Informatyka, studia dzienne, inż. I st. semestr VI

Sztuczna inteligencja i systemy ekspertowe 2016/2017

Prowadzący: dr inż. Krzysztof Lichy piątek, 10:15

Data oddania: _____ Ocena: _____

Maciej Ślusarz 189789

Alicja Gałkiewicz 195589

Zadanie 1: Piętnastka

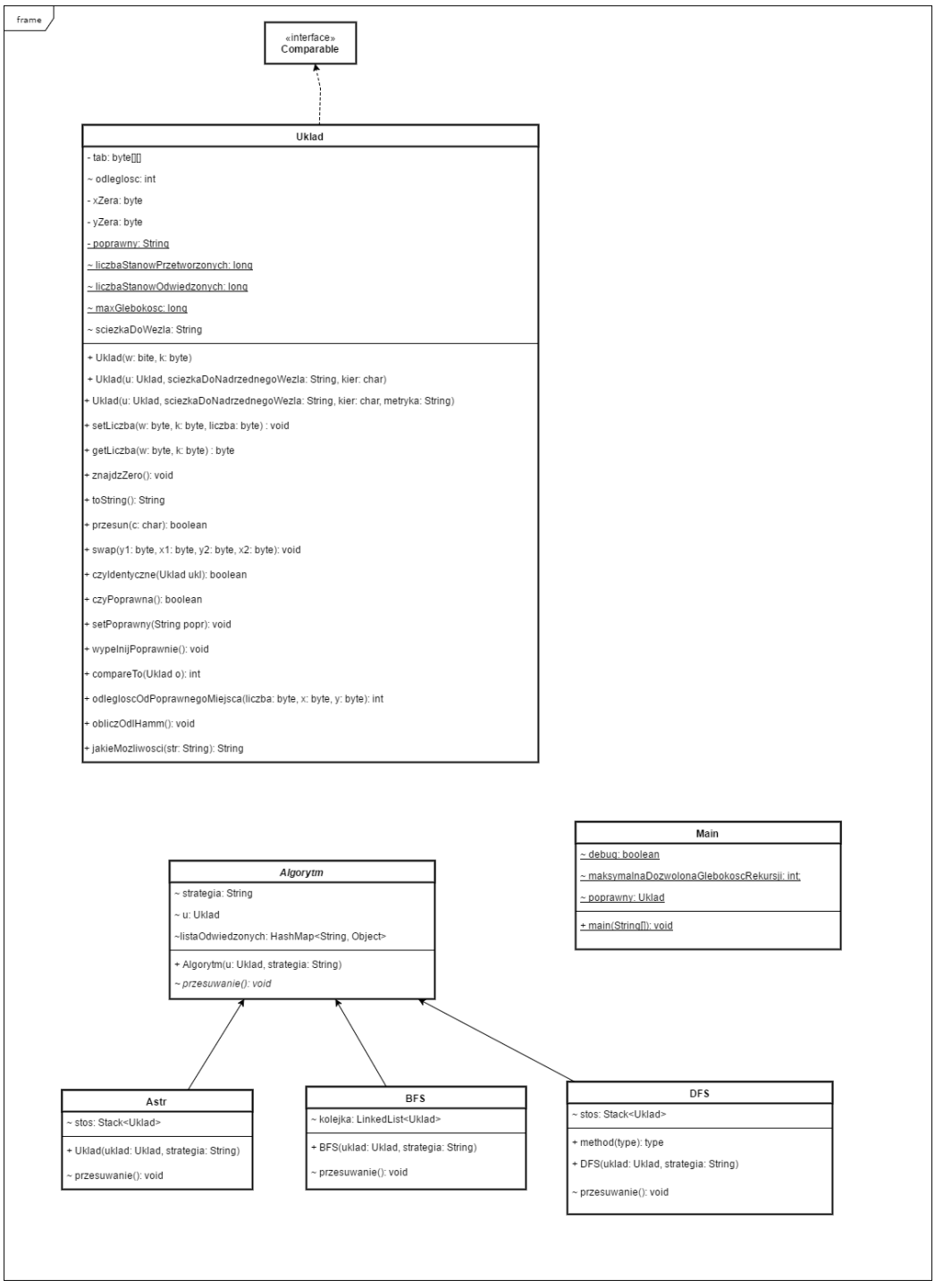
1. Cel

Celem zadania było napisanie programu implementującego różne algorytmy rozwiązywania układanki – „przesuwanki”, zwanej „piętnastką”, oraz zbadanie, jak poszczególne metody wpływają na pewne końcowe parametry. Metody przeszukiwania to: przeszukiwanie wszerz (BFS, ang. breadth-first serach), przeszukiwanie w głąb (DFS, ang. depth-first serach), oraz odmiany DFS polegające na wyborze najpierw najlepszego układu. Program został napisany w języku Java.

2. Wprowadzenie

Aby znaleźć ciąg ruchów pozwalający na przekształcenie układanki z pewnego stanu początkowego do poprawnego układu, oczywistym sposobem jest sprawdzenie każdej możliwej kombinacji ruchów. Przy dowolnym układzie liczb na układance, możliwe do wykonania są (w przypadku prostokątnej układanki) 2 albo 3 albo 4 ruchy, polegające na zamianie pustego pola z liczbą sąsiadującą z pustym polem. Zamianę pustego pola z liczbą znajdującą się wyżej będziemy nazywali przesunięciem w górę, zamianę pustego pola z liczbą znajdującą się po lewej stronie będziemy nazywali przesunięciem w lewo, itp. W przypadku, gdy puste pole znajduje się przy krawędzi układanki, liczba możliwych ruchów jest mniejsza. Z danego układu można więc „stworzyć” do 4. innych układów. Można to przedstawić w postaci grafu, gdzie układem początkowym będzie wierzchołek-ojciec, a układami pochodnymi będą potomkowie tego wierzchołka. Można przyjąć, że krawędziami w takim grafie są kierunki, wyznaczające ścieżkę do węzła. Kolejność przeszukiwania węzłów w takim grafie można różnie wybierać. Sprawdzenie każdego z układów pochodnych danego węzła czy jest poprawny, a następnie wykonanie tej samej czynności dla wszystkich węzłów pochodnych nazywane jest przeszukiwaniem wszerz (BFS). Sprawdzenie, czy poprawnym układem jest pierwszy z możliwych niesprawdzonych wcześniej ruchów, a następnie wykonanie tej samej czynności dla otrzymanego węzła pochodnego nazywane jest przeszukiwaniem w głąb (DFS). Dodatkowo, odmianą przeszukiwania w głąb jest użycie pewnej metody oceny możliwych pochodnych układów, i rozpoczęcie od układu który jest pod pewnym względem najlepszy. Miarą oceny takiego układu jest liczba, określająca odległość układu od poprawnego – wzorcowego układu. Odległość w przypadku macierzy szesnastu liczb może być mierzona w metryce Hamminga, albo w metryce Manhattan. Pierwsza metoda polega na określeniu, ile liczb znajduje się w nieodpowiednim miejscu. Druga metoda polega na określeniu, jak daleko każda z liczb znajduje się od odpowiedniego miejsca.

3. Opis implementacji



Program został napisany w języku Java.

Jedną z najistotniejszych klas z punktu widzenia kodu jest klasa Uklad, będąca modelem pojedynczego węzła w grafie. Liczby reprezentujące układ są przechowywane w tablicy liczb całkowitych, gdzie 0 oznacza puste pole. Dodatkowymi polami tej klasy są na przykład zmienne określające

współrzędne pustego pola - po to, aby nie trzeba było "znajdować" go za każdym razem przed przesunięciem w celu wykonania ruchu. Każdy obiekt typu Uklad zawiera również ciąg liter określający ścieżkę do węzła - sekwencję ruchów odpowiadającą przekształceniom układu początkowego do układu reprezentowanego przez węzeł. W przypadku użycia algorytmu z heurystyką używana jest dodatkowo zmienna odleglosc, określająca ocenę tego jak blisko od układu poprawnego jest układ. Klasa Uklad ma zaimplementowane operacje umożliwiające zamodelowanie przesuwania liczb po planszy - czyli zamianę pustego pola z którymś z sąsiadujących pól, określanie możliwych do wykonania ruchów, oraz sprawdzanie, czy układ jest poprawny. W przypadku użycia algorytmu z heurystyką używane są dodatkowo operacje obliczające odległość układu od układu poprawnego, jedną z dwóch możliwych metod - w zależności od strategii. Jedną z ważniejszych metod tej klasy jest również konstruktor, przyjmujący jako argument kierunek według którego ma zostać utowrzony nowy obiekt typu Uklad. Dodatkowo, klasa Uklad zawiera informacje przydatne później do umieszczenia w pliku ze statystykami - liczbę stanów odwiedzonych, liczbę stanów przetworzonych, maksymalną osiągniętą głębokość rekursji - te zmienne są zmiennymi statycznymi, czyli każda zmienna jest pojedyncza dla całej klasy, a nie dla każdego z obiektów osobno. Trzema klasami odpowiadającymi za implementację algorytmów DFS, BFS i Astr są klasy nazwane właśnie tymi słowami. Dziedziczą one po klaaie abstrakcyjnej Algorytm, którego jedyną metodą jest przesuwanie(), a różnice w implementacji wynikają ze stosowanej kolejności przeszukiwania węzłów w grafie. Klasa implementująca algorytm DFS zawiera iteracyjną postać implementacji przeszukiwania w głąb. Struktura danych użyta w tej klasie to stos, na który odkładane są kolejne tworzone wierzchołki. Użycie stosu, który jest strukturą danych typu LIFO, zapewnia odpowiednią kolejność sprawdzania wierzchołków. Klasa implementująca algorytm BFS zawiera implementację przeszukiwania wszerz. Struktura danych użyta w tej klasie to kolejka typu FIFO, której użycie zapewnia odpowiednią kolejność sprawdzania wierzchołków. Klasa implementująca algorytm z heurystykami zawiera te same ciągi intrukcji co klasa DFS, jednakże, dodatkowo znajdują się tam fragmenty kodu odpowiadające za odpowiednie posortowanie i odłożenie na stos wierzchołków w założonej kolejności. Pierwszymi instrukcjami wykonywanymi po uruchomieniu programu są: odpowiednie ustawienie zmiennych określających parametry - w zależności od argumentów wywołania programu. Następnie, wczytywany jest plik tekstowy reprezentujący układ początkowy. Generowany jest układ poprawny, z którym można porównywać analizowane wierzchołki grafów. Następnie, wywoływany jest odpowiedni algorytm. Dodatkowymi fragmentami kodu są fragmenty odpowiedzialne za obsługę wyjątków związanych z odczytem / zapisem plików, obliczanie czasu działania algorytmów, itp.

4. Materiały i metody

Na początku wygenerowaliśmy pliki tekstowe, zawierające wszystkie możliwe stany układanki o rozmiarach 4 na 4 do maksymalnej głębokości rozwiązania 7, korzystając z generatora udostępnionego na stronie przedmiotu, na platformie WIKAMP.

Pliki ze statystykami otrzymaliśmy uruchamiając nasz program z następującymi parametrami:

— Algorytm - przekazaliśmy tutaj wszystkie zaimplementowane przez nas algorytmy znajdujące rozwiązania: DFS, BFS, A*

— Strategia - w przypadku DFS i BFS jest to porządek przeszukiwania sąsiedztwa: RDUL, RDLU, DRUL, DRLU, LUDR, LURD, ULDR, ULRD (każda z liter oznacza sąsiada, który powstaje poprzez przesunięcie wolnego pola w odpowiednim kierunku: U - góra, D - dół, L - lewo, R - prawo), a w przypadku A* - funkcja heurystyczna: hamm (odległość Hamminga), manh (odległość Manhattan).

— Plik z układanką - tutaj podaliśmy ścieżki do wcześniej wygenerowanych plików z wariantami układanki

— Plik z rozwiązaniem - zawiera ścieżkę do pliku, w którym zapisywane będą informacje o rozwiązaniu - jego długości oraz listy ruchów pustego pola

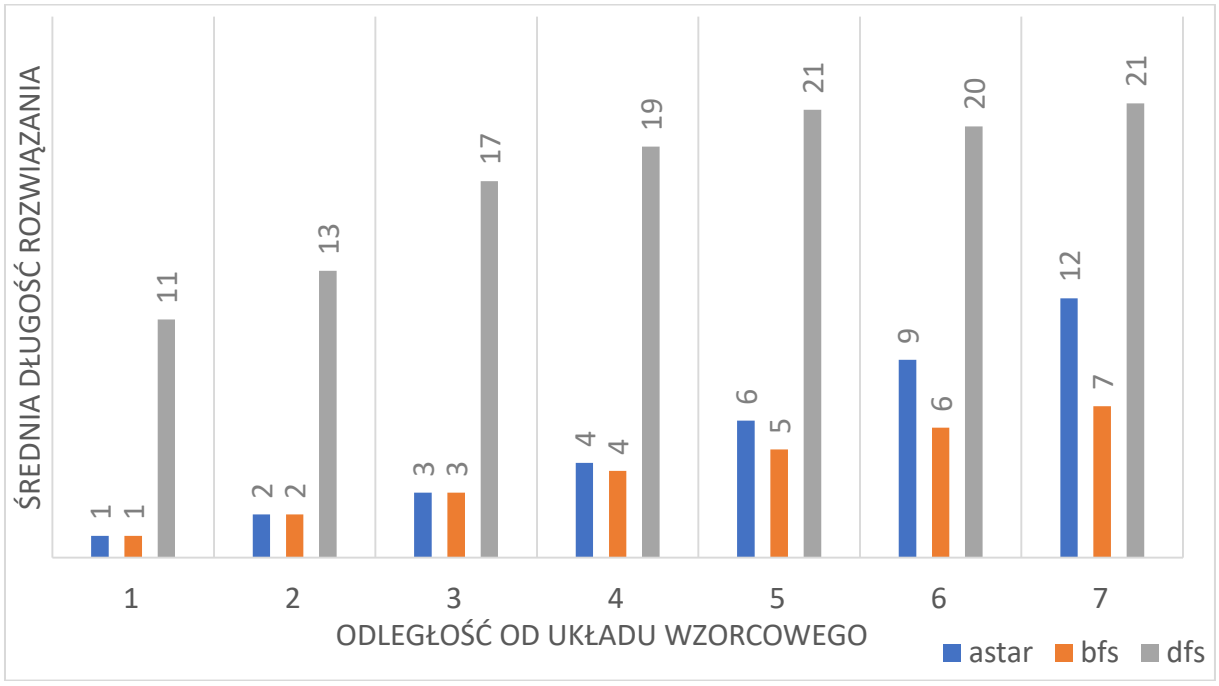
— Plik ze statystykami - zawiera ścieżkę do pliku, w którym zapisywane będą statystyki dotyczące rozwiązania - długość rozwiązania, maksymalną osiągniętą głębokość rekursji, czas wykonania programu, liczbę wierzchołków odwiedzonych oraz liczbę wierzchołków przetworzonych.

W kolejnej sekcji sprawozdania porównamy następujące statystyki:

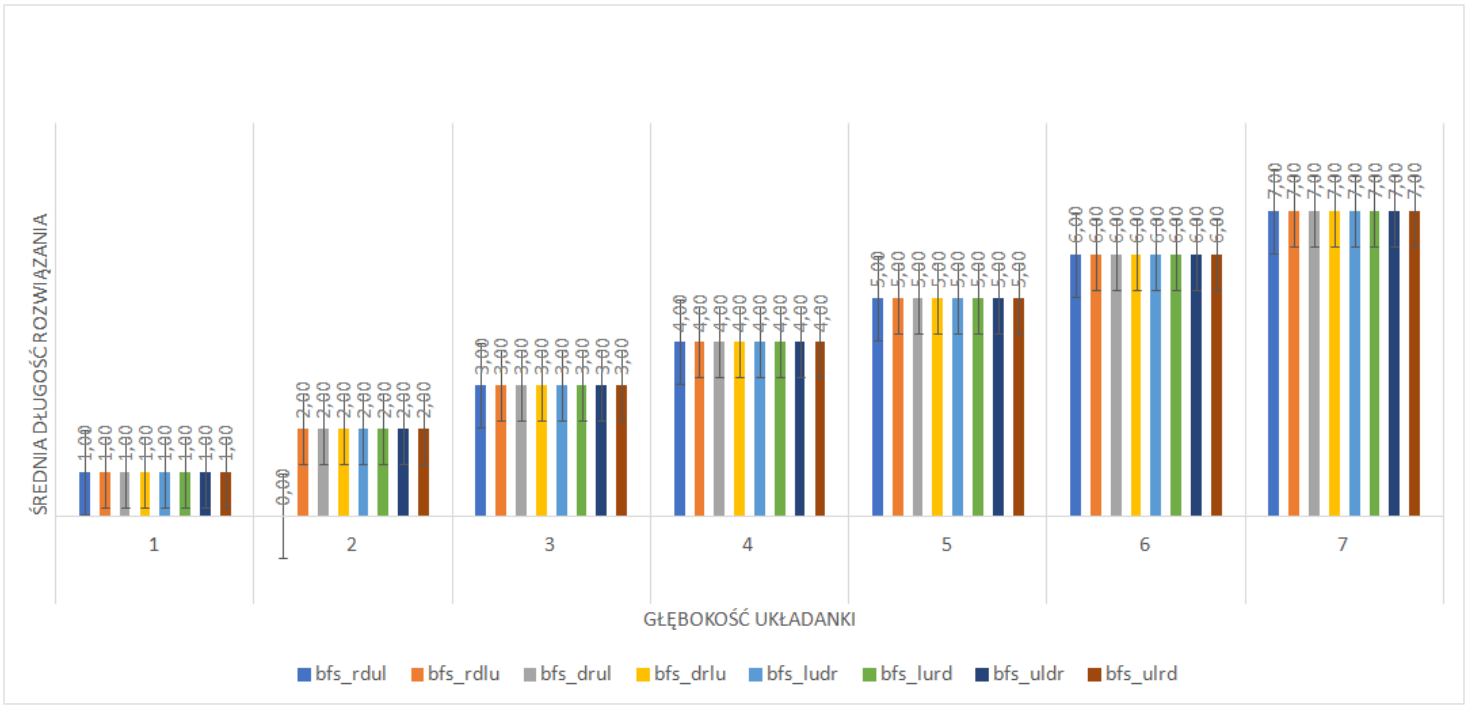
1. Średnią długość znalezionego rozwiązania
2. Średnią maksymalną głębokość rekursji
3. Średni czas znalezienia rozwiązania
4. Średnią liczbę wierzchołków odwiedzonych
5. Średnią liczbę wierzchołków przetworzonych

5. Wyniki

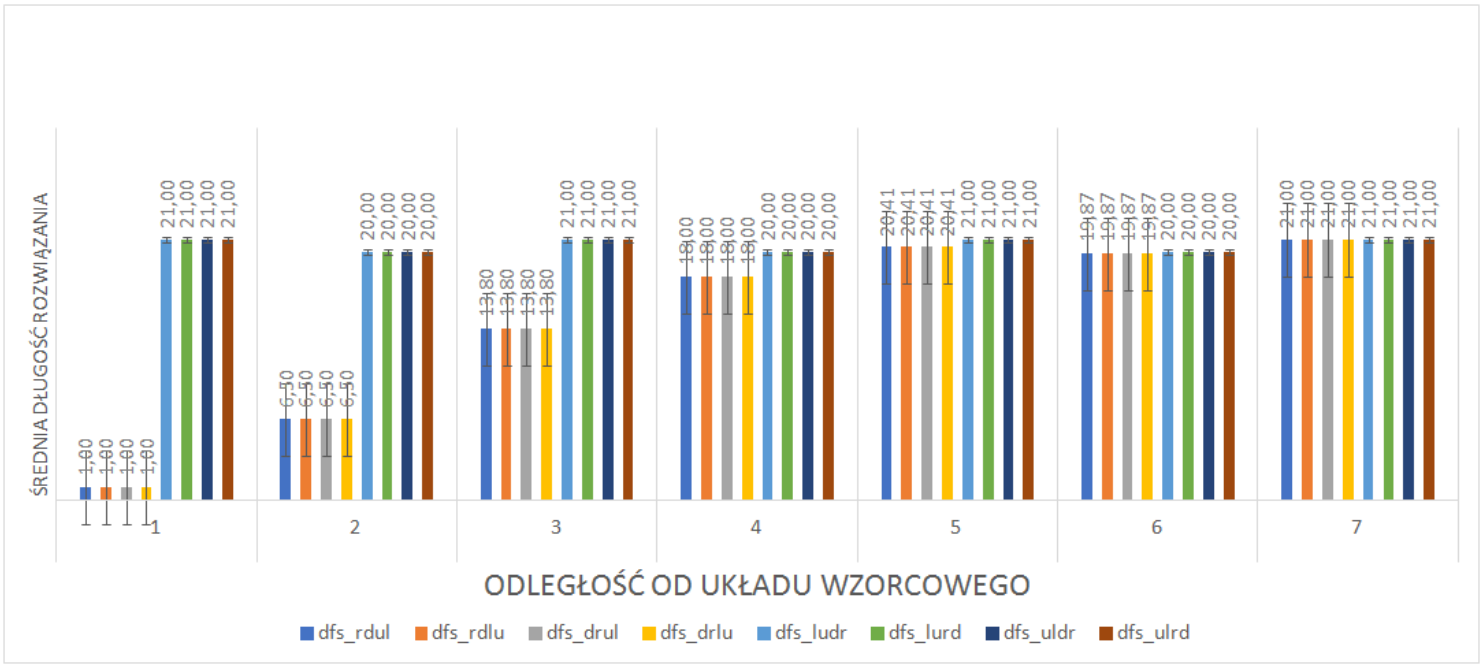
5.1. Średnia długość rozwiązania



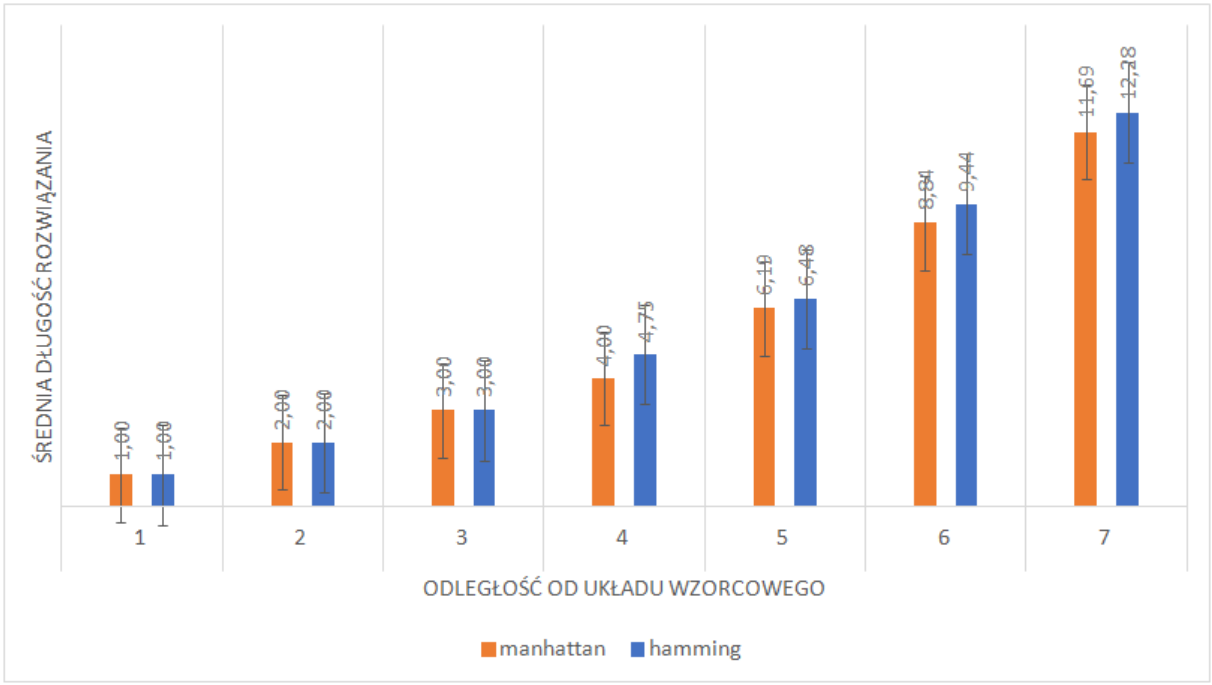
Rysunek 2. Średnia długość rozwiązania



Rysunek 3. BFS - średnia długość rozwiązania dla poszczególnych porządków przeszukiwania sąsiedztwa

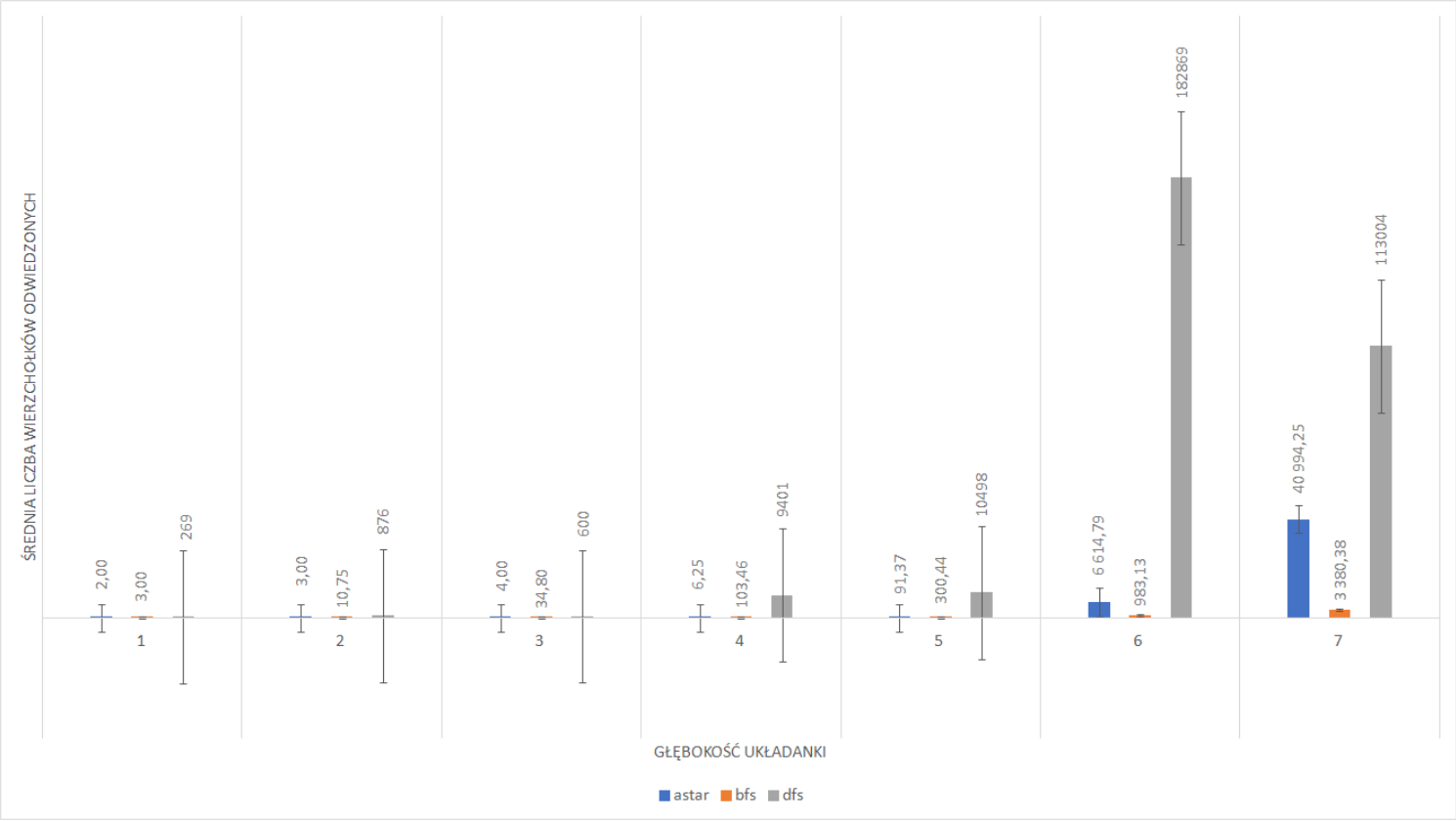


Rysunek 4. DFS - średnia długość rozwiązania dla poszczególnych porządków przeszukiwania sąsiedztwa

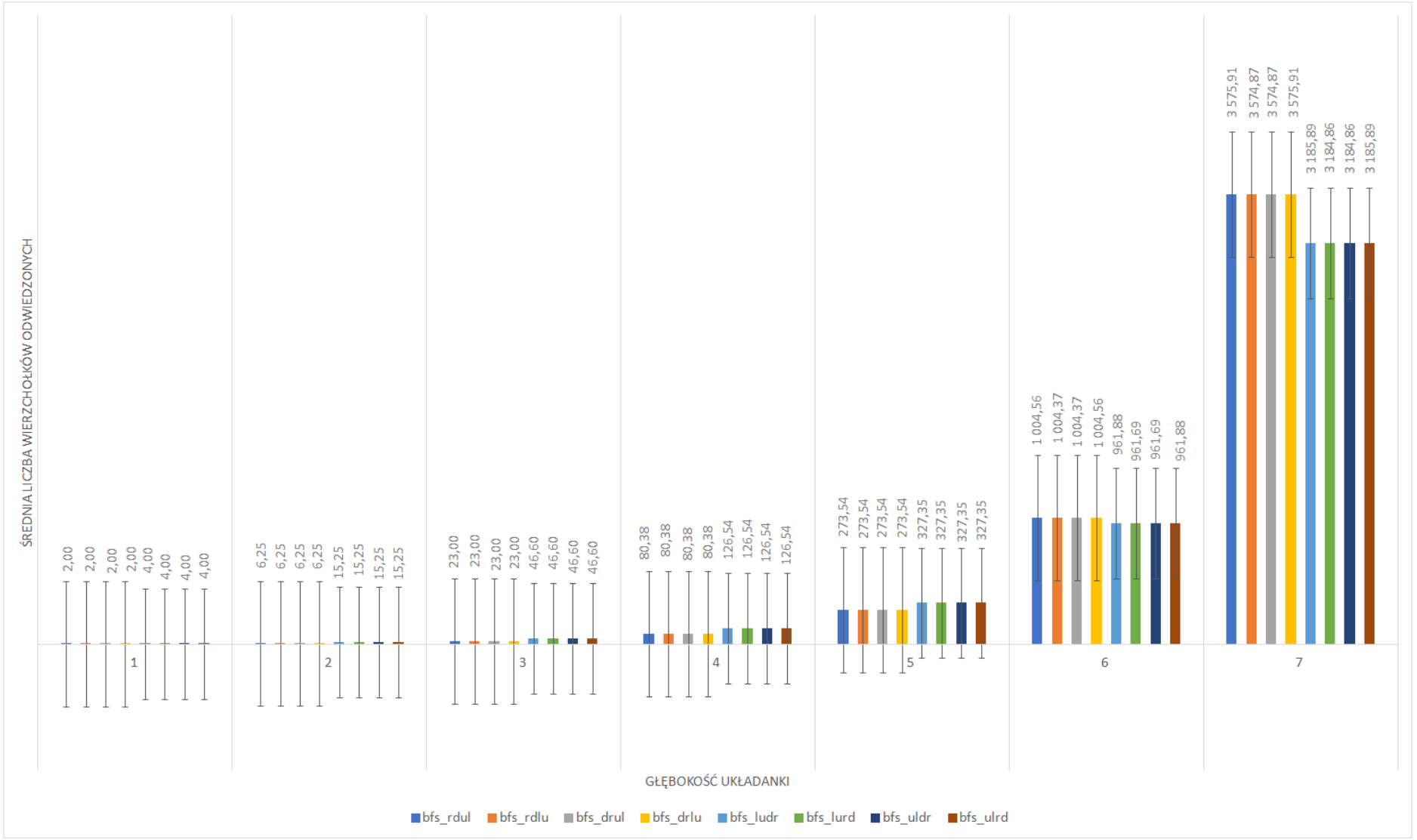


Rysunek 5. A* - średnia długość rozwiązania dla różnych heurystyk

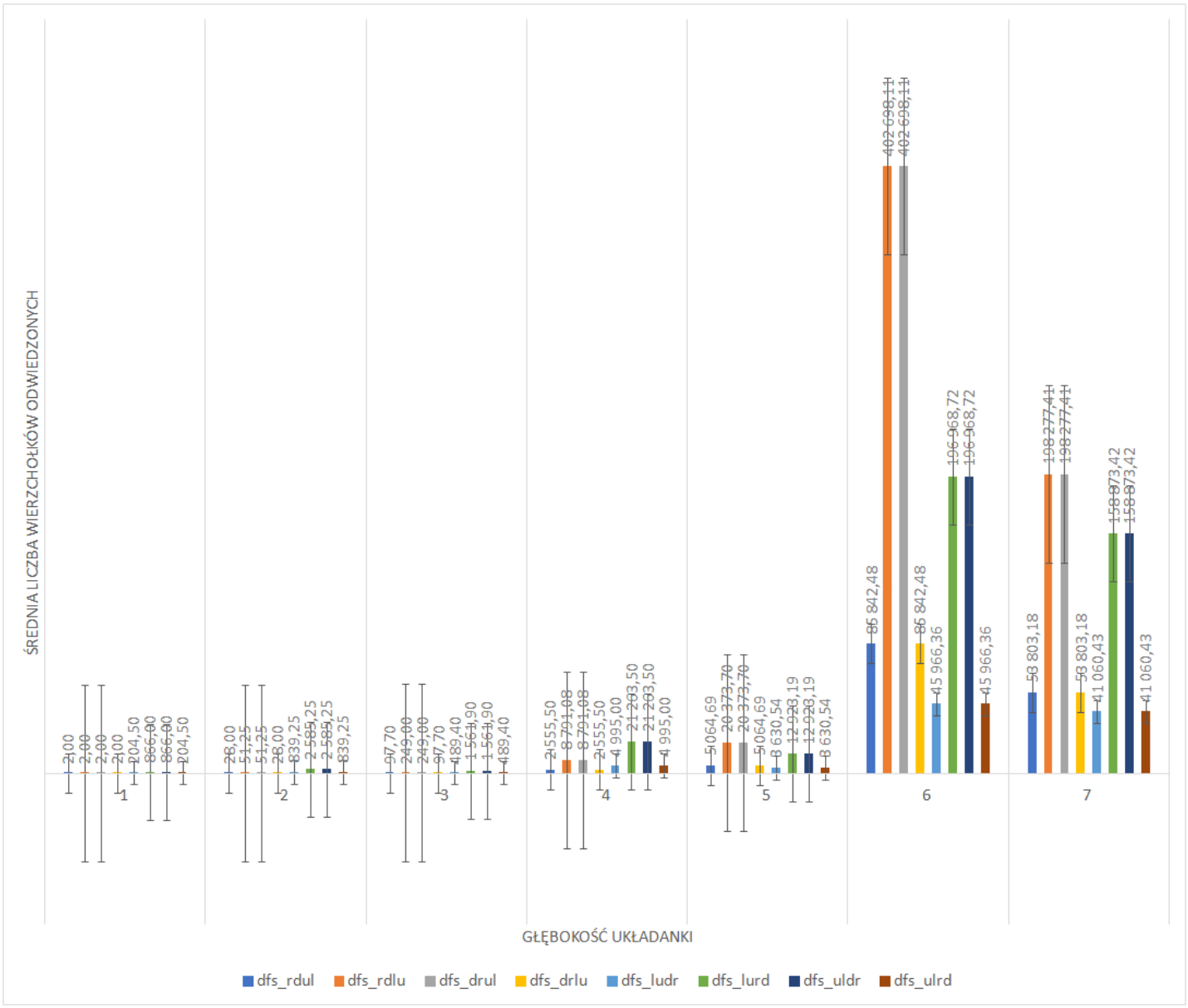
5.2. Średnia liczba wierzchołków odwiedzonych



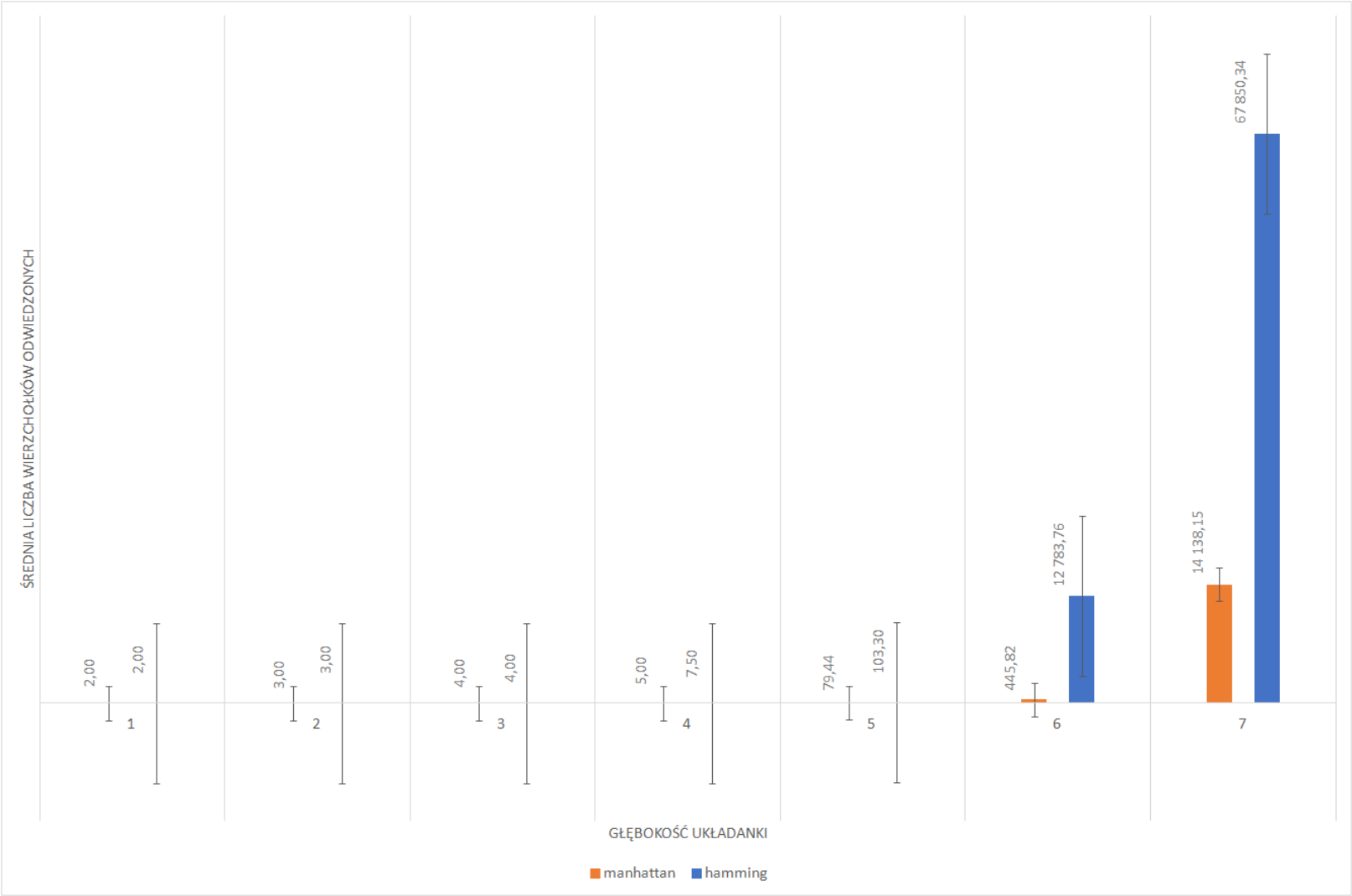
Rysunek 6. Średnia liczba wierzchołków odwiedzonych



Rysunek 7. BFS – średnia liczba wierzchołków odwiedzonych dla poszczególnych porządków przeszukiwania sąsiedztwa

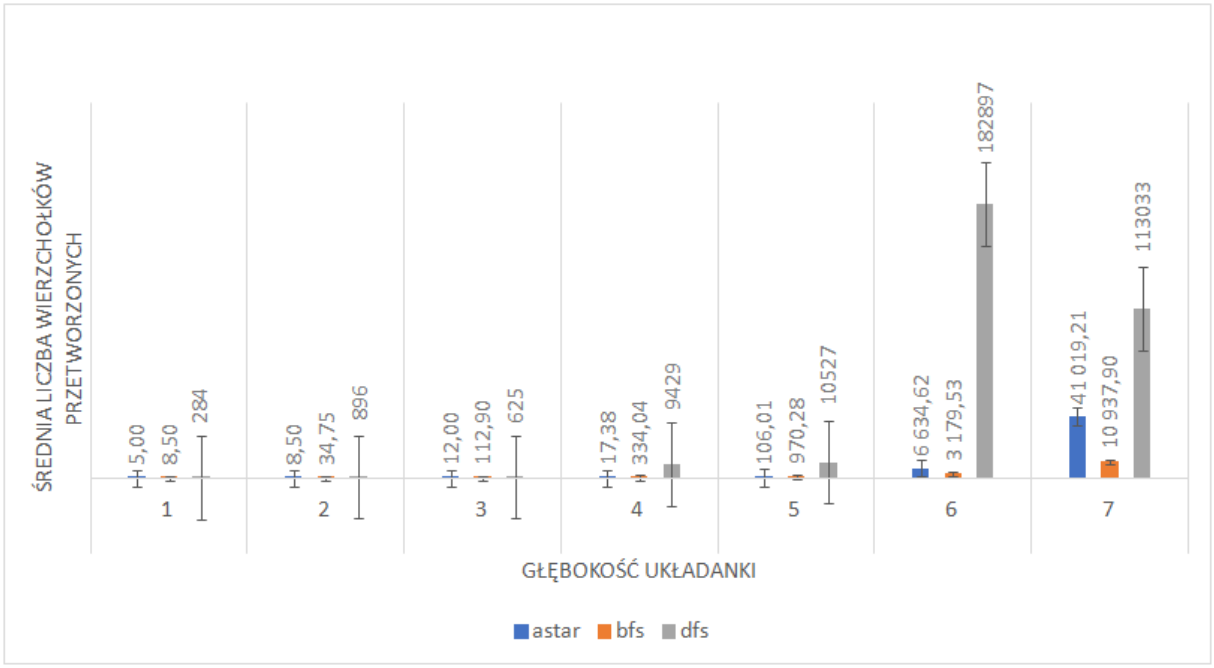


Rysunek 8. DFS – średnia liczba wierzchołków odwiedzonych dla poszczególnych porządków przeszukiwania sąsiedztwa

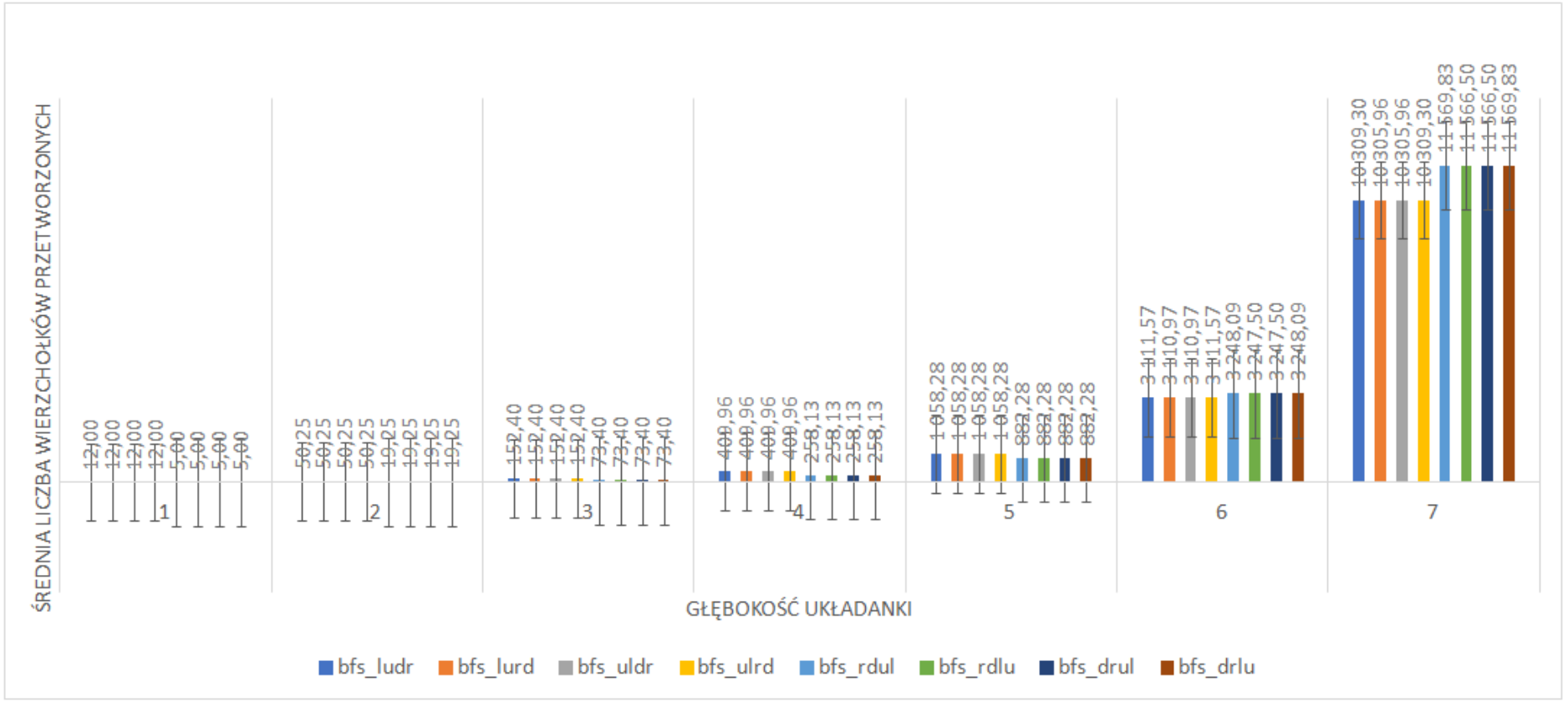


Rysunek 9. A* - średnia liczba wierzchołków odwiedzonych dla poszczególnych heurystyk

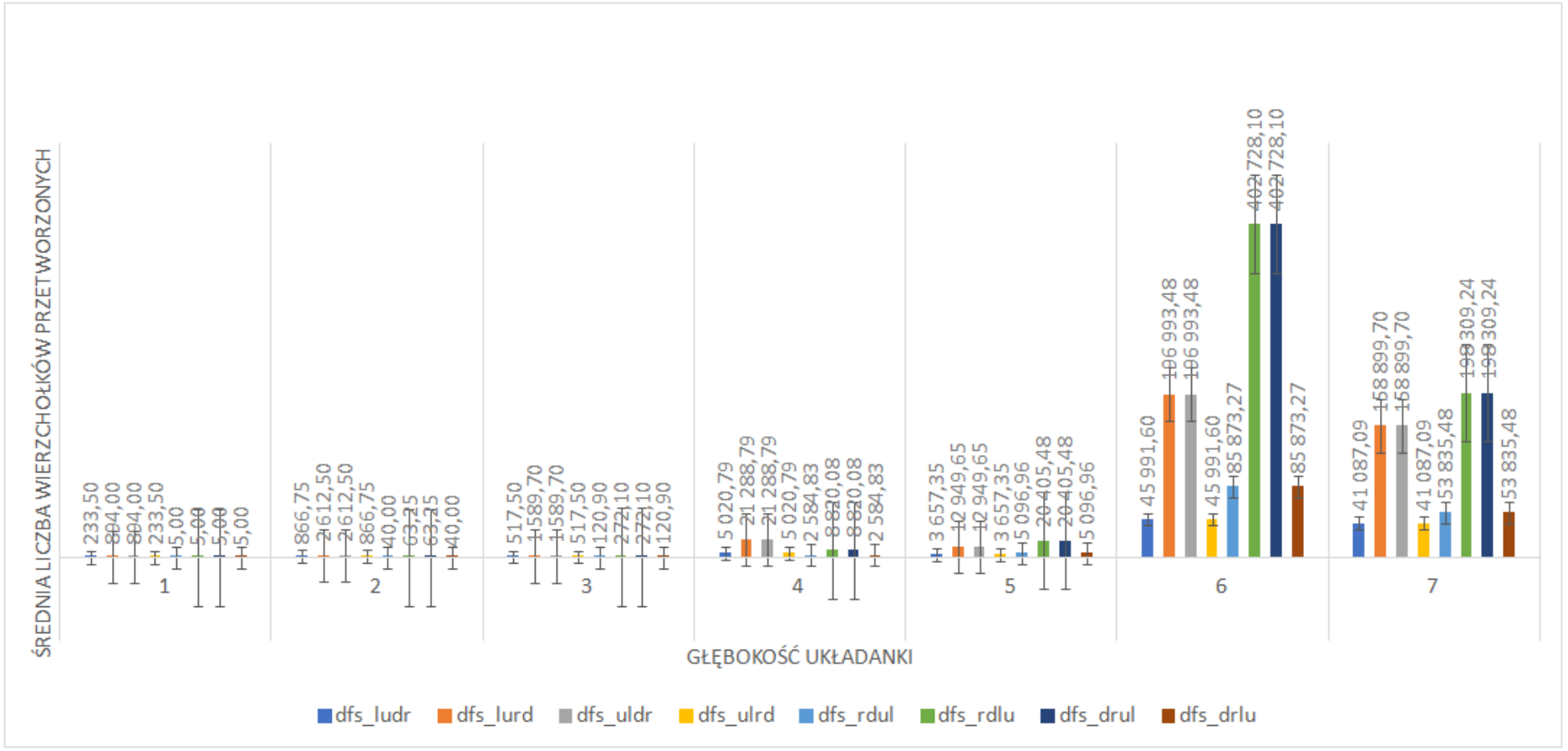
5.3. Średnia liczba wierzchołków przetworzonych



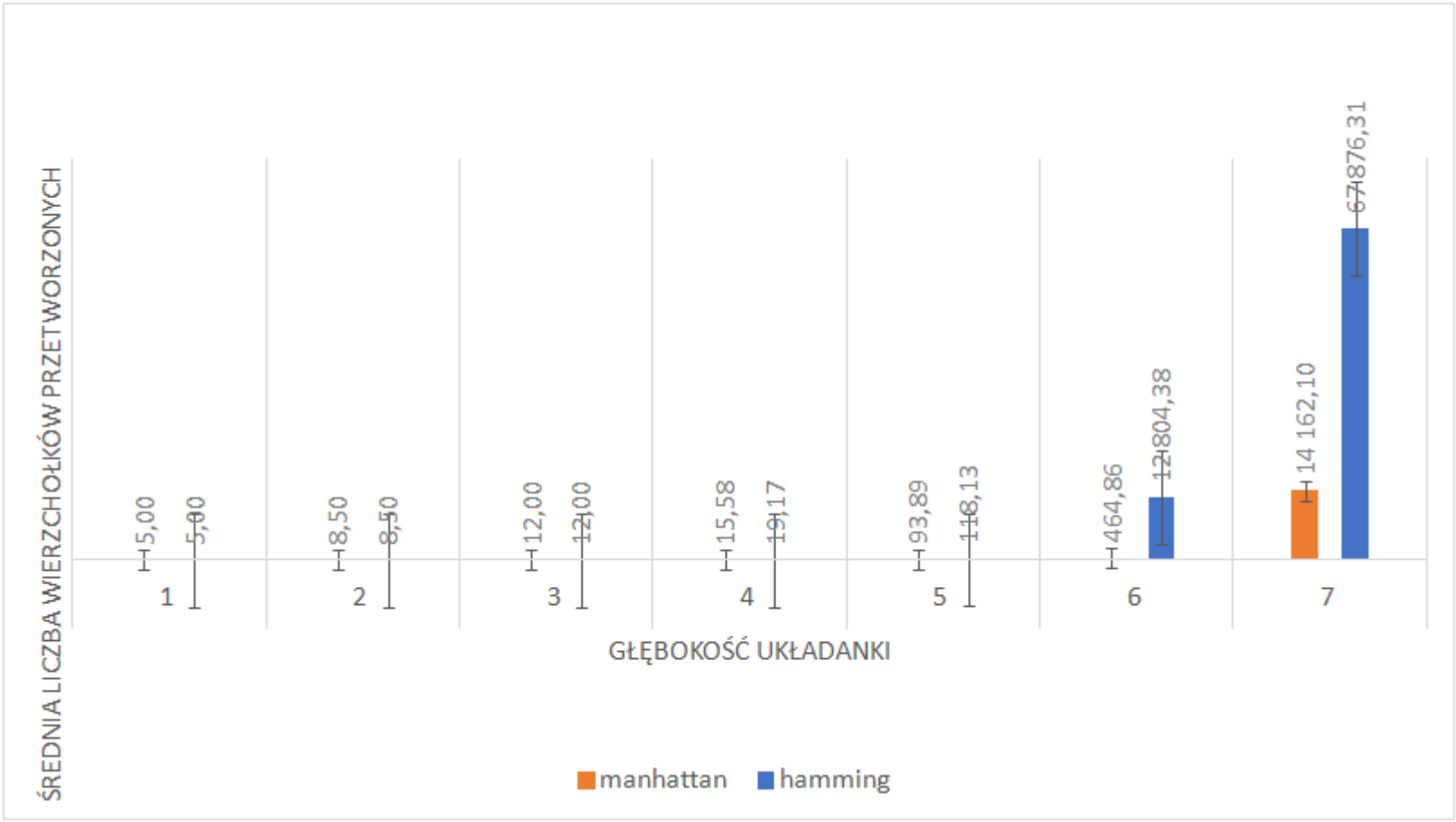
Rysunek 10. Średnia liczba wierzchołków przetworzonych



Rysunek 11. BFS – średnia liczba wierzchołków przetworzonych dla poszczególnych porządków przeszukiwania sąsiedztwa

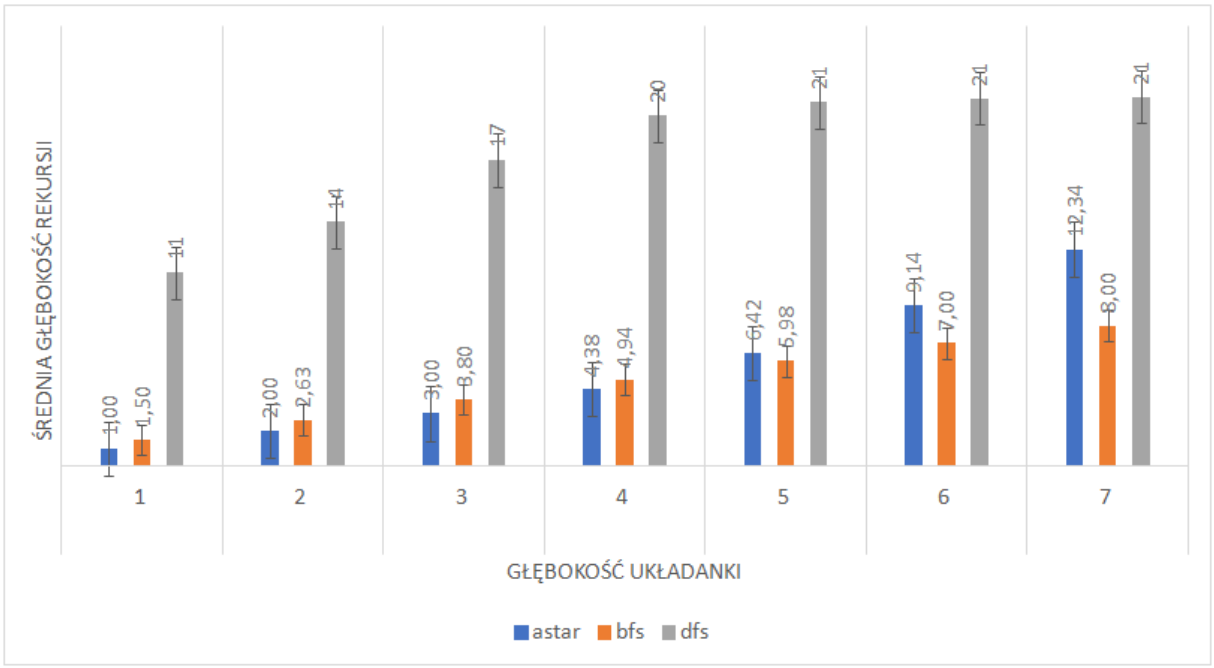


Rysunek 12. DFS – średnia liczba wierzchołków przetworzonych dla poszczególnych porządków przeszukiwania sąsiedztwa

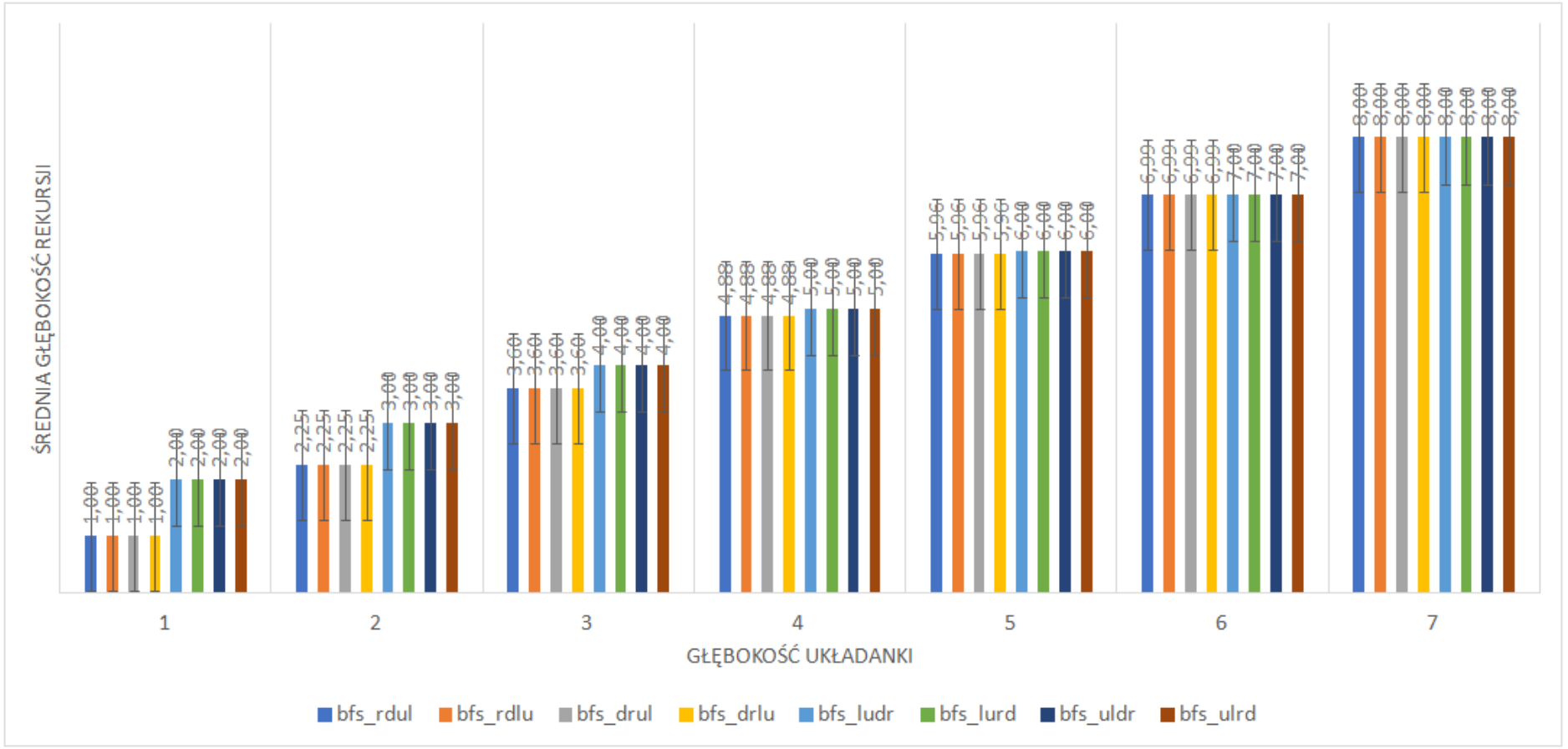


Rysunek 13. A* -średnia liczba wierzchołków przetworzonych dla poszczególnych heurystyk

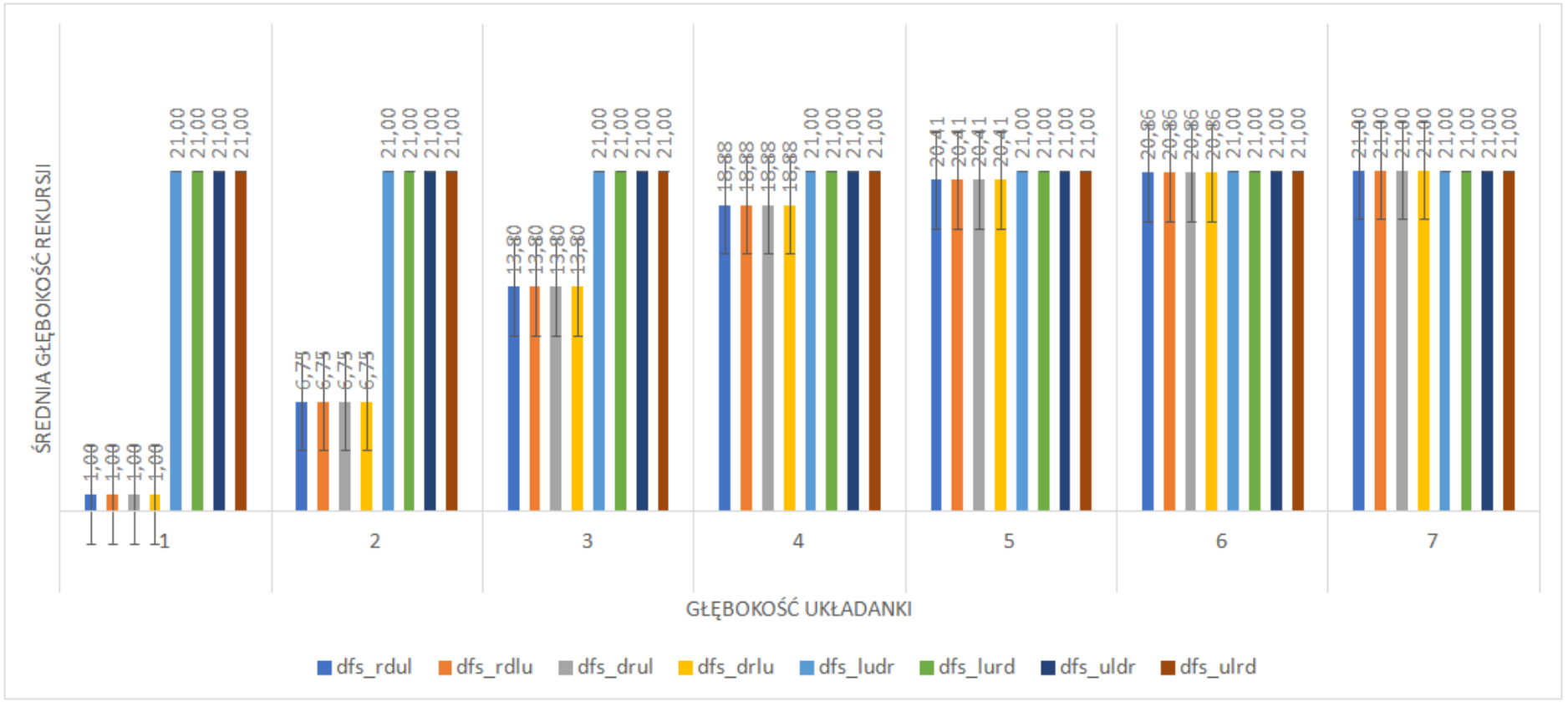
5.4. Średnia maksymalna głębokość rekursji



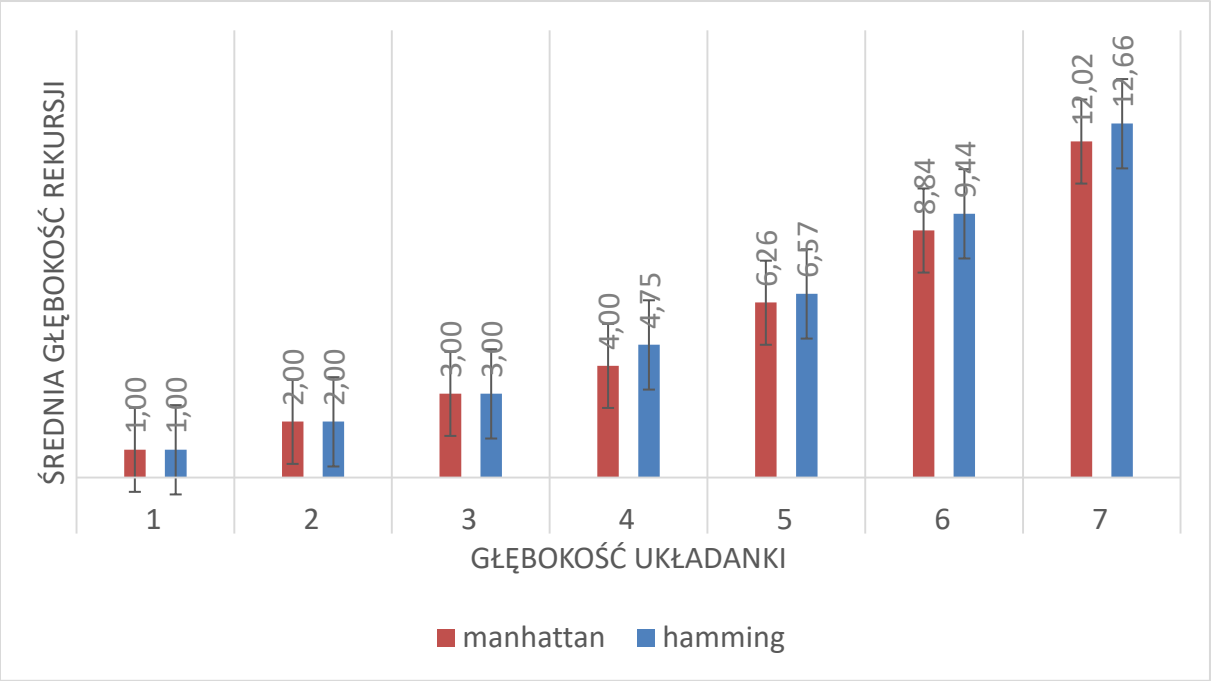
Rysunek 14. Średnia maksymalna głębokość rekursji



Rysunek 15. BFS – średnia maksymalna głębokość rekursji dla poszczególnych porządków przeszukiwania sąsiedztwa

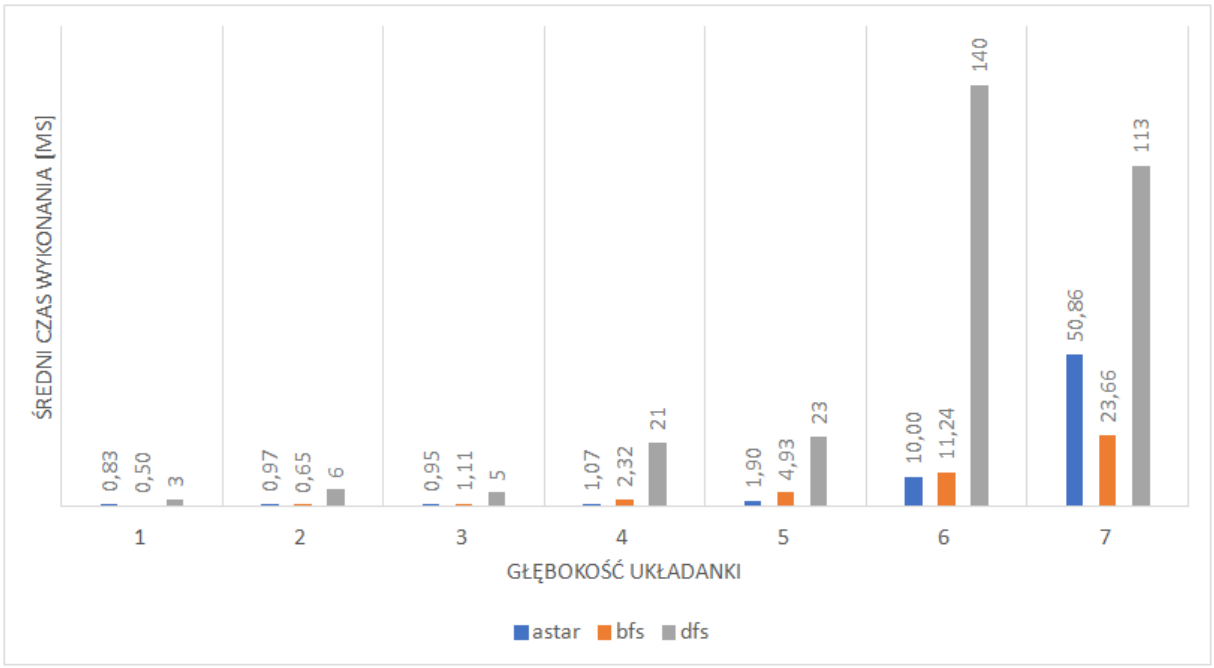


Rysunek 16. DFS – średnia maksymalna głębokość rekursji dla poszczególnych porządków przeszukiwania sąsiedztwa

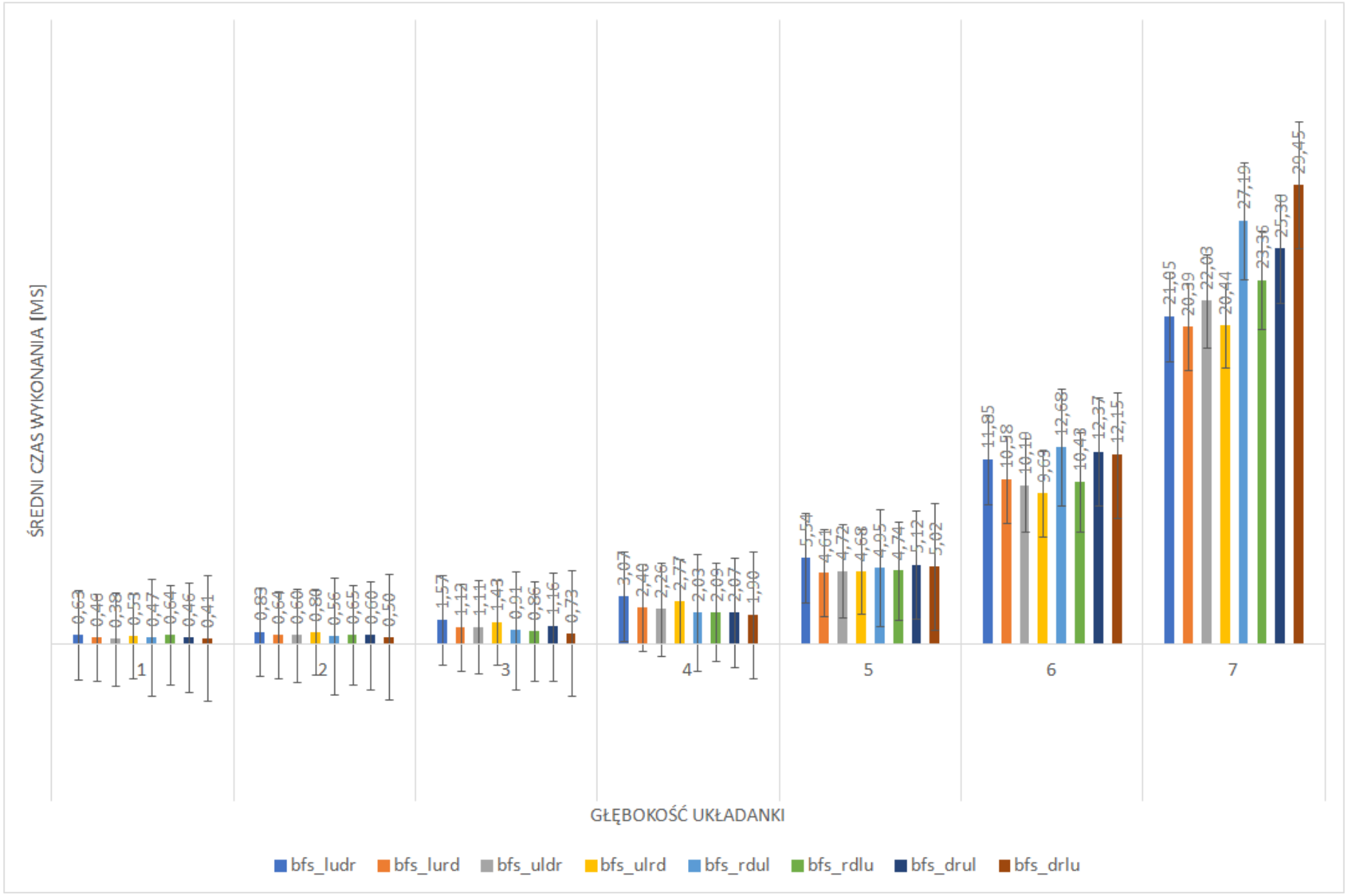


Rysunek 17. A* -średnia maksymalna głębokość rekursji dla poszczególnych heurystyk

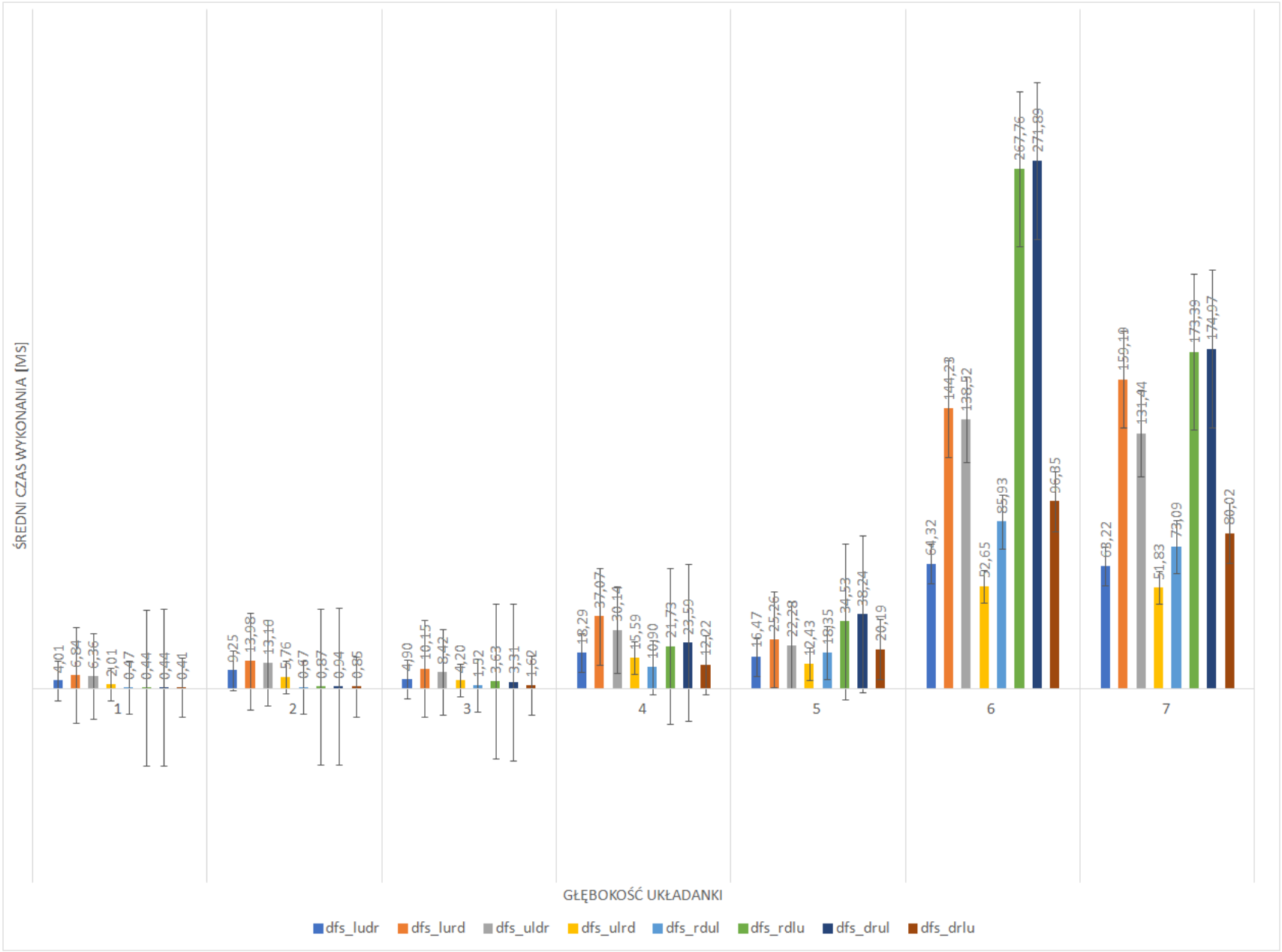
5.5. Średni czas trwania procesu obliczeniowego



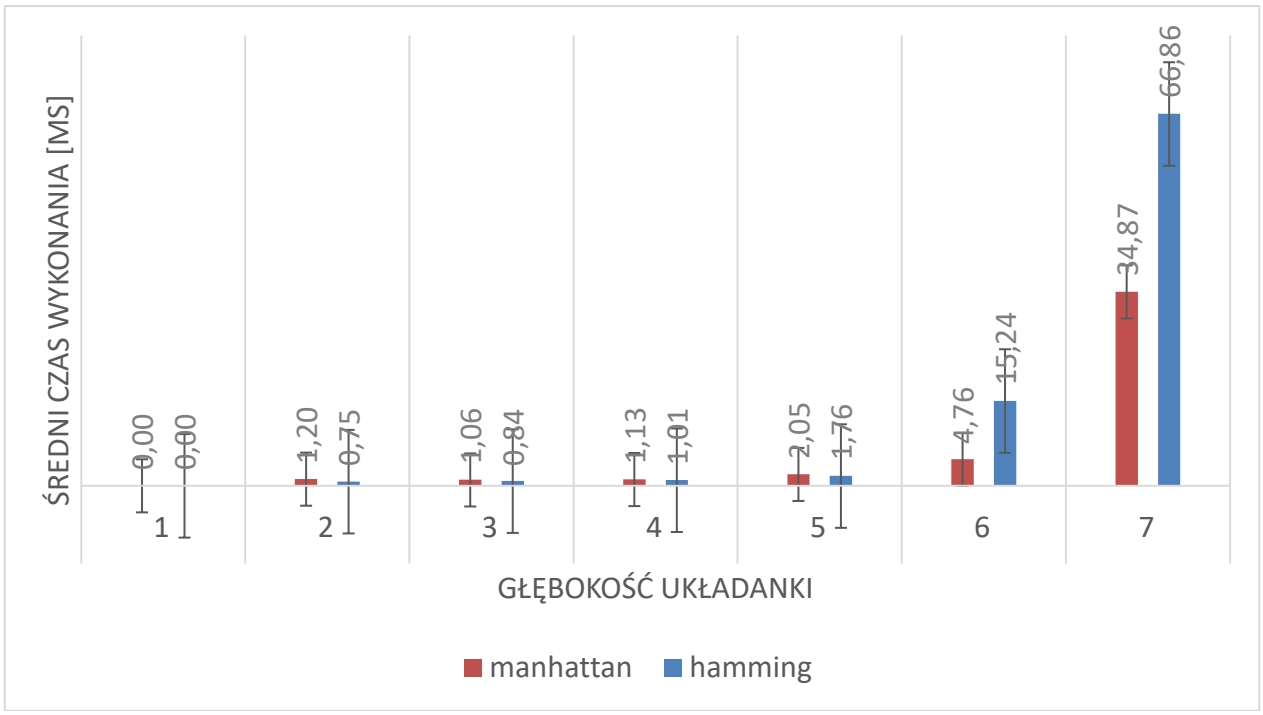
Rysunek 18. Średni czas trwania procesu obliczeniowego



Rysunek 19. BFS – średni czas trwania procesu obliczeniowego dla poszczególnych porządków przeszukiwania sąsiedztwa



Rysunek 20. DFS – średni czas trwania procesu obliczeniowego dla poszczególnych porządków przeszukiwania sąsiedztwa



Rysunek 21. A* - średni czas trwania procesu obliczeniowego dla poszczególnych heurystyk

6. Dyskusja

6.1. Średnia długość rozwiązania

Porównując całościowo strategie przeszukiwania, najlepiej z zadaniem rozwiązania układanki poradził sobie BFS. Średnia długość rozwiązania odpowiadała odległości układanki od układu wzorcowego. Z zadaniem rozwiązania układanki dobrze poradziła sobie również strategia A*. Przy odległościach 1 – 4 długość rozwiązania odpowiadała tym odległościom. Przy większych odległościach układanki dało się zaobserwować zwiększanie się rozbieżności pomiędzy odległością układanki a długością rozwiązania.

Porównując poszczególne porządki przeszukiwania sąsiedztwa, BFS – długość rozwiązania wynosiła tyle co odległość układanki od układu wzorcowego niezależnie od porządku. DFS – przy odległościach 1 -3 widać zdecydowaną przewagę porządków rd../dr.. w stosunku do porządków ul../lu.. . Im większa odległość, tym mniejsza różnica pomiędzy tymi porządkami.

A* - przy odległościach 1 -3 obie heurystyki sprawdzały się tak samo. Przy większych odległościach w układance Hamming okazał się być gorszy od Manhattana.

6.2. Średnia liczba wierzchołków odwiedzonych

BFS – można zauważyć pewną prawidłowość – przy coraz większej odległości w układance, liczba wierzchołków zwiększa się 3 – krotnie względem liczby wierzchołków mniejszej o jeden.

A* - przy odległościach 1 – 4 strategia ta radziła sobie najlepiej. Przy odległości 5 widać znaczący wzrost liczby wierzchołków odwiedzonych. Przy odległości 6 liczba wierzchołków względem odległości 5 zwiększyła się aż 60 - krotnie, przy odległości 7 – ponad 6 – krotnie.

DFS wypadł zdecydowanie najgorzej. Przy każdej z odległości liczba wierzchołków była znacząco większa od liczby w innych strategiach. Co ciekawe – przy odległości 7 liczba ta była o 70 tysięcy mniejsza niż liczba przy odległości 6 przy tej samej strategii.

BFS – liczba wierzchołków dla poszczególnych porządków przeszukiwania sąsiedztwa jest porównywalna.

DFS – dla odległości 1 – 3 najlepsze okazały się porządki przeszukiwań sąsiedztwa DR../RD.. . Przy większej odległości porządki RDLU, DRUL, LURD i ULRD okazały się najgorsze.

W odległościach 1 – 5 Hamming i Manhattan były porównywalne. Od odległości 6 nastąpił zdecydowany, gwałtowny wzrost liczby w wierzchołków w metryce Hamminga.

Najstabilniejsza okazała się metoda BFS. Wzrost liczby wierzchołków odwiedzonych był regularny, co wynika z natury tej metody.

6.3. Średnia liczba wierzchołków przetworzonych

Odległości 1 – 4 – A* przetwarzał najmniej wierzchołków. Gwałtowny wzrost liczby tych wierzchołków nastąpił dla odległości 6, 7.

BFS – regularny wzrost, każda kolejna odległość to 3 – krotność liczby wierzchołków w poprzedniej odległości. DFS – największa liczba wierzchołków przetworzonych.

BFS – liczby wierzchołków dla poszczególnych porządków przeszukiwania sąsiedztwa są porównywalne.

DFS – odległości 1 – 4 – rezultaty dla poszczególnych porządków przeszukiwania sąsiedztwa porównywalne. Dla odległości 6, 7 najgorzej wypadły porządki LURD, ULRD, RDLU, DRUL.

A* - odległości 1 – 5 – porównywalna liczba wierzchołków przetworzonych. Odległości 6, 7 – metryka Hamminga generuje nieporównywalnie większe liczby wierzchołków przetworzonych.

6.4. Średnia maksymalna głębokość rekursji

DFS osiąga największą głębokość rekursji. Wynika to z natury tego algorytmu. Przy odległościach 5, 6, 7 osiągnięta zostaje największa głębokość ustawiona w programie.

A*, BFS – głębokość rekursji porównywalna dla tych metod, zdecydowanie mniejsza niż dla DFS.

BFS – porównywalna głębokość rekursji dla poszczególnych porządków przeszukiwania sąsiedztwa.

DFS – dla odległości 1 – 3 głębokość rekursji znacznie mniejsza dla porządków RDUL, RDLU, DRUL, DRLU. Dla pozostałych odległości głębokość rekursji porównywalna dla poszczególnych porządków przeszukiwania sąsiedztwa.

A* - metryki Manhattan i Hamming osiągają porównywalne głębokości rekursji.

6.5. Średni czas znalezienia rozwiązania

Dla odległości w układance 1 – 3 A* oraz BFS znajdują rozwiązanie w porównywalnym czasie. Dla odległości 4 i 5 BFS znajduje rozwiązanie w czasie dwukrotnie dłuższym niż A*. Co ciekawe, tendencja odwraca się przy odległości o wartości 5. Dla tej odległości czas znalezienia rozwiązania dla BFS i A* jest porównywalny, zaś dla odległości 7 czas znalezienia rozwiązania w BFS jest dwukrotnie krótszy niż dla A*.

DFS znajduje rozwiązanie najdłużej ze wszystkich metod.

BFS – czas znalezienia rozwiązania dla poszczególnych porządków przeszukiwania sąsiedztwa jest porównywalny.

DFS - podczas gdy dla odległości 1 – 5 czasy znalezienia rozwiązania są porównywalne dla poszczególnych porządków przeszukiwania sąsiedztwa, przy odległościach 6 i 7 czas znalezienia rozwiązania jest najdłuższy dla RDLU, DRUL.

Metryka Hamminga, Manhattana – dla odległości 1 – 5 czasy porównywalne, dla odległości 6, 7 czas znalezienia rozwiązania dla Hamminga 2, 3 – krotnie dłuższy.

7. Wnioski

- Algorytmy BFS i A* zawsze znajdywały najkrótszą ścieżkę.
- Maksymalna głębokość rekurencji w przypadku algorytmów BFS i A* nigdy nie przekraczała głębokości układu początkowego.
- Algorytm DFS często nie znajdywałby rozwiązania, gdyby nie ograniczenie maksymalnej głębokości rekurencji (dochodziłoby do wyczerpania pamięci).
- Liczba wierzchołków przetworzonych jest porównywalna do liczby wierzchołków odwiedzonych dla strategii DFS. Liczba wierzchołków przetworzonych 3 – krotnie większa niż wierzchołków odwiedzonych - dla BFS.
- Algorytm BFS zachowywał się najbardziej przewidywalnie. Omówione wyżej wyniki dla poszczególnych kryteriów zmieniały się regularnie.
- Dla większych odległości metryka Manhattana sprawdza się lepiej niż metryka Hamminga.

8. Literatura

— https://en.wikipedia.org/wiki/Depth-first_search
— https://en.wikipedia.org/wiki/Breadth-first_search
— https://en.wikipedia.org/wiki/A*_search_algorithm
— https://en.wikipedia.org/wiki/Hamming_distance
— https://en.wikipedia.org/wiki/Taxicab_geometry