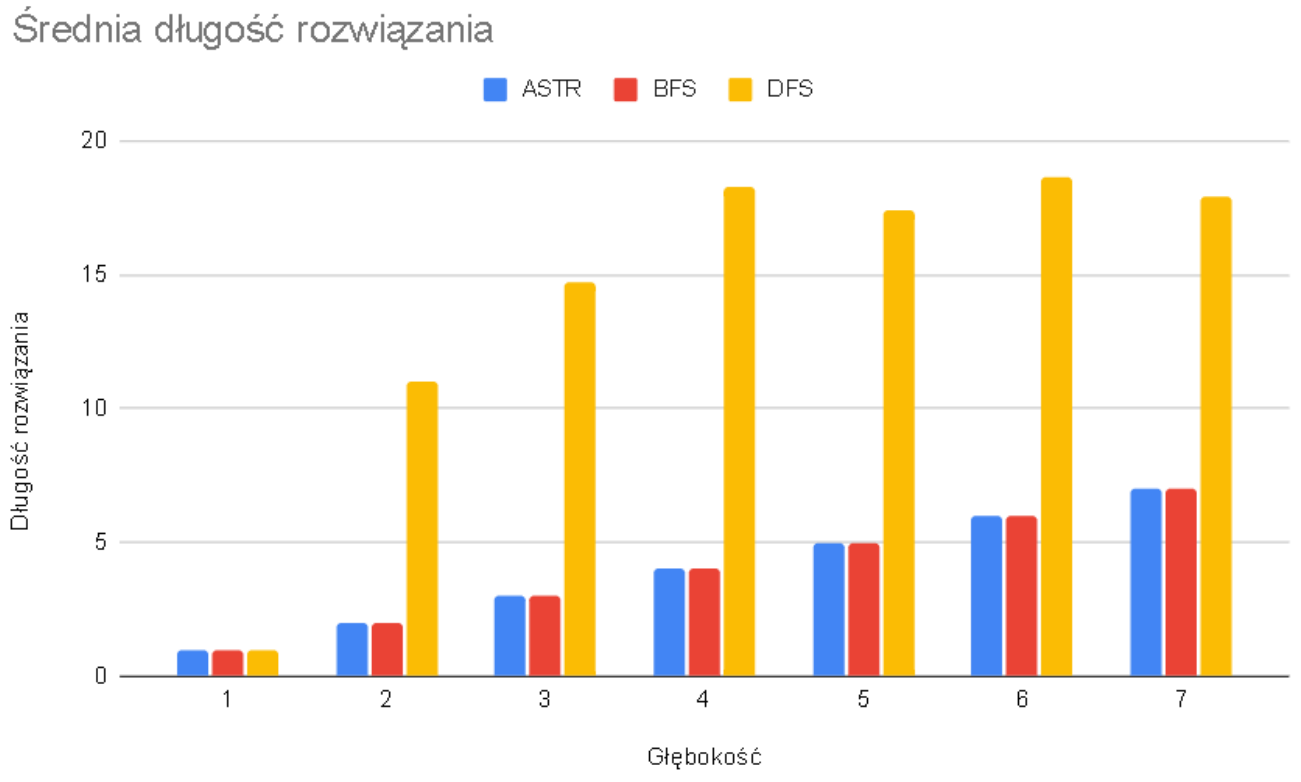
Uzyskane statystyki zostały zebrane również za pomocą udostępnionego skryptu, co pozwoliło wygenerować wykresy.

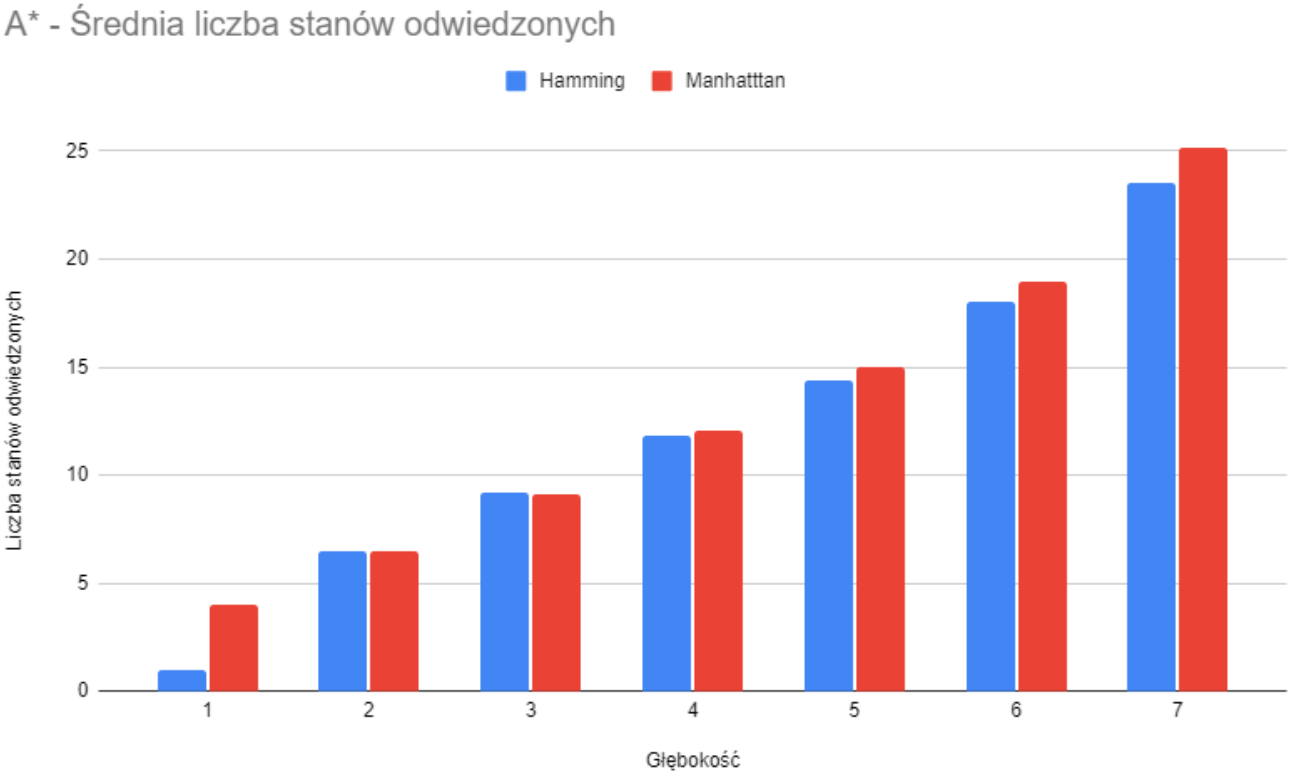
5. Wyniki

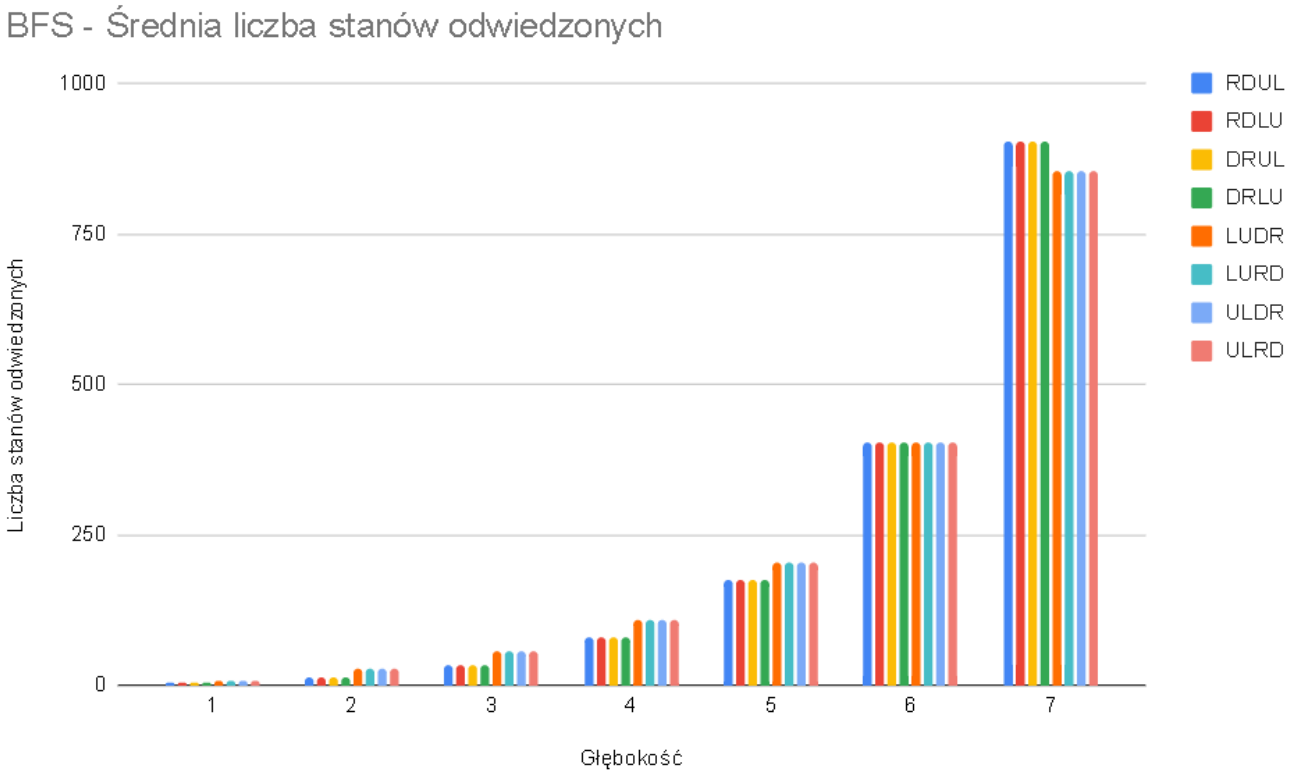

A* - Średnia długość rozwiązania
Hamming
Manhattan
Długość rozwiązania
0
1
2
3
4
5
6
7
1
2
3
4
5
6
7
Głębokość

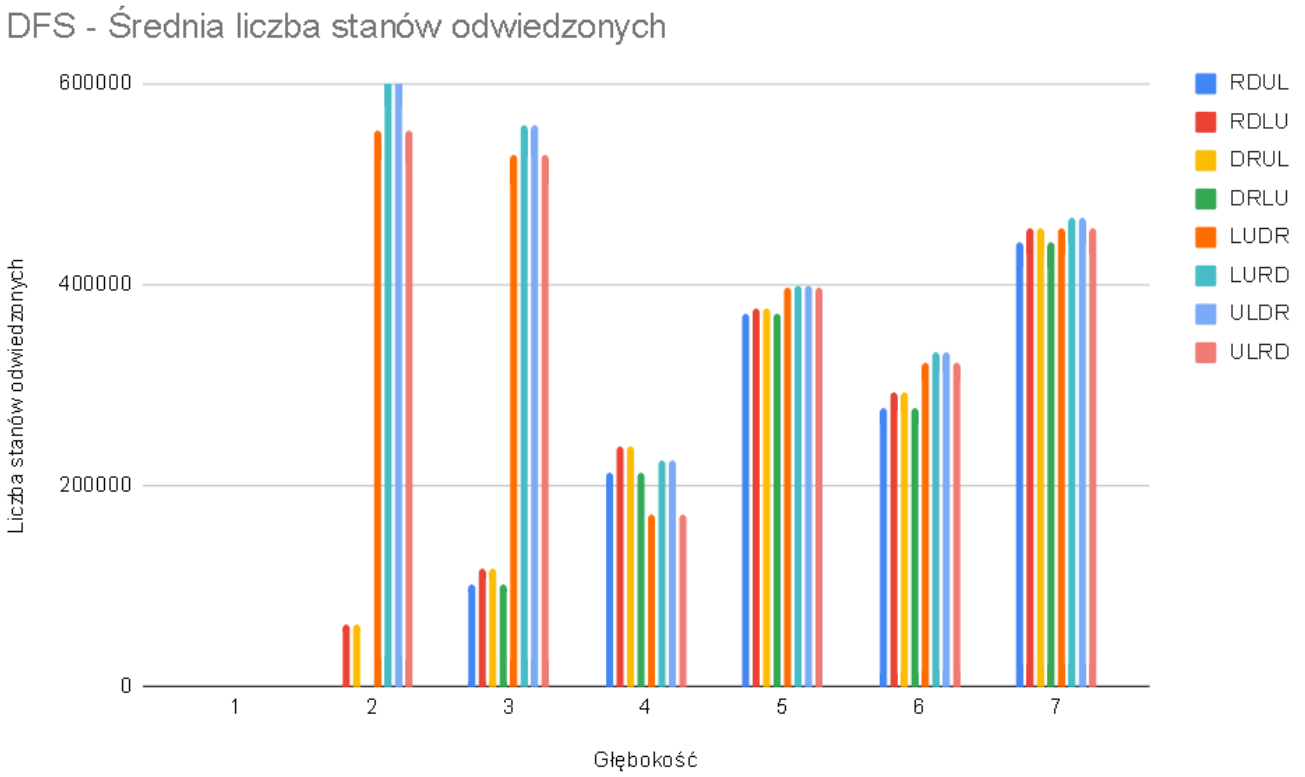

BFS - Średnia długość rozwiązania
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD
Długość rozwiązania
0
1
2
3
4
5
6
7
1
2
3
4
5
6
7
Głębokość

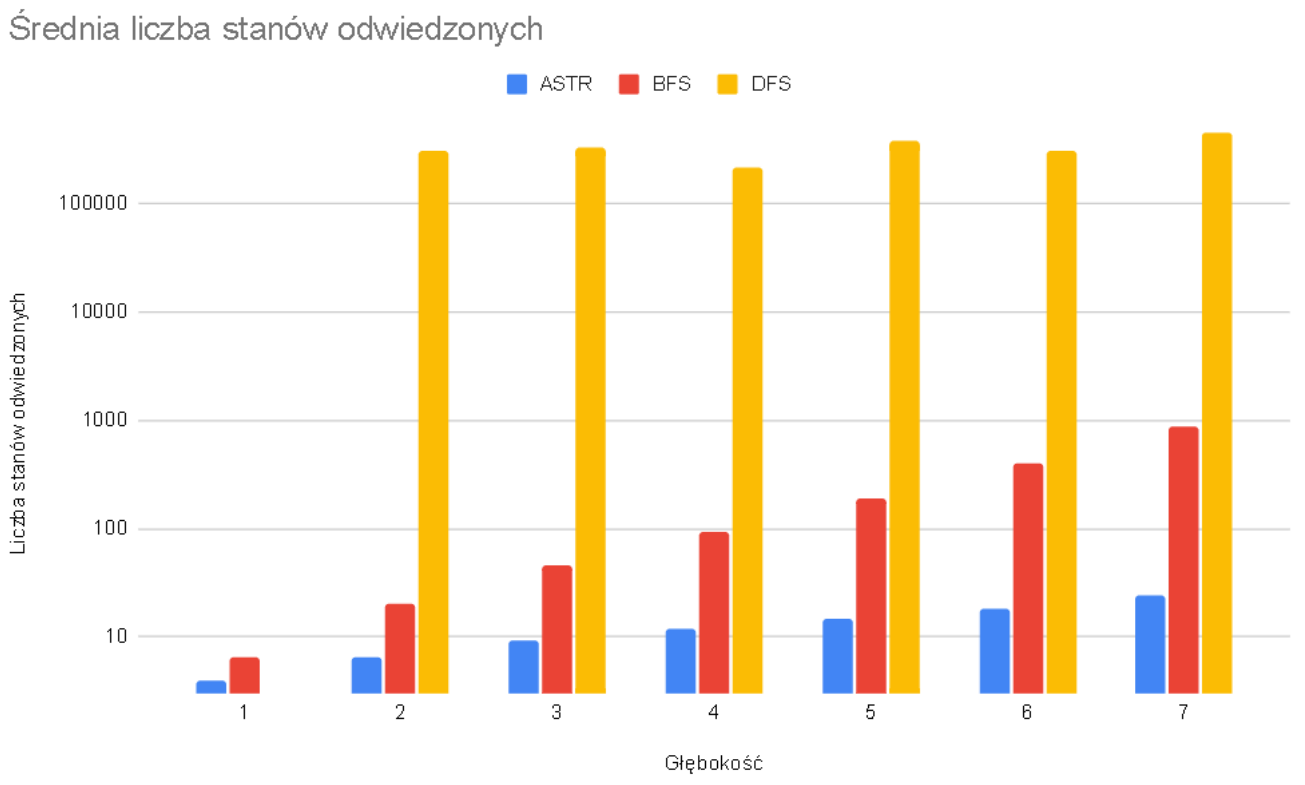

DFS - Średnia długość rozwiązania
20
15
10
5
0
Długość rozwiązania
1
2
3
4
5
6
7
Głębokość
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD

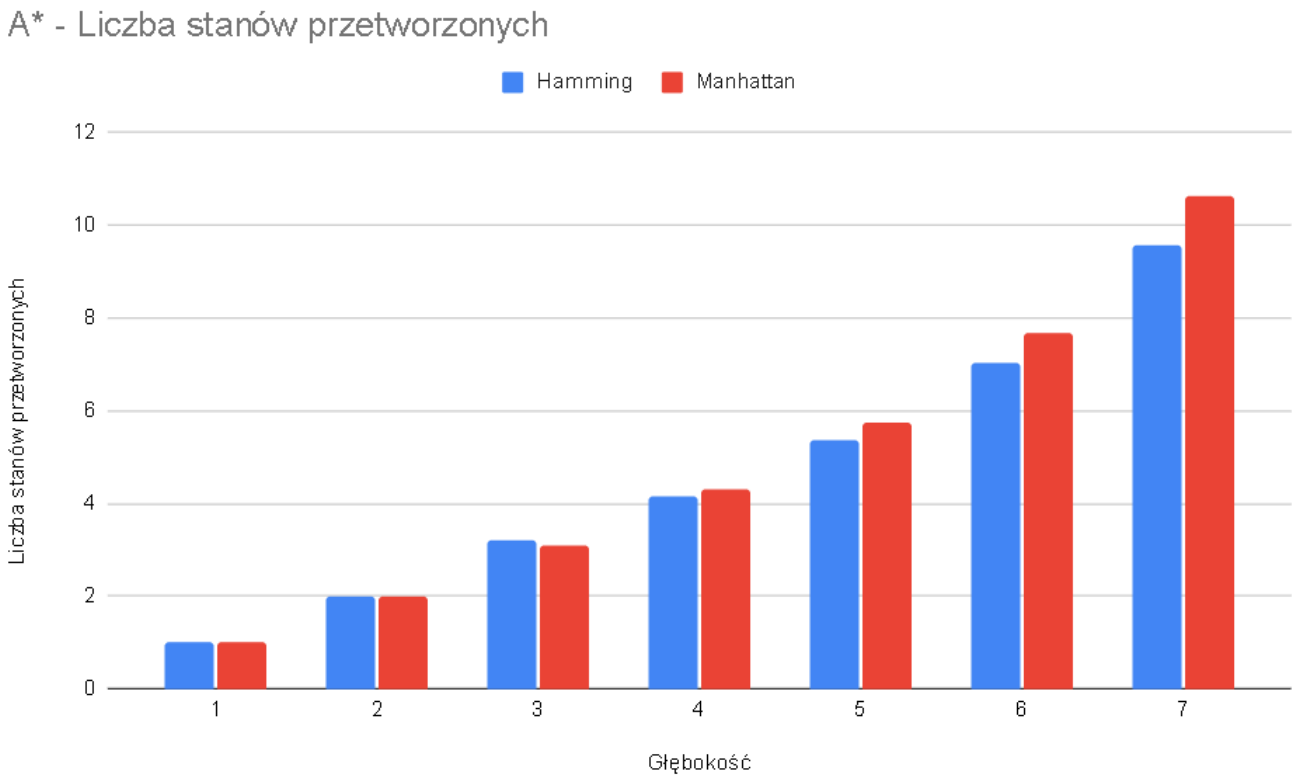

Średnia długość rozwiązania
ASTR
BFS
DFS
20
15
10
5
0
Długość rozwiązania
1
2
3
4
5
6
7
Głębokość

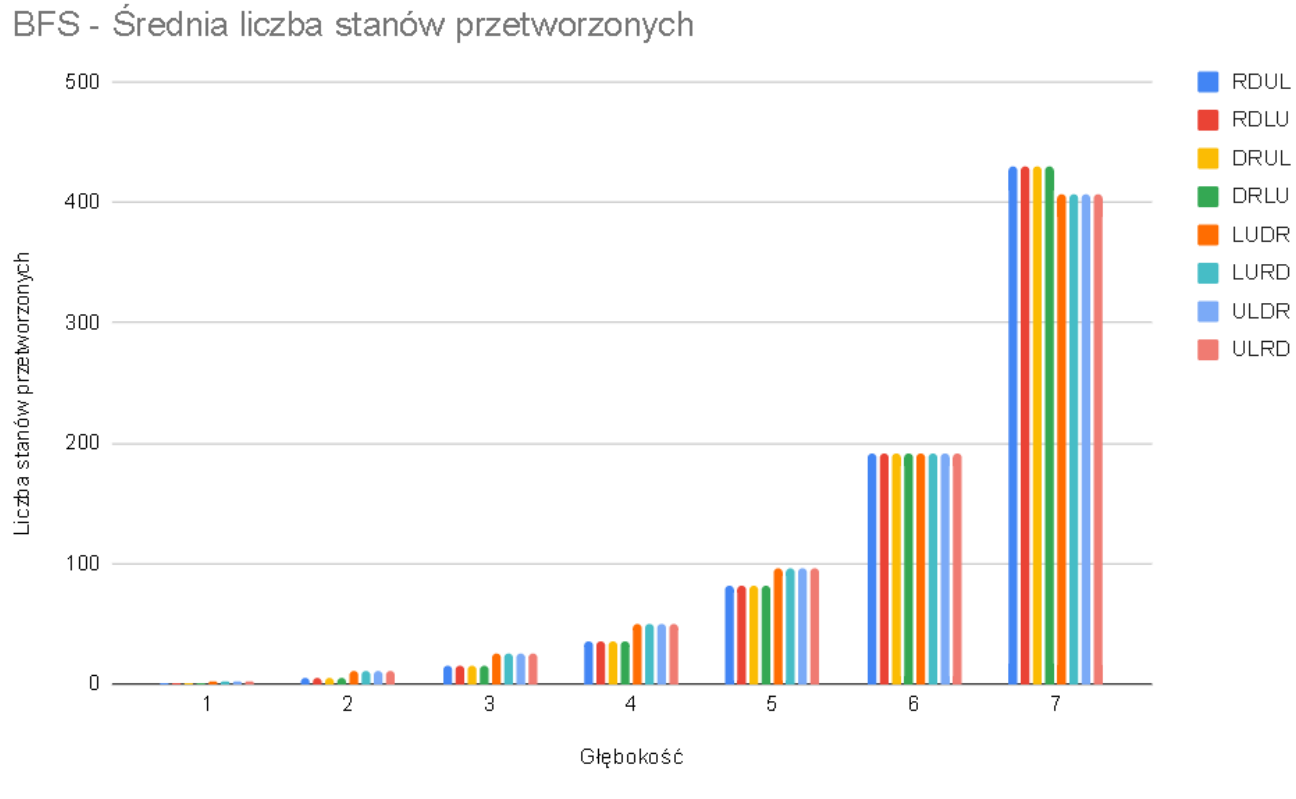

A* - Średnia liczba stanów odwiedzonych
Hamming
Manhatttan
25
20
15
10
5
0
Liczba stanów odwiedzonych
1
2
3
4
5
6
7
Głębokość


BFS - Średnia liczba stanów odwiedzonych
1000
750
500
250
0
Liczba stanów odwiedzonych
1
2
3
4
5
6
7
Głębokość
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD

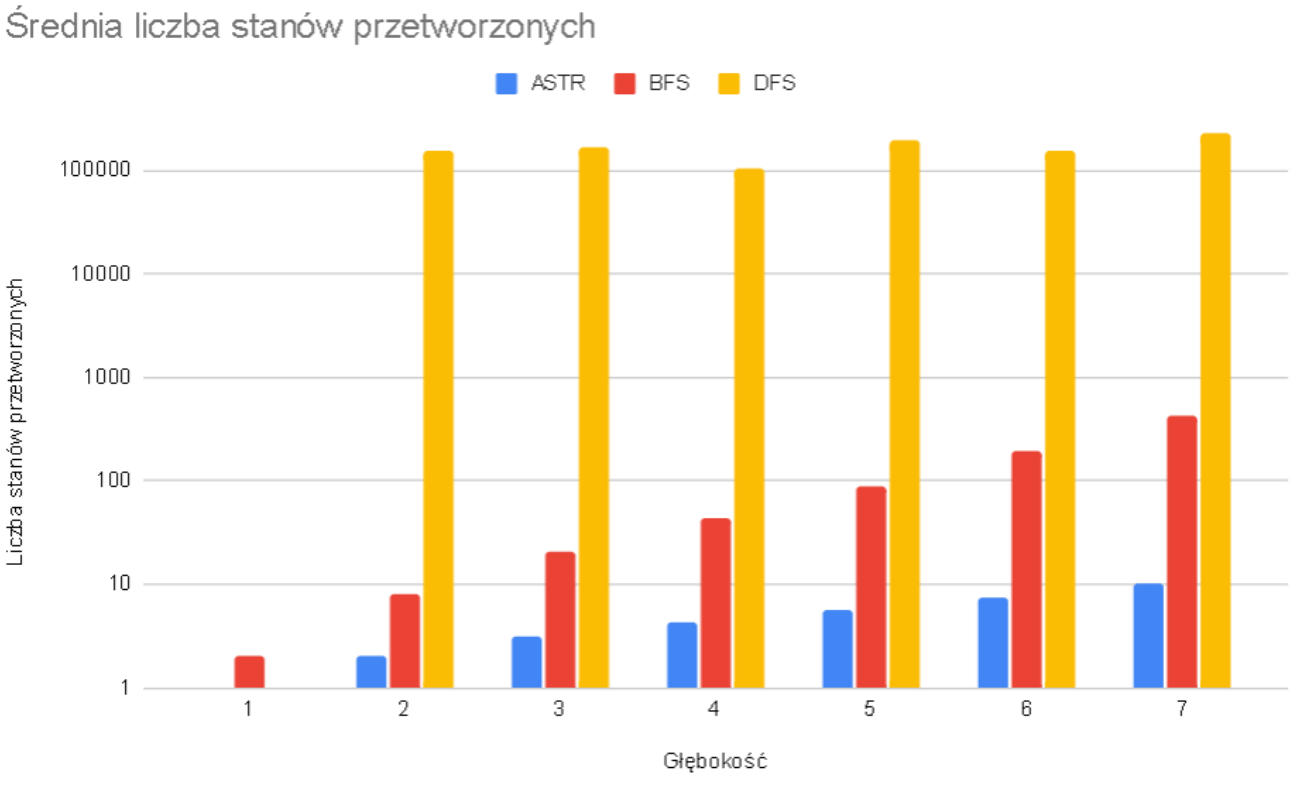

DFS - Średnia liczba stanów odwiedzonych
600000
400000
200000
0
Liczba stanów odwiedzonych
1
2
3
4
5
6
7
Głębokość
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD

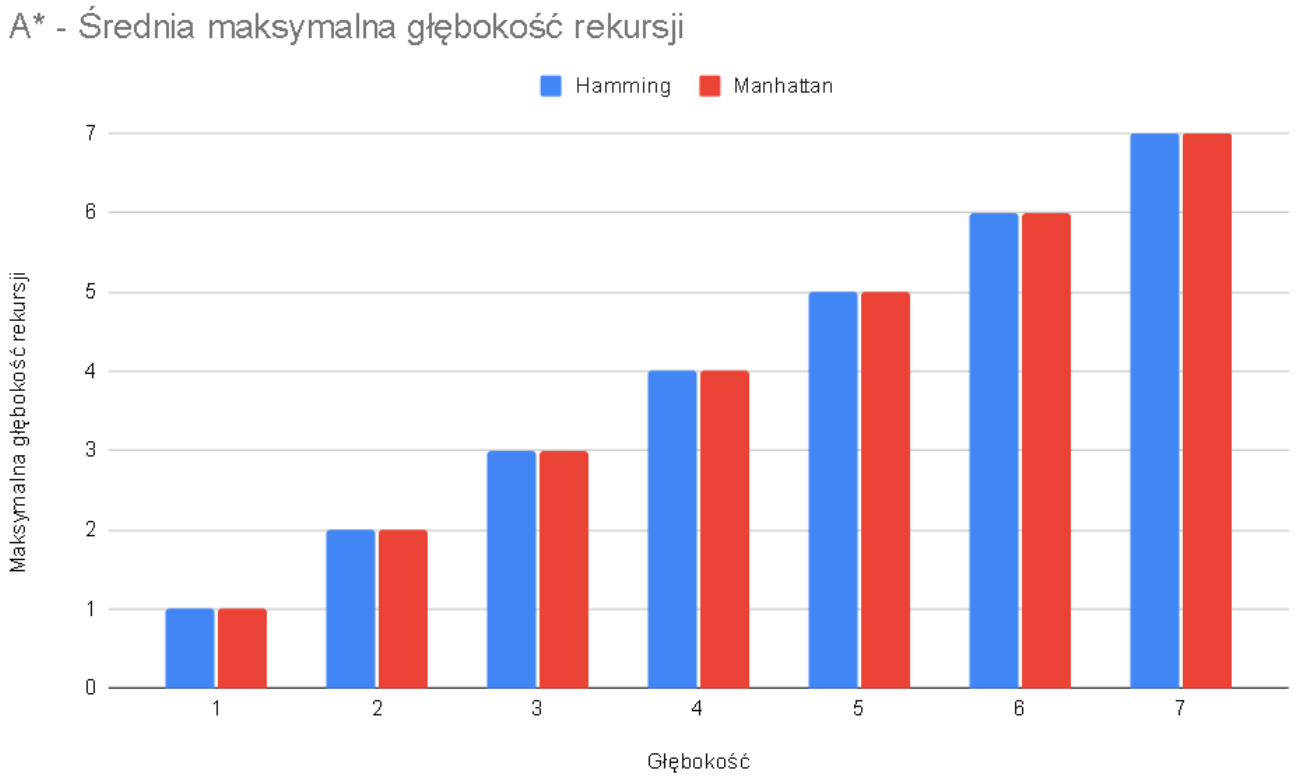

Średnia liczba stanów odwiedzonych
ASTR
BFS
DFS
100000
10000
1000
100
10
Liczba stanów odwiedzonych
1
2
3
4
5
6
7
Głębokość

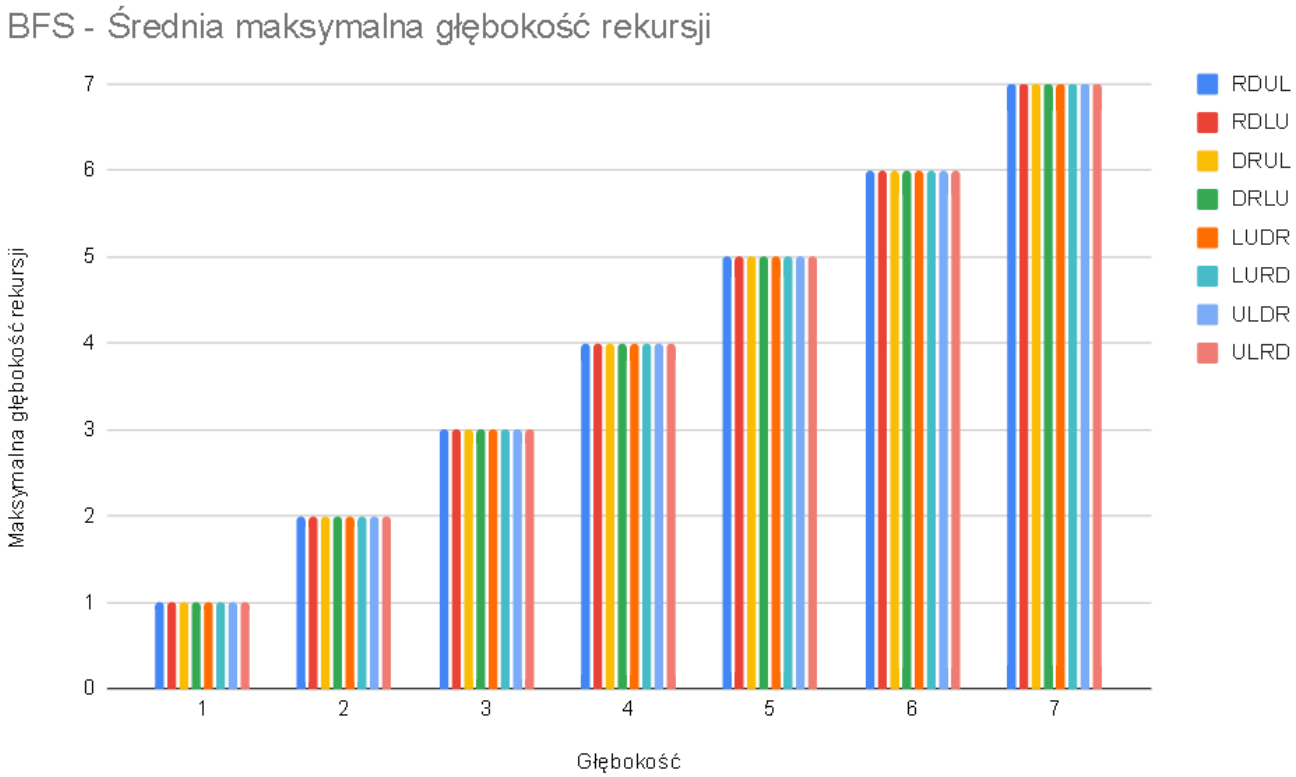

A* - Liczba stanów przetworzonych
Hamming
Manhattan
12
10
8
6
4
2
0
Liczba stanów przetworzonych
1
2
3
4
5
6
7
Głębokość

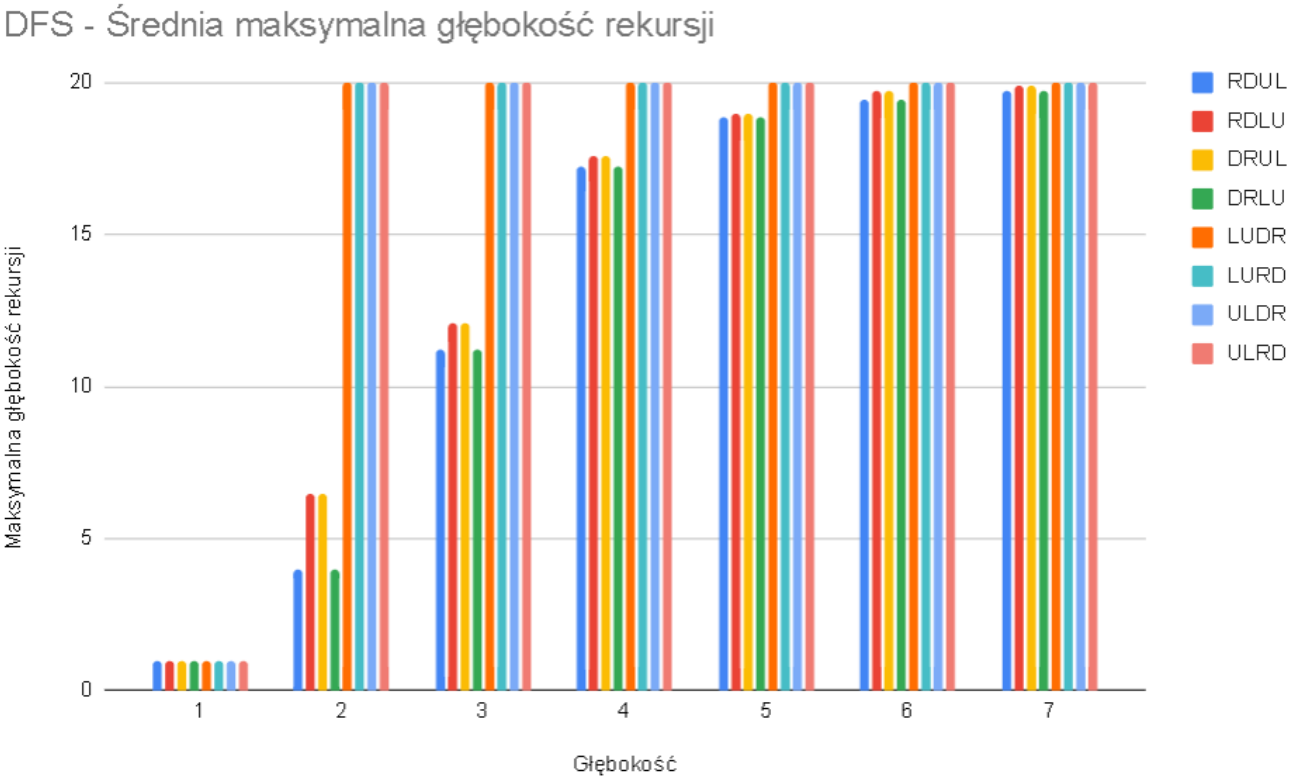

BFS - Średnia liczba stanów przetworzonych
500
400
300
200
100
0
Liczba stanów przetworzonych
1
2
3
4
5
6
7
Głębokość
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD

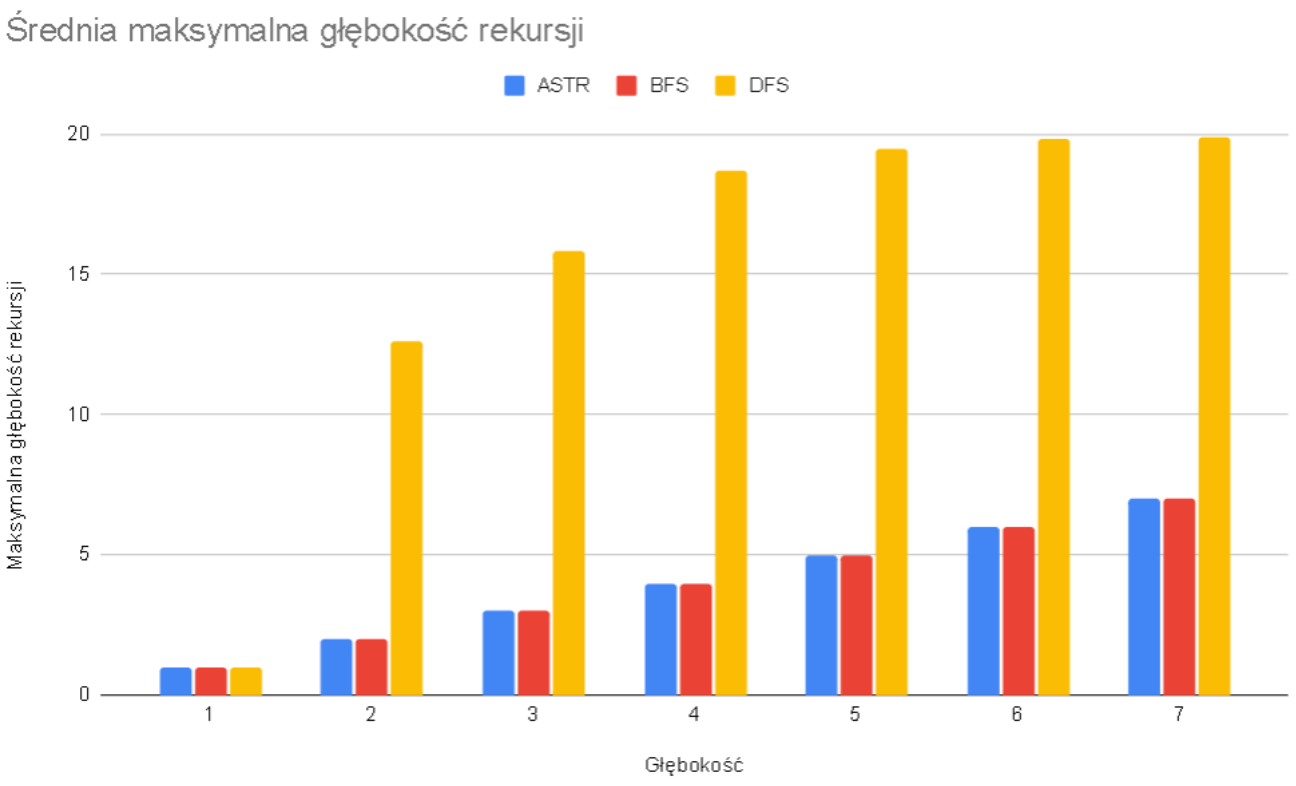

DFS - Średnia liczba stanów przetworzonych
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD
100000
10000
1000
100
10
1
Liczba stanów przetworzonych
1
2
3
4
5
6
7
Głębokość

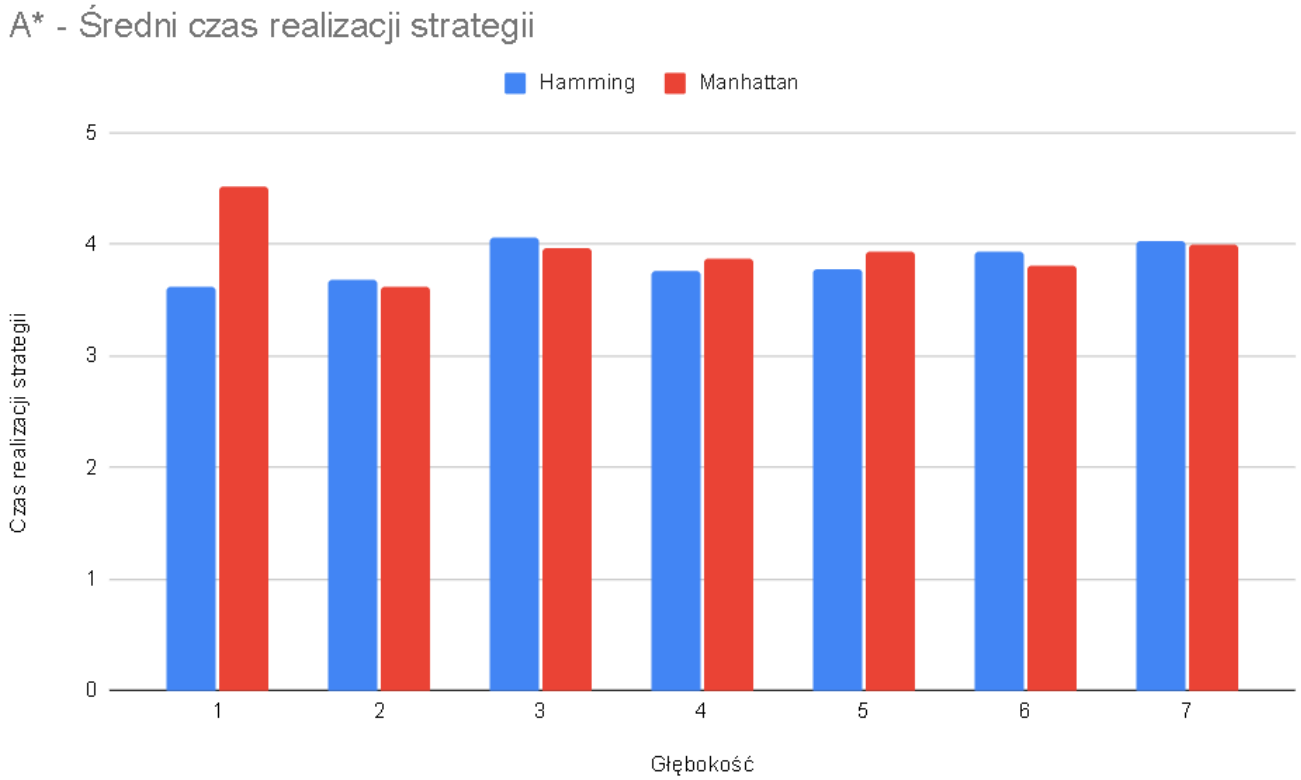

Średnia liczba stanów przetworzonych
ASTR
BFS
DFS
100000
10000
1000
100
10
1
Liczba stanów przetworzonych
1
2
3
4
5
6
7
Głębokość

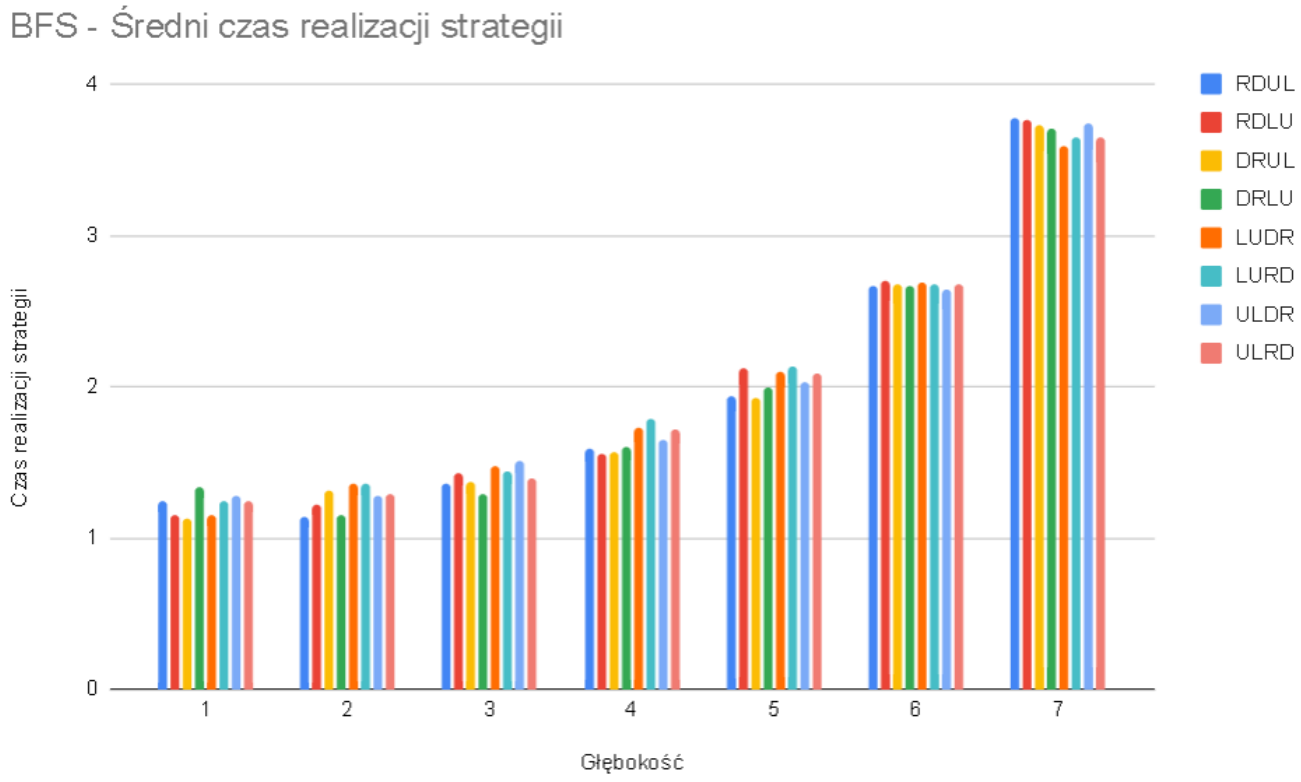

A* - Średnia maksymalna głębokość rekursji
Hamming
Manhattan
Maksymalna głębokość rekursji
0
1
2
3
4
5
6
7
1
2
3
4
5
6
7
Głębokość

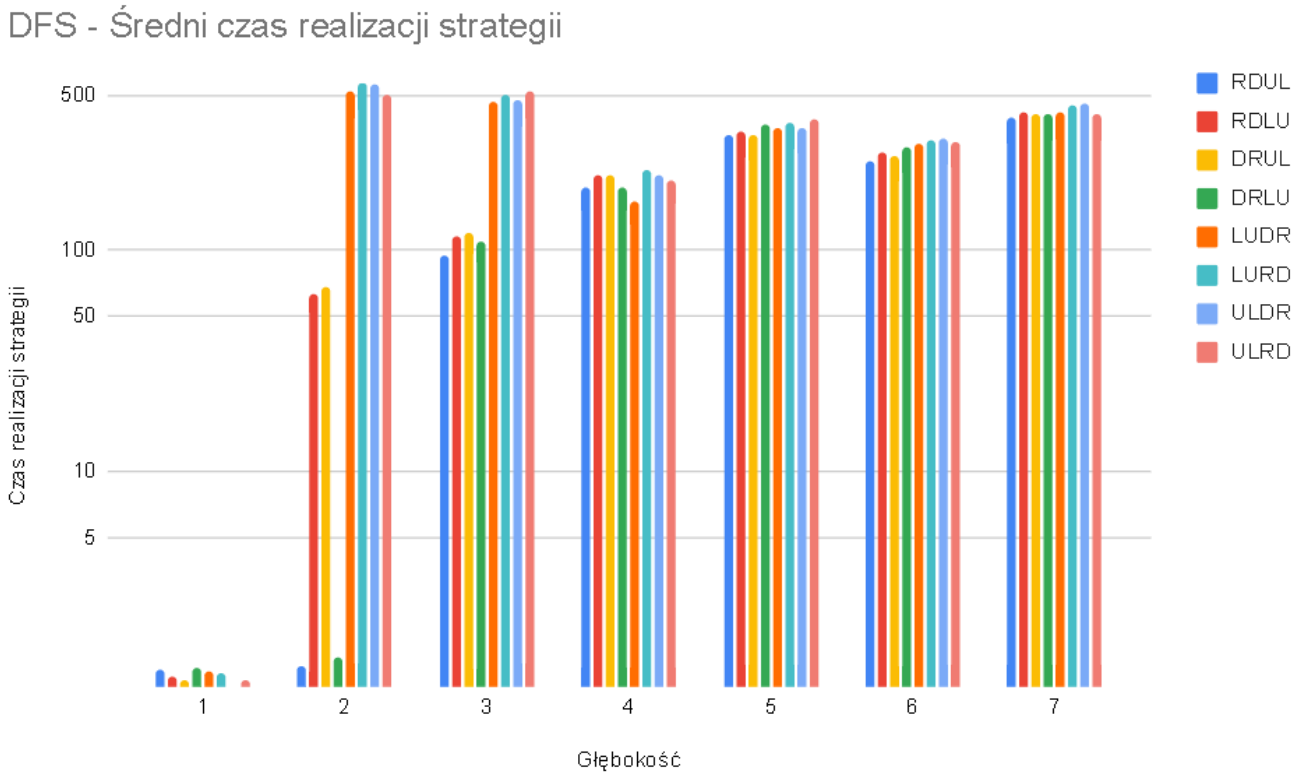

BFS - Średnia maksymalna głębokość rekursji
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD
Maksymalna głębokość rekursji
0
1
2
3
4
5
6
7
1
2
3
4
5
6
7
Głębokość

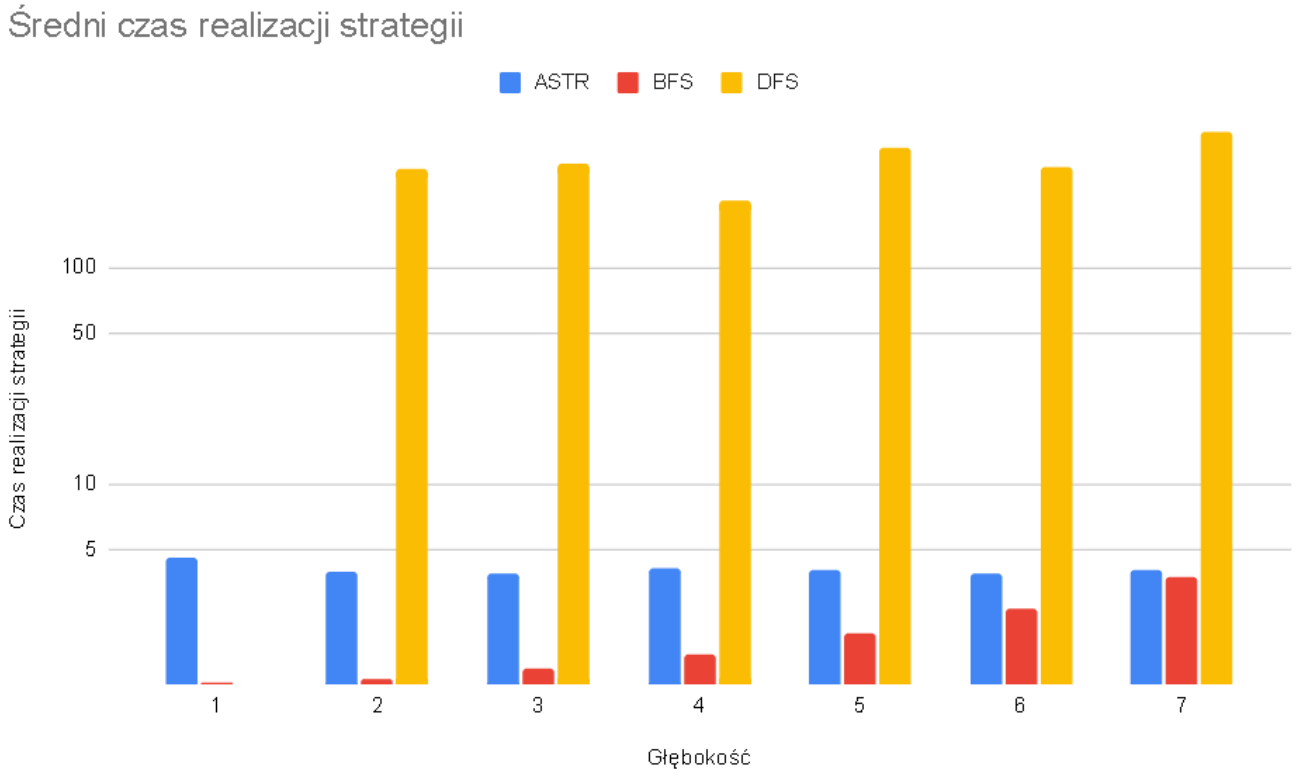

DFS - Średnia maksymalna głębokość rekursji
Maksymalna głębokość rekursji
20
15
10
5
0
1
2
3
4
5
6
7
Głębokość
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD


Średnia maksymalna głębokość rekursji
ASTR
BFS
DFS
Maksymalna głębokość rekursji
20
15
10
5
0
1
2
3
4
5
6
7
Głębokość


A* - Średni czas realizacji strategii
Hamming
Manhattan
Czas realizacji strategii
0
1
2
3
4
5
1
2
3
4
5
6
7
Głębokość


BFS - Średni czas realizacji strategii
RDUL
RDLU
DRUL
DRLU
LUDR
LURD
ULDR
ULRD
Czas realizacji strategii
0
1
2
3
4
1
2
3
4
5
6
7
Głębokość





6. Dyskusja

Maksymalna głębokość rekursji równa odległości układanek świadczy o tym, że A* i BFS szukają rozwiązania optymalnego, o minimalnym koszcie. W A* wynika to z działania heurystyki, która wybiera stany najbliższe rozwiązaniu, a w BFS - z góry ustalonego porządku, który nie pozwoli przetwarzać stanów o koszcie wyższym przed przeanalizowaniem wszystkich stanów o koszcie niższym. Dla metody DFS rozwiązania były optymalne tylko dla odległości 1, a od odległości 4 ich długość

zbliżała się do maksymalnej dopuszczalnej głębokości rekursji w niektórych ją osiągając. Wszystkim strategiom udało się rozwiązać każdy przypadek testowy. Dla odległości 1 czasy wykonania najniższe są dla algorytmu DFS, a najwyższe dla A*. Dzieje się tak dlatego, że jesteśmy bardzo blisko rozwiązania i znajdujemy je w maksymalnie czterech ruchach, a A* i tak musi wykonać swoje obliczenia na podstawie wybranej heurystyki. Dla odległości 2 - 7 najszybciej działa algorytm BFS, najgorzej wypada DFS. Dla odległości równej 7 A* oraz BFS osiągają zbliżone czasy, im większa jest odległość tym bardziej BFS traci swoją przewagę nad metodą A*. Analizując wykres jesteśmy w stanie przypuścić, iż dla większych głębokości metoda przeszukiwania 'wszerz' straci miano najszybszego rozwiązywania łamigłówki na rzecz heurystyki. Dla DFS wszystkie czasy dla odległości poza 1 są zbliżone jest to spowodowane przetwarzaniem kolejnych stanów z dala od rozwiązania, bez względu na to, jak blisko było ono na początku. Łączna liczba stanów odwiedzonych i przetworzonych w A* była najniższa ze wszystkich strategii Wynika to z faktu, że w A* kolejność stanów do przetwarzania ustalana jest na podstawie heurystyki, a nie wyłącznie na podstawie z góry ustalonego porządku, tak jak ma to miejsce w strategiach BFS i DFS. Dla BFS liczby te i tak są wiele razy niższe niż dla DFS. W BFS nigdy nie przekroczymy głębokości. W DFS jeśli nie znajdziemy rozwiązania w pierwszych kilku iteracjach to odbiegamy od niego bardzo daleko, do maksymalnej dopuszczalnej głębokości rekursji, przetwarzając kolejne stany w okolicach tej głębokości, aż do trafienia rozwiązania. BFS musi przeszukać wszystkie stany na danej odległości, a ich ilość wzrasta bardzo znacznie z każdą kolejną, dlatego też im dalej jest tym gorzej sobie radzi (jak każdy z algorytmów, ale u BFS jest to znacznie bardziej widoczne, dla odległości powyżej 5 wyniki robią się kilkukrotnie większe). Niektóre pary porządków operatorów osiągnęły identyczne rezultaty bez względu na użyty algorytm, są to: (LUDR, ULRD), (DRLU, RDUL), (LURD, ULDR), (DRUL, RDLU). Dla DFS różnice w niektórych porządkach są bardzo duże jest to spowodowane tym, że ostatnie ruchy prowadzące do rozwiązania to zawsze D oraz R, więc porządki zawierające te operatory jako pierwsze mają przewagę co widać na wykresach. Dla algorytmu A* różnice w Heurystykach nie były duże. Dla odległości poniżej 4 różnice pomiędzy obiema heurystykami są bardzo małe i trudno wyciągnąć jakiekolwiek wnioski. Dla odległości 4 i powyżej możemy za to zauważyć przewagę heurystyki Hamminga, w ilości stanów oraz w czasie wykonania. Są to jednak dość małe różnice i ciężko wykazać ich przyczynę.

7. Wnioski

— Wszystkim 3 strategiom : A*, BFS, DFS udało się rozwiązać każdy z zadanych przypadków testowych.
— Wydajność strategii BFS znacząco spada wraz ze wzrostem odległości od układu wzorcowego..
— DFS jest strategią najmniej optymalną i juz na początku potrafi odbiec bardzo daleko od rozwiązania co powoduje przetwarzanie dużej ilości stanów.
— Strategia DFS jest najwolniejsza.
— Strategia A* poza najprostszymi przypadkami jest najszybsza i jej przewaga rośnie wraz ze wzrostem odległości od układu wzorcowego.
— W strategii DFS nie ma zależności pomiędzy czasem, a odległością od układu wzorcowego.
— W strategii A* różnice pomiędzy heurystykami są nieznaczne.
— Porządki operatorów z R oraz D na początku sprawdzają się dużo lepiej ze względu na to iż ostatnie ruchy to zawsze R oraz D.

.

Literatura

[1] https://pl.wikipedia.org/wiki/Pi%C4%99tnastka_(uk%C5%82adanka)
[2] https://ftims.edu.p.lodz.pl/mod/page/view.php?id=47860
[3] https://ftims.edu.p.lodz.pl/course/view.php?id=16
[4] https://pl.wikipedia.org/wiki/Przestrze%C5%84_metryczna

[5] https://pl.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/the-breadth-first-search-algorithm