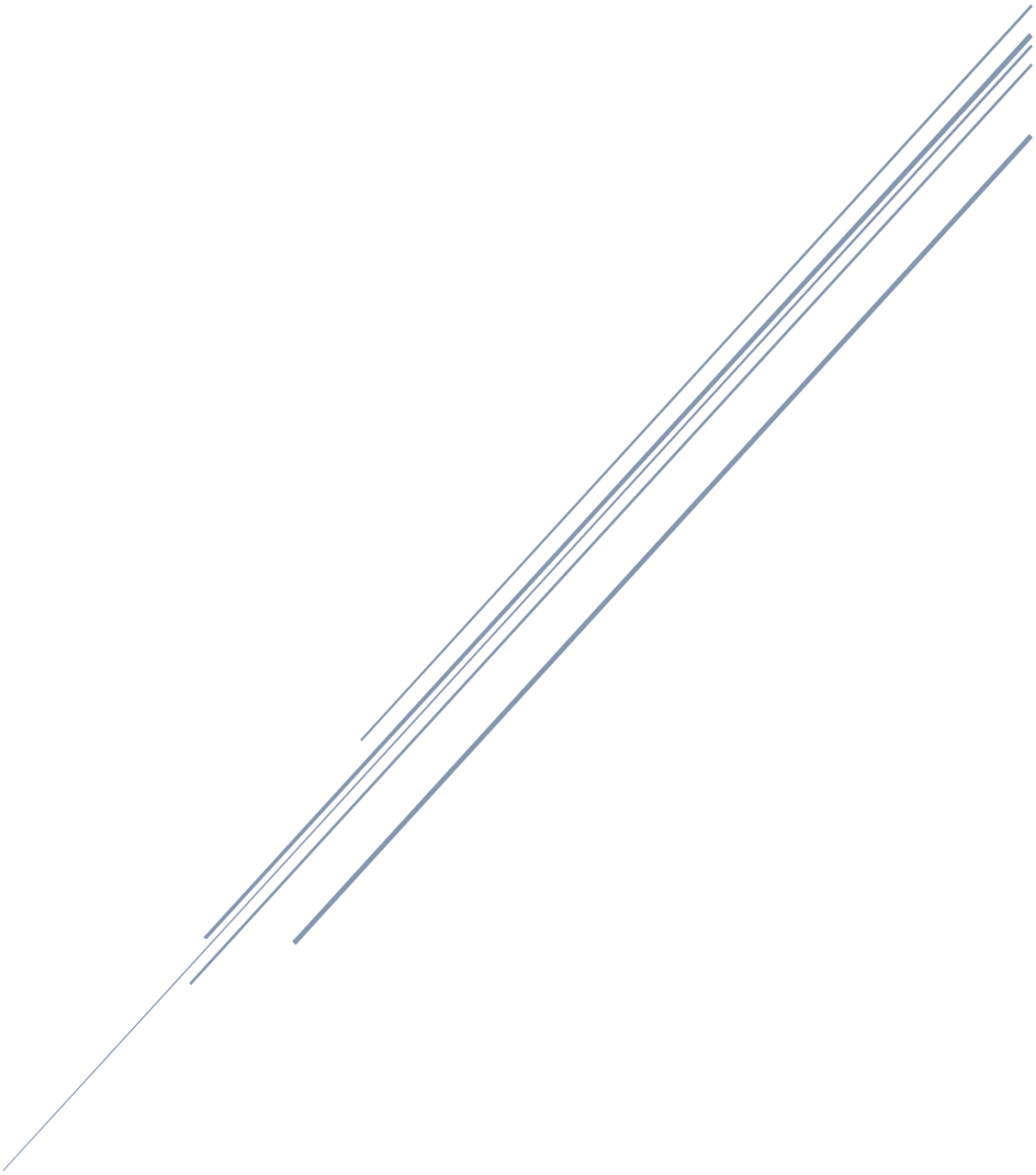
PROJEKT 3

Kacper Kaciłowicz 248951



Mgr inż. Magda Skoczeń

Projektowanie algorytmów i metody sztucznej inteligencji

1. Wprowadzenie

Celem projektu była zaprojektowanie grafów, czyli zbioru wierzchołków, które mogą być połączone krawędziami. W celu dokładnego opisu struktury danego grafu określa się jego właściwości, takie jak liczba wierzchołków, czy gęstość, czyli stosunek liczby krawędzi do największej możliwej liczby krawędzi , a w przypadku tego projektu przyjęte zostały również założenia, że graf jest:

- Spójny – dla każdej pary wierzchołków istnieje ścieżka, która je łączy
- Skierowany – każda krawędź ma wyznaczony kierunek, a jego gęstość określa się wzorem $D = \frac{|E|}{|V|(|V|-1)}$, gdzie |E| oznacza liczbę krawędzi, a |V| liczbę wierzchołków
- Ważony – każda krawędź ma swoją wagę

Istnieją różne reprezentacje matematyczne służące do wykonywania obliczeń i obsługi grafów, lecz w tym przypadku zaimplementowane oraz zbadane zostały lista oraz macierz sąsiedztwa. Ponadto celem projektu, było wykorzystanie grafów do rozwiązania problemu najkrótszej ścieżki, czyli znalezieniu w grafie ważonym połączenia między dwoma wierzchołkami o najmniejszej sumie wag krawędzi – najkrótszej drodze. Do rozwiązania tego problemu zostały zaimplementowane oraz poddane testom dwa algorytmy – Bellmana-Forda oraz Dijkstry. W celu realizacji badań zostały zmierzone czasy 100 grafów, a następnie wyniki zostały uśrednione. Rozpatrzone zostały przypadki grafów:

- O różnym rozmiarze (liczbie wierzchołków): 10, 25, 50, 75, 100, 150, 200, 250, 300
- O różnej gęstości: 25%, 50%, 75%, 100%
- O różnych reprezentacjach: macierz sąsiedztwa oraz lista sąsiedztwa
- Z wykorzystanie dwóch algorytmów najkrótszej ścieżki: Bellman-Ford oraz Dijkstra

2. Opis algorytmów najkrótszej ścieżki

2.1 Algorytm Dijkstry

Algorytm ten jest znacznie szybszy niż algorytm Bellmana-Forda, lecz jest mniej uniwersalny, gdyż nie obsługiwane są w nim ujemne wagi krawędzi. Pierwszym krokiem algorytmu jest ustawienie wartości etykiety długości ścieżki dla każdego wierzchołka na nieskończoność (czyli w programie wybrać liczbę bardzo dużą), natomiast etykieta długości ścieżki wierzchołka startowego powinna zostać ustawiona na 0. Następnym krokiem jest wyciągnięcie ze zbioru wierzchołka o najmniejszej wartości dystansu (etykiecie ścieżki), czyli przy pierwszym wykonaniu jest to wierzchołek startowy. W informatyce do przeprowadzenia sortowania dystansu wierzchołków zaleca się korzystanie z kolejki priorytetowej o budowie kopca, która jest w stanie znacznie zwiększyć efektywność algorytmu. Kolejnym krokiem jest porównanie ścieżki dla wszystkich sąsiadów danego wierzchołka oraz ewentualne przypisanie nowych wartości dystansu, gdy znaleziono krótszą ścieżkę, czyli gdy spełniony jest warunek, że *dystans[sąsiada]>dystans[wierzchołka] +waga krawędzi między nimi*. Gdy w kolejce priorytetowej nie znajduje się już żaden wierzchołek, następuje koniec algorytmu i znana jest najkrótsza ścieżka od wierzchołka startowego do wszystkich wierzchołków grafu.

Złożoność obliczeniowa zależy przede wszystkim od metody wyszukiwania wartości najmniejsze w zbiorze. W przypadku zastosowania przeszukiwania liniowego złożoność może sięgnąć $O(|V|^2)$, natomiast w przypadku zastosowania kolejki priorytetowej o budowie kopca, zmniejsza się ona do $O(|E| + |V|log|V|)$.

2.2 Algorytm Bellmana-Forda

Jest to algorytm bardziej uniwersalny, niż algorytm Dijkstry, lecz zdecydowanie mniej wydajny. Zezwala on na ujemne wagi, natomiast nie pozwala na istnienie ujemnego cyklu, czyli cyklu o łącznej ujemnej wadze ze źródła. Z tego powodu nie jest możliwe wykorzystanie tego algorytmu do grafów nieskierowanych. Podobnie jak w przypadku algorytmu Dijkstry pierwszym krokiem jest ustawienie etykiety dystansu wszystkich wierzchołków na nieskończoność, oprócz wierzchołka startowego, którego wartość ustawia się na 0. Następnie dla kolejnych |V|-1 wierzchołków bada się wartości dystansu każdego z sąsiadów danego wierzchołka, a gdy warunek *dystans[sąsiada] > dystans[wierzchołka] + waga krawędzi między nimi* jest spełniony dokonuje się zmiany etykiety dystansu w wierzchołku badanym. Teoretycznie po zakończeniu procedury znana już jest najkrótsza ścieżka z wierzchołka startowego do pozostałych wierzchołków, lecz konieczne jest wykonanie procedury sprawdzającej czy nie wystąpił ujemny cykl.

Złożoność obliczeniowa algorytmu wynosi $O(|V||E|)$.

3.Prezentacja wyników

Przedstawione wartości dotyczą uśrednionego czasu wykonania dla 1 grafu na bazie pomiaru 100 losowych grafów. Jednostką czasu w każdym przypadku jest sekunda.

3.1 Tabele

- Algorytm Bellmana-Forda zaimplementowany za pomocą macierzy sąsiedztwa

Liczba wierzchołków	Gęstość grafu			
	25%	50%	75%	100%
10	0,000108	0,000206	0,000275	0,000353
25	0,004060	0,007848	0,011466	0,012217
50	0,009492	0,018982	0,026581	0,034683
75	0,032140	0,063276	0,091728	0,119746
100	0,077035	0,150085	0,217580	0,287304
150	0,261103	0,510220	0,734846	0,955448
200	0,614397	1,198220	1,931710	2,479030
250	1,300400	2,624560	3,587530	4,738310
300	2,216140	4,189990	6,162900	7,919330

- Algorytm Bellmana-Forda zaimplementowany za pomocą macierzy sąsiedztwa

Liczba wierzchołków	Gęstość grafu			
	25%	50%	75%	100%
10	0,000099	0,000164	0,000241	0,000309
25	0,001291	0,002399	0,003927	0,005384
50	0,009558	0,015467	0,023078	0,031542
75	0,027187	0,053141	0,081518	0,104776
100	0,062926	0,126366	0,186612	0,249299
150	0,211584	0,423616	0,628923	0,954580
200	0,498688	1,000050	1,711500	2,270820
250	1,120600	2,200390	3,242520	4,397850
300	1,880390	3,658000	5,599980	7,333800

- Algorytm Dijkstry zaimplementowany za pomocą macierzy sąsiedztwa

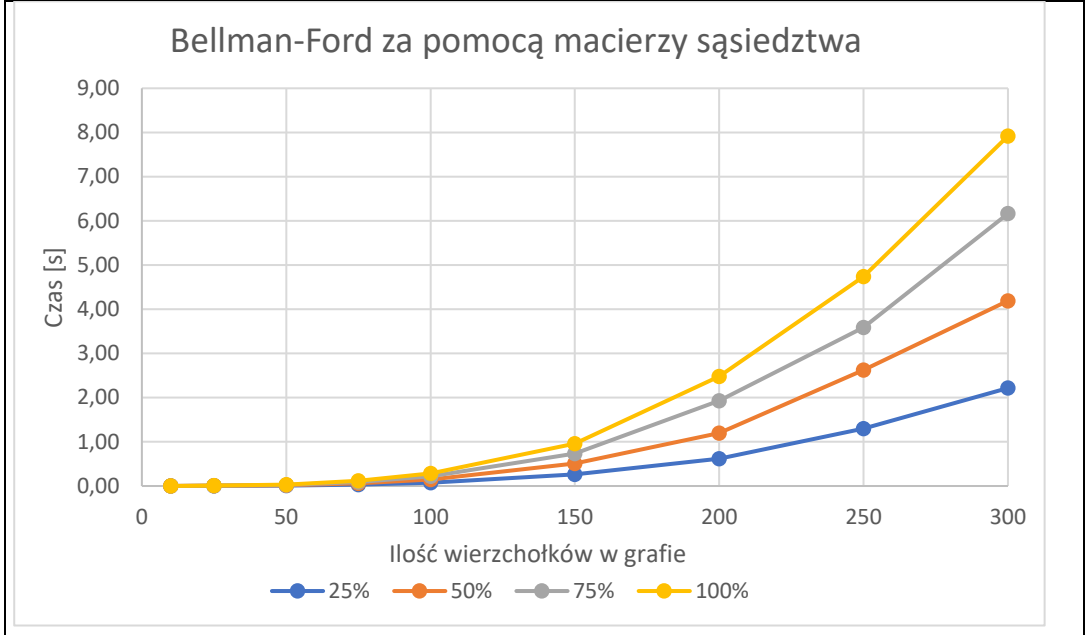
Liczba wierzchołków	Gęstość grafu			
	25%	50%	75%	100%
10	0,000100	0,000166	0,000237	0,000295
25	0,001129	0,002176	0,003543	0,004144
50	0,007553	0,012471	0,018448	0,024319
75	0,020571	0,040210	0,060317	0,081179
100	0,047972	0,095998	0,143218	0,191528
150	0,162178	0,328202	0,495593	0,726756
200	0,379992	0,759231	1,139850	1,668980
250	0,824257	1,677500	2,457600	3,284170
300	1,441920	2,786560	4,255900	5,588080

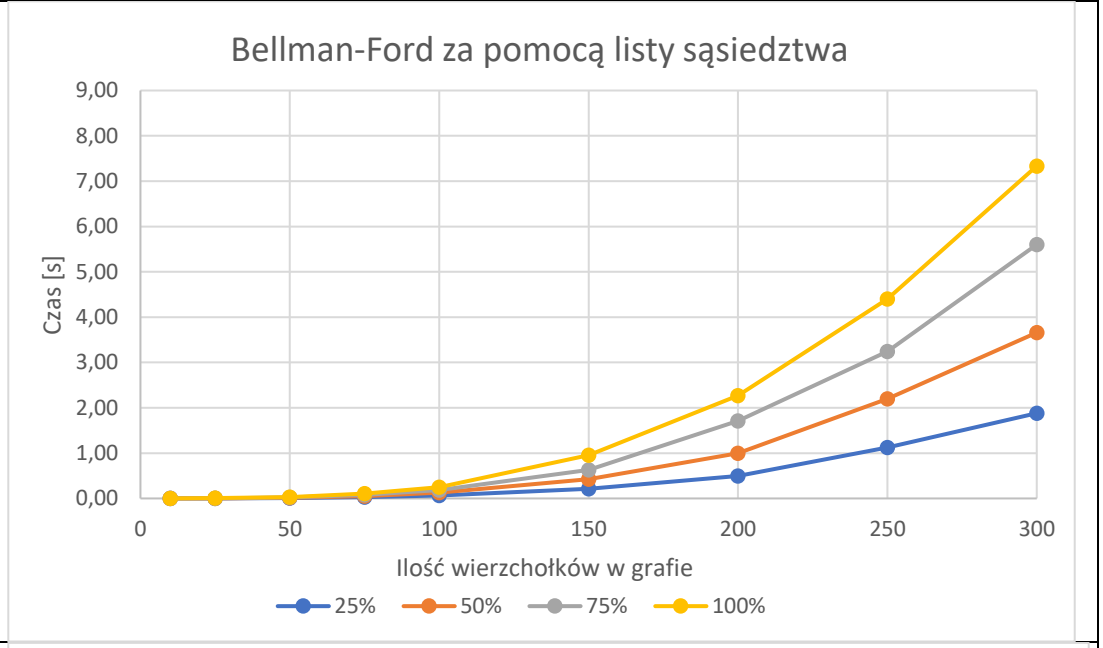
- Algorytm Dijkstry zaimplementowany za pomocą listy sąsiedztwa

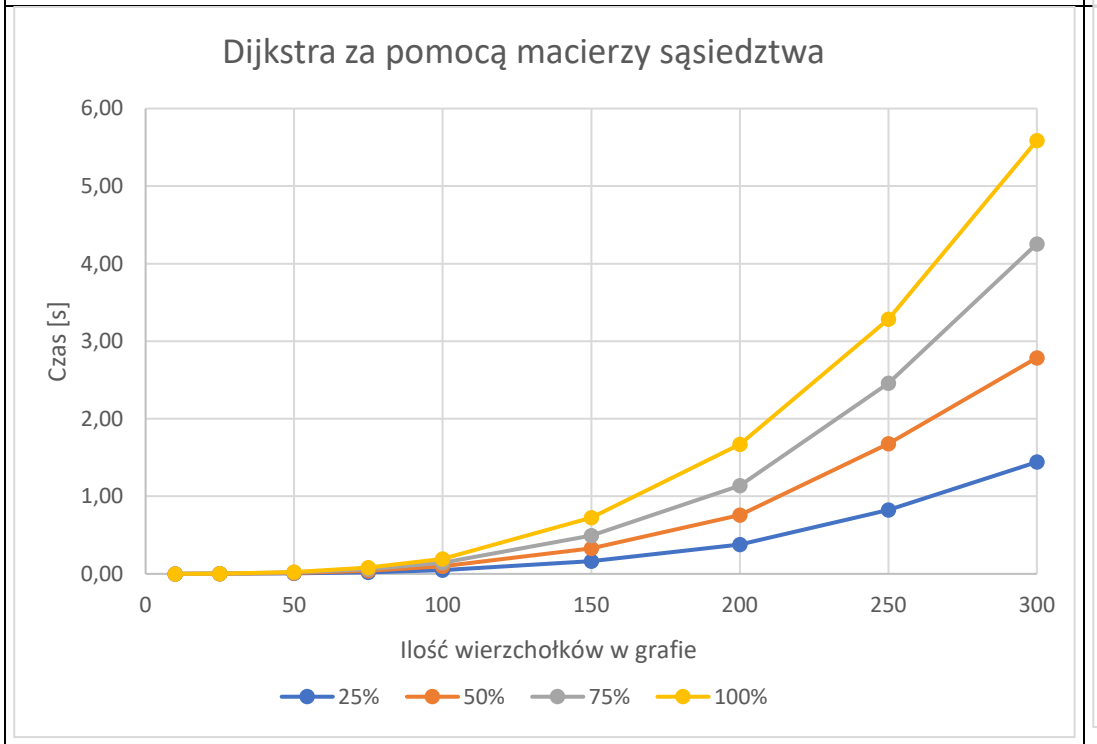
Liczba wierzchołków	Gęstość grafu			
	25%	50%	75%	100%
10	0,000094	0,000166	0,000221	0,000307
25	0,001152	0,002092	0,003094	0,003949
50	0,007493	0,012162	0,018860	0,023953
75	0,022090	0,056787	0,062506	0,082108
100	0,048563	0,096161	0,146401	0,191909
150	0,162892	0,326649	0,487674	0,647361
200	0,384712	0,765645	1,312530	1,716680
250	0,846131	1,694470	2,466660	3,306420
300	1,444080	2,781820	4,209710	5,472730

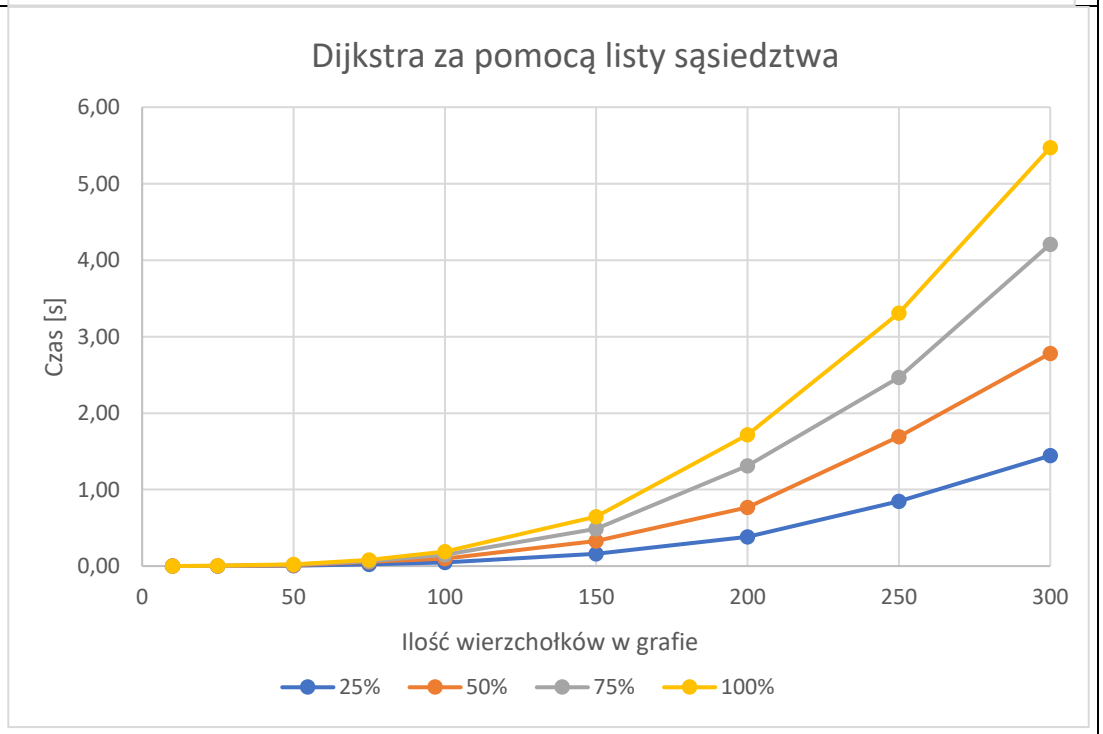
3.2 Wykresy

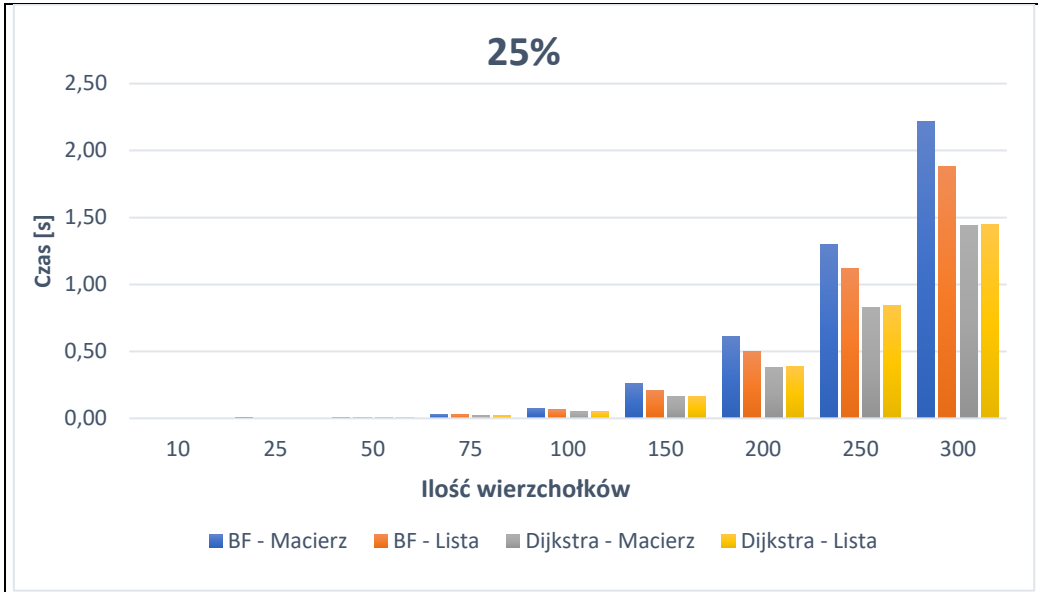
Pierwsza strona zawiera wykresy, w których parametrami są gęstość grafu oraz typ algorytmu, natomiast na drugiej stronie znajdują się wykresy w których parametrami są typ reprezentacji oraz typ algorytmu.

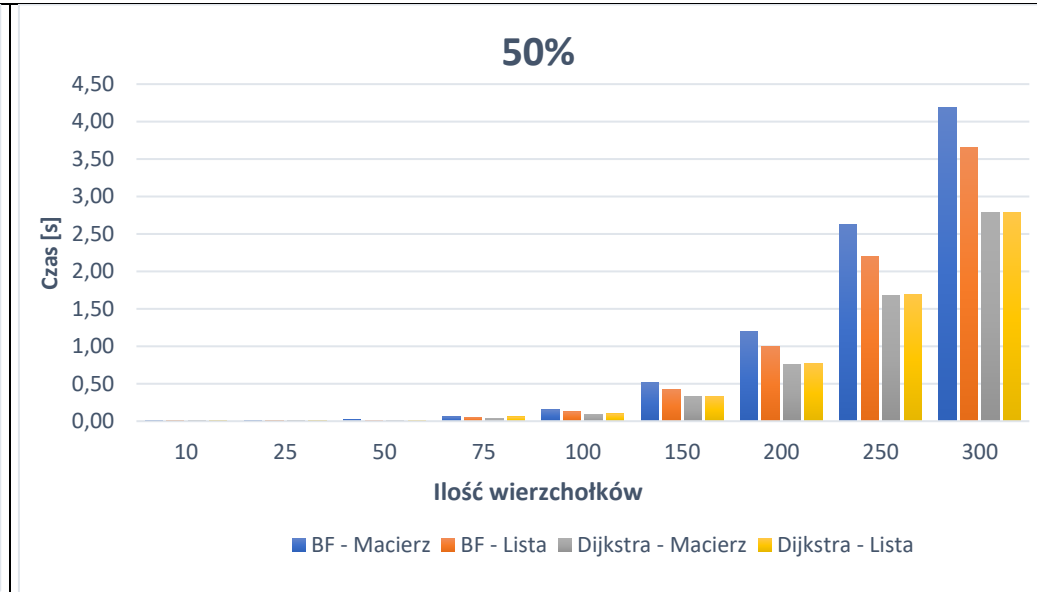

Bellman-Ford za pomocą macierzy sąsiedztwa
Czas [s]
9,00
8,00
7,00
6,00
5,00
4,00
3,00
2,00
1,00
0,00
0
50
100
150
200
250
300
Ilość wierzchołków w grafie
25%
50%
75%
100%

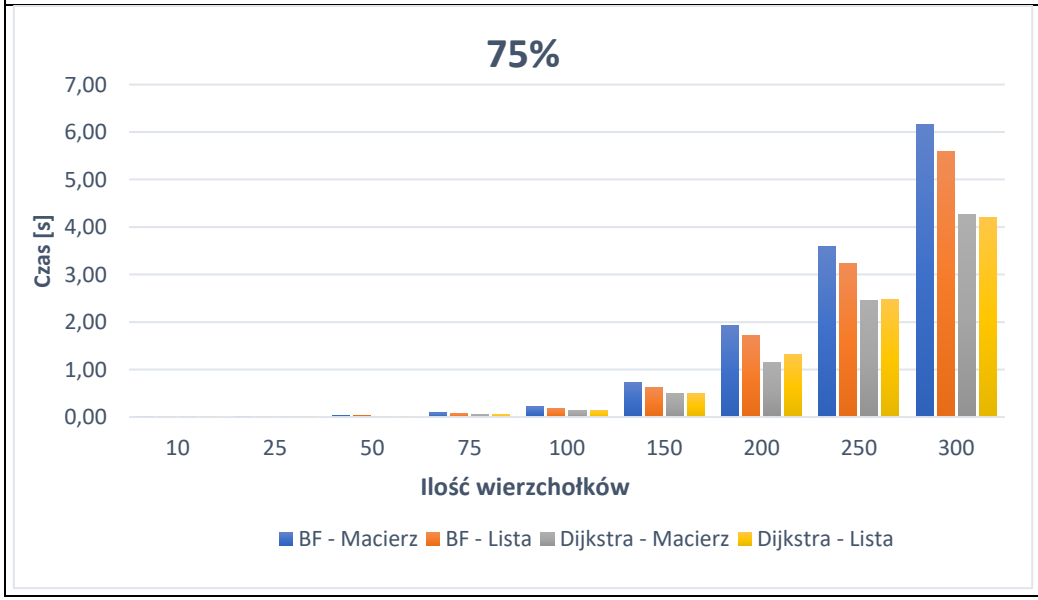

Bellman-Ford za pomocą listy sąsiedztwa
Czas [s]
9,00
8,00
7,00
6,00
5,00
4,00
3,00
2,00
1,00
0,00
0
50
100
150
200
250
300
Ilość wierzchołków w grafie
25%
50%
75%
100%

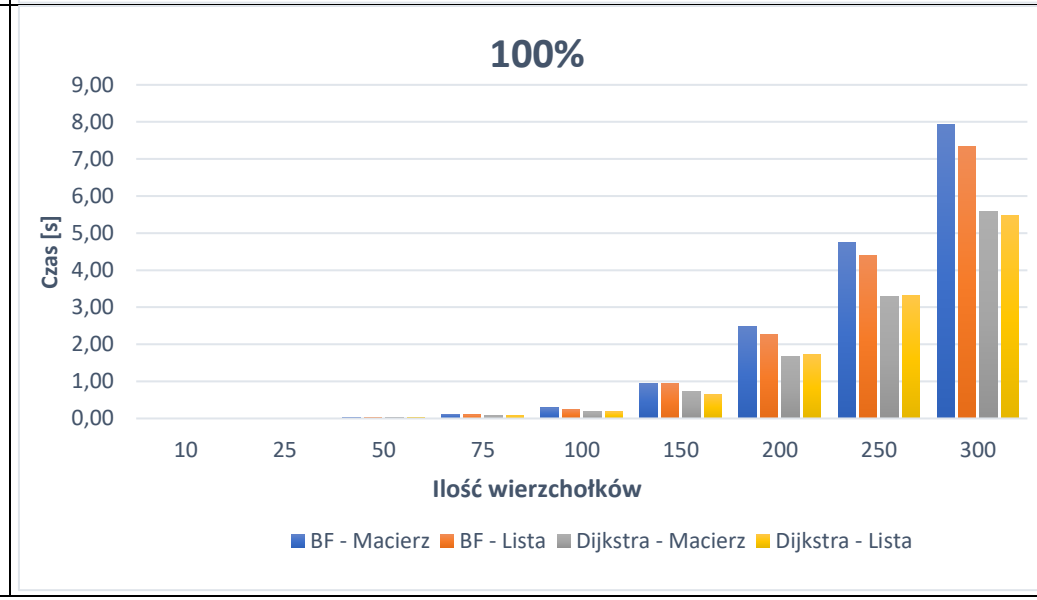

Dijkstra za pomocą macierzy sąsiedztwa
Czas [s]
6,00
5,00
4,00
3,00
2,00
1,00
0,00
0
50
100
150
200
250
300
Ilość wierzchołków w grafie
25%
50%
75%
100%


Dijkstra za pomocą listy sąsiedztwa
Czas [s]
6,00
5,00
4,00
3,00
2,00
1,00
0,00
0
50
100
150
200
250
300
Ilość wierzchołków w grafie
25%
50%
75%
100%


25%
2,50
2,00
1,50
1,00
0,50
0,00
Czas [s]
10
25
50
75
100
150
200
250
300
Ilość wierzchołków
BF - Macierz
BF - Lista
Dijkstra - Macierz
Dijkstra - Lista


50%
4,50
4,00
3,50
3,00
2,50
2,00
1,50
1,00
0,50
0,00
Czas [s]
10
25
50
75
100
150
200
250
300
Ilość wierzchołków
BF - Macierz
BF - Lista
Dijkstra - Macierz
Dijkstra - Lista


75%
7,00
6,00
5,00
4,00
3,00
2,00
1,00
0,00
Czas [s]
10
25
50
75
100
150
200
250
300
Ilość wierzchołków
BF - Macierz
BF - Lista
Dijkstra - Macierz
Dijkstra - Lista


100%
9,00
8,00
7,00
6,00
5,00
4,00
3,00
2,00
1,00
0,00
Czas [s]
10
25
50
75
100
150
200
250
300
Ilość wierzchołków
BF - Macierz
BF - Lista
Dijkstra - Macierz
Dijkstra - Lista

Wnioski z zaprezentowanych wyników:

- Algorytm Dijkstry w obu reprezentacjach wykonuje się znacznie szybciej niż algorytm Bellmana-Forda w obu reprezentacjach
- Implementacja za pomocą macierzy sąsiedztwa wykonuje się dłużej niż za pomocą listy sąsiedztwa w przypadku algorytmu Bellmana-Forda.
- W przypadku algorytmu Dijkstry czasy obu reprezentacji są do siebie zbliżone
- Gęstość grafu wpływa na czas wykonywania w każdym przypadku – im więcej krawędzi tym dłuższy czas wykonania
- Ze wszystkich przypadków najwolniejszym jest algorytm Bellmana-Forda dla macierzy sąsiedztwa, natomiast najszybciej się wykonują obie reprezentacje dla algorytmu Dijkstry

4. Wnioski

Przy doborze algorytmów rozwiązujących problem najkrótszej ścieżki bardziej wydajnym algorytmem jest algorytm Dijkstry co potwierdziły testy. Zbliżone wartości obu reprezentacji dla tego algorytmu najprawdopodobniej wynikają ze sposobu zaimplementowania kolejki priorytetowej o budowie kopca. W celu jej realizacji został wykorzystany kopiec oparty o implementacje tablicową wykorzystywany w poprzednich projektach i najprawdopodobniej on jest źródłem tak zbliżonych czasów, gdyż jego funkcje wywoływane są wielokrotnie w trakcie działania programu. Natomiast w przypadku możliwości wystąpienia ujemnych wag krawędzi konieczny będzie wybór algorytmu Bellmana-Forda w którym graf zaimplementowany za pomocą listy sąsiedztwa jest szybszy.

5. Bibliografia

1) https://stackoverflow.com/questions/25032308/2d-array-of-pointers-classes
2) https://stackoverflow.com/questions/2041517/random-simple-connected-graph-generation-with-given-sparseness
3) https://en.wikipedia.org/wiki/Dense_graph
4) https://pl.wikipedia.org/wiki/Graf_(matematyka)
5) https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
6) https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
7) https://zasoby1.open.agh.edu.pl/dydaktyka/matematyka/c_badania_operacyjne/krok/krok2_08.html