Tymoteusz Konkol 160845
Tomasz Kontek 160553
Sztuczna Inteligencja, projekt - Porównanie algorytmów przeszukiwania grafów.

Sprawozdanie.

I. Tematem projektu było porównanie algorytmów przeszukiwania grafów nieskierowanych ważonych.

II. Jako kryterium porównania wybraliśmy czas odnajdywania ścieżki od wybranego do zadanego wierzchołka, a także długość tej ścieżki.
Do badań utworzyliśmy algorytm tworzący graf na bazie macierzy *N*x*N*, który losował położenie $\frac{N \cdot N}{3}$ węzłów na tej mapie, po czym losował połączenia między tymi węzłami.

$\frac{N \cdot N}{3} \cdot 4$ dla grafów gęstych
$\frac{N \cdot N}{3}$ dla grafów rzadkich

III. Algorytmy przeszukiwania grafu wykorzystane w projekcie:

1. Algorytm Dijkstry dla odległości między wierzchołkami liczonymi z twierdzenia Pitagorasa, zależnie od położenia tych wierzchołków w macierzy.

   Sposób działania algorytmu Dijkstry:

   Niech *x* - aktualnie badany wierzchołek, *d(v)* - odległość wierzchołka *v* od startu, *d(x,v)* - odległość między *x*, a jego sąsiadem *v* .

   1. Wierzchołek startowy oznacz jako “odwiedzony” i wyznacz mu odległość od startu równą 0, pozostałym ustaw na ∞.
   2. Utwórz kolejkę priorytetową wierzchołków nie odwiedzonych. Priorytetem kolejki jest aktualnie wyznaczona odległość od wierzchołka startowego. Na początek kolejki wstaw wierzchołek startowy.
   3. Pobierz z kolejki wierzchołek o najniższym priorytecie.
   4. Dla każdego sąsiada *x* sprawdź czy nie został on już odwiedziony oraz czy *d(v) > d(x) + d(x,v)*, jeśli oba warunki są prawdziwe to podstaw *d(v) = d(x) + d(x,v)* i zaktualizuj pozycję *v* w kolejce.
   5. Powtarzaj punkty 3 i 4 dopóki nie osiągniesz ostatniego wierzchołka, lub kolejka nie będzie pusta.
   6. Zwróć odległość od startu ostatniego wierzchołka, lub informację że nie udało się go osiągnąć.

2. Algorytm A* dla odległości między wierzchołkami liczonymi z twierdzenia Pitagorasa, zależnie od położenia tych wierzchołków w macierzy.
   Jest to algorytm heurystyczny znajdowania najkrótszej ścieżki w grafie ważonym z dowolnego wierzchołka do wierzchołka spełniającego określony warunek zwany testem celu. Algorytm jest zupełny i optymalny, w tym sensie, że znajduje ścieżkę, jeśli tylko taka istnieje. Stosowany głównie w dziedzinie sztucznej inteligencji do rozwiązywania problemów i w grach komputerowych do imitowania inteligentnego zachowania.
   Zasada działania - Algorytm A* należy traktować jako uogólnienie algorytmu Dijkstry. Podczas wyznaczania kolejności odwiedzanych wierzchołków stosuje się w nim funkcje heurystyczne do określenia przypuszczalnej odległości do celu. Dlatego też algorytm A* przeszukuje najczęściej mniej węzłów grafu niż algorytm Dijkstry, co czyni go istotnie szybszym i zarazem bardziej efektywnym.

Najczęściej stosowanymi funkcjami heurystycznymi są:

- funkcje typu Manhattan (odległość dwóch węzłów to suma ich odległości w pionie i w poziomie (szybkie i mało dokładne)
- funkcje obliczające bezpośrednią odległość pomiędzy dwoma punktami Algorytm oceniając przypuszczalne drogi do celu zawsze wybiera węzeł o najmniejszej funkcji heurystycznej.

3. Algorytm Dijkstry dla odległości między wierzchołkami liczonymi z twierdzenia Pitagorasa, zależnie od położenia tych wierzchołków w macierzy, oraz dodaną do tej wartości wielkość losowa z przedziału $< 0, N >$.
   Sposób wyznaczania najkrótszej ścieżki jest taki sam jak w algorytmie nr 1.

4. Algorytm A* dla odległości między wierzchołkami liczonymi z twierdzenia Pitagorasa, zależnie od położenia tych wierzchołków w macierzy, oraz dodaną do tej wartości wielkość losowa z przedziału $< 0, N >$.
   Sposób wyznaczania najkrótszej ścieżki jest taki sam jak w algorytmie nr 2.

5. Algorytm Dijkstry dla odległości między wierzchołkami liczonymi z sumy wartości bezwzględnych różnic położenia obu wierzchołków odpowiednio na osi X i Y.
   Sposób wyznaczania najkrótszej ścieżki jest taki sam jak w algorytmie nr 1.

6. Algorytm A* dla odległości między wierzchołkami liczonymi z sumy wartości bezwzględnych różnic położenia obu wierzchołków odpowiednio na osi X i Y.
   Sposób wyznaczania najkrótszej ścieżki jest taki sam jak w algorytmie nr 2.

IV. Wyniki:

- Testy przeprowadzaliśmy dla *N* = 15, 50, 75, 100, 130.
  Co dawało ilość węzłów równą odpowiednio 75, 833, 1875, 3333, 5633.
  Oraz ilość połączeń równą odpowiednio 300, 3332, 7500, 13332, 22532
  dla grafów gęstych, a dla grafów rzadkich ilość połączeń równą ilości węzłów.

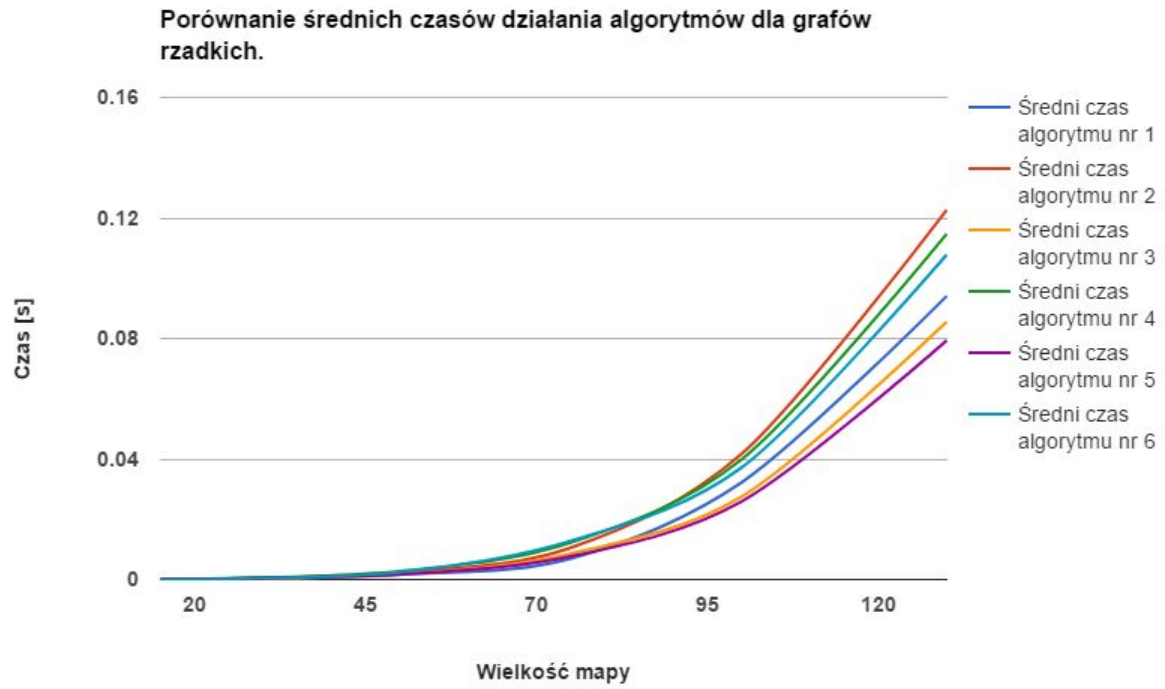
- Testy były wykonywane na losowych macierzach, więc dla każdego *N* utworzyliśmy 50 losowych grafów i uruchomiliśmy na nich nasze algorytmy. Po czym wyznaczyliśmy wartość średnią z uzyskanych 50 wyników dla każdego N.

- Otrzymane wyniki to czas działania dla każdego algorytmu oraz obliczona odległość zadanego wierzchołka od startowego.

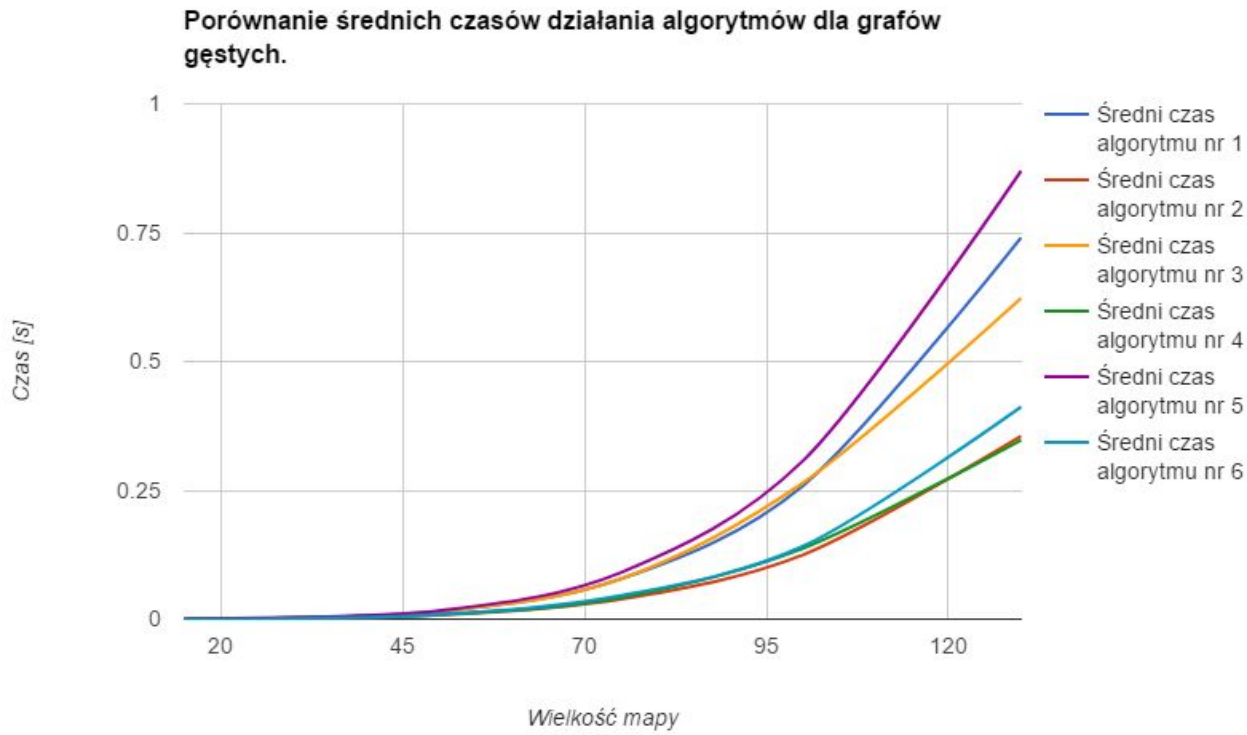
- Dla grafów rzadkich wszystkie metody pracowały mniej więcej taki sam przedział czasu, chociaż algorytmy Dijkstry wykazały się krótszym czasem działania niż algorytmy A*.

- Dla grafów gęstych algorytmy A* osiągały szybciej poszukiwany wierzchołek, wynika to z tego że dzięki funkcji heurystycznej odwiedzały mniejszą ilość wierzchołków

Tymoteusz Konkol 160845
Tomasz Kontek 160553
Sztuczna Inteligencja, projekt - Porównanie algorytmów przeszukiwania grafów.



Wykres nr 1 - Porównanie średnich czasów działania algorytmów dla grafów rzadkich.



Wykres nr 2 - Porównanie średnich czasów działania algorytmów dla grafów gęstych.

Tymoteusz Konkol 160845
Tomasz Kontek 160553
Sztuczna Inteligencja, projekt - Porównanie algorytmów przeszukiwania grafów.

- Porównując minimalne i maksymalne czasy działania algorytmu dla grafów rzadkich nie ma prawie różnicy.
  Jednak dla grafów gęstych widać że metody oparte na algorytmie Dijkstry mają większy czas działania, a także jest on zależny od sposobu wyznaczania odległości między wierzchołkami.
  Czas działania algorytmów A* jest bardziej niezależny od sposobu wyznaczania odległości między wierzchołkami.

	Odległości między węzłami obliczane z twierdzenia Pitagorasa	Odległości między węzłami obliczane z wartością losową	Odległości między węzłami obliczane funkcją Manhattan
Minimalny czas algorytmu nr 1	0	0	0
Minimalny czas algorytmu nr 2	0	0	0
Minimalny czas algorytmu nr 3	0.154	0.16	0.154
Minimalny czas algorytmu nr 4	0.163	0.154	0.153

Tabela nr 1 - Porównanie minimalnych i maksymalnych czasów działania algorytmów dla grafów rzadkich.

	Odległości między węzłami obliczane z twierdzenia Pitagorasa	Odległości między węzłami obliczane z wartością losową	Odległości między węzłami obliczane funkcją Manhattan
Minimalny czas algorytmu nr 1	0	0	0
Minimalny czas algorytmu nr 2	0	0	0
Minimalny czas algorytmu nr 3	0.826	0.933	1.237
Minimalny czas algorytmu nr 4	0.404	0.436	0.498

Tabela nr 2 - Porównanie minimalnych i maksymalnych czasów działania algorytmów dla grafów gęstych.

Tymoteusz Konkol 160845
Tomasz Kontek 160553
Sztuczna Inteligencja, projekt - Porównanie algorytmów przeszukiwania grafów.

- Wyznaczone odległości między pierwszym a ostatnim wierzchołkiem w większości przypadków różnią się. Wielkość różnicy jest zależna od gęstości grafu oraz metody jego przeszukiwania.
  Dla gęstych grafów różnica ta jest stosunkowo mniejsza od tej wyznaczonej dla grafów rzadkich.
  Jeżeli wartość jest różna to Algorytm Dijkstry wykazuje krótszą ścieżkę.

Wielkość mapy	Odległości między węzłami obliczane z twierdzenia Pitagorasa	Odległości między węzłami obliczane z wartością losową	Odległości między węzłami obliczane funkcją Manhattan
15	6.50	6.05	10.72
50	81.77	215.61	152.52
75	263.92	615.84	254.80
100	329.37	968.81	704.56
130	850.41	1,665.69	1,040.56

Tabela nr 3 - Porównanie średnich różnic w długości ścieżki wyznaczonej przez algorytmy dla grafów rzadkich.

Wielkość mapy	Odległości między węzłami obliczane z twierdzenia Pitagorasa	Odległości między węzłami obliczane z wartością losową	Odległości między węzłami obliczane funkcją Manhattan
15	1.14	2.91	0.52
50	31.99	50.91	39.00
75	66.80	87.07	94.96
100	104.89	217.94	132.28
130	207.49	290.17	194.24

Tabela nr 4 - Porównanie średnich różnic w długości ścieżki wyznaczonej przez algorytmy dla grafów gęstych.

Tymoteusz Konkol 160845
Tomasz Kontek 160553
Sztuczna Inteligencja, projekt - Porównanie algorytmów przeszukiwania grafów.

V. Wnioski:
1. Algorytm A* jest bardziej opłacalny dla poszukiwań w gęstych grafach, a także jeżeli wyznaczana trasa jest zależna od kilku czynników.
2. Dla grafów rzadkich bardziej opłacalny jest algorytm Dijkstry, może to wynikać z tego że algorytm A* musi wyznaczać poza wartością odległości od początku danego wierzchołka, także wartość funkcji heurystycznej.
3. Algorytm A* lepiej sobie radzi w gęstych grafach dzięki funkcji heurystycznej, która ogranicza ilość odwiedzanych wierzchołków, co za tym idzie, zmniejsza ilość iteracji potrzebną do znalezienia ścieżki między dwoma wierzchołkami, co skraca czas działania algorytmu.
4. Dzięki funkcji heurystycznej Algorytm A* może brać pod uwagę także inne wartości niż odległość od początku, np. wysokość nad poziomem morza danego punktu.
5. Różnice w długości wyznaczonej ścieżki mogą być spowodowane funkcją heurystyczną wykorzystywaną w algorytmie A*. Zmusza ona czasem do wybrania mniej opłacalnej ścieżki, ponieważ prowadzi ona szybciej do końca wg niej. Dobrym przykładem takiego zachowania jest góra na mapie, algorytm Dijkstry oblicza tylko ścieżkę bezpośrednią, nie licząc kosztów wspięcia się na szczyt, algorytm A* omija szczyt prowadząc łatwiejszą choć dłuższą ścieżkę.
6. Algorytm Dijkstry dobrze radzi sobie gdy odległość między wierzchołkami jest wartością z twierdzenia Pitagorasa, czyli dla odległości bezpośrednich. Gdy ta wartość jest inaczej wyznaczana czas działania funkcji znacząco wzrasta. Może to symulować ścieżki niebezpośrednie, takie jak np. obchodzenie góry, z którymi lepiej sobie radził algorytm A*.