Sprawozdanie z grafów

Nazwisko, Imię, Indeks	Kajetan Zdanowicz, 248933
Prowadzący kurs	Mgr inż. Marcin Ochman
Termin zajęć	WT 15:15
Data oddania sprawozdania	05.05.2020r.

1 Wstęp

Celem projektu jest porównanie wydajności dwóch algorytmów znajdowania najkrótszych ścieżek w grafach ważonych - Dijkstry i Bellmana-Forda oraz dwóch rodzajów reprezentacji grafów - macierzy sąsiedztwa i listy sąsiedztwa. Testy będą obejmowały grafy rozmiarów $10, 50, 100, 250$ oraz 500 wierzchołków o gęstościach $0.25, 0.5, 0.75, 1$. Dla każdego rodzaju grafu wygenerowano 100 losowych instancji i zmierzono czasy znajdowania najkrótszych ścieżek od wierzchołka zerowego. Badania zostały przeprowadzone na grafach nieskierowanych o nieujemnych wagach krawędzi.

2 Opis sposobów reprezentacji grafów

2.1 Macierz sąsiedztwa

Zaimplementowano jako dwuwymiarową tablicę dynamiczną o rozmiarze N x N (N - liczba wierzchołków). W przypadku połączenia dwóch wierzchołków a i b, do pól macierzy o indeksach (a,b) i (b,a) została wpisana waga połączenia. W przypadku braku połączenia lub jednakowym wierzchołkom - przypisywano wartość 0. Taki zabieg pozwolił przyspieszyć późniejsze obliczenia związane z algorytmami.

2.2 Lista sąsiedztwa

Zaimplementowano jako dwuwymiarową tablicę dynamiczną, która dla konkretnego numeru wierzchołka przechowuje obiekt utworzonej wcześniej struktury. Obiekt zawiera pola: wagę i number wierzchołka, z którym ma połączenie. Takie rozwiązanie, w porównaniu ze sposobem implementacji macierzy, nie potrzebuje przechowywania pustych połączeń między wierzchołkami.

3 Opis zaimplementowanych algorytmów

3.1 Algorytm Bellmana-Forda

Dłużej działający niż algorytm Dijkstry, jednakże bardziej uniwersalny. Działa dla ujemnych wag oraz potrafi znaleźć ujemne cykle w grafie. Jego działanie opiera się na metodzie relaksacji dla każdej krawędzi. Relaksacja następuje dokładnie N-1 razy (N - liczba wierzchołków). Polega ona na porównaniu, czy istnieje ścieżka "na około", krótsza niż dotychczasowa. Złożoność czasowa algorytmu: $O(V \cdot E)$ (V- ilość wierzchołków, E - ilość krawędzi).

3.2 Algorytm Dijkstry

Działa tylko dla grafów o wagach dodatnich. Jest szybszy niż algorytm Bellmana-Forda. Metoda polega na utworzeniu dwóch zbiorów. Na początku, pierwszy zawiera wszystkie wierzchołki grafu, a drugi jest tablicą, przechowującą aktualne, najkrótsze odległości wierzchołków od wierzchołka źródłowego. Z pierwszego zbioru usuwany jest element o najmniejszej wadze, a następnie przeprowadzana jest relaksacja. Dla przyspieszenia działania algorytmu, dla macierzy zaimplementowano funkcję znajdującą wierzchołek z najmnejszą wagą (w zależności od punktu odniesienia), a dla listy - kolejkę priorytetową. Dzięki temu, złożoność czasowa wynosi: $O(E \cdot logV)$ (V- ilość wierzchołków, E - ilość krawędzi).

4 Przykłady działania programu - dla listy i macierzy

```
./Graphs
ADJACENCY MATRIX


***** graph size *****8

---graph density--- 0.25
0 0 0 0 252 0 0 0
0 0 0 998 0 0 0 0
0 0 0 708 763 869 0 0
0 998 708 0 0 0 0 0
252 0 763 0 0 0 352 0
0 0 869 0 0 0 0 0
0 0 0 0 352 0 0 154
0 0 0 0 0 0 154 0

Dijkstra
0. 0
1. 2721
2. 1015
3. 1723
4. 252
5. 1884
6. 604
7. 758

Bellman-Ford
0. 0
1. 2721
2. 1015
3. 1723
4. 252
5. 1884
6. 604
7. 758


---graph density--- 1
0 815 760 745 601 454 245 115
815 0 816 787 807 501 894 138
760 816 0 111 914 193 864 322
745 787 111 0 719 419 675 546
601 807 914 719 0 269 669 421
454 501 193 419 269 0 132 807
245 894 864 675 669 132 0 649
115 138 322 546 421 807 649 0

Dijkstra
0. 0
1. 253
2. 437
3. 548
4. 536
5. 377
6. 245
7. 115

Bellman-Ford
0. 0
1. 253
2. 437
3. 548
4. 536
5. 377
6. 245
7. 115
```

```
./Graphs
ADJACENCY LIST


***** graph size ***** 8

---graph density--- 0.25
0. 4(9)
1. 3(8)
2. 4(7)  5(5)  3(6)
3. 1(8)  2(6)
4. 2(7)  6(1)  0(9)
5. 2(5)
6. 7(1)  4(1)
7. 6(1)

Dijkstra
Distances from source
0. 0
1. 30
2. 16
3. 22
4. 9
5. 21
6. 10
7. 11

Bellman-Ford
0. 0
1. 30
2. 16
3. 22
4. 9
5. 21
6. 10
7. 11


---graph density--- 1
0. 1(9)  2(7)  3(7)  4(9)  5(9)  6(9)  7(5)
1. 0(9)  2(2)  3(2)  4(6)  5(1)  6(1)  7(4)
2. 0(7)  1(2)  3(6)  4(4)  5(2)  6(8)  7(5)
3. 0(7)  1(2)  2(6)  4(8)  5(7)  6(1)  7(1)
4. 0(9)  1(6)  2(4)  3(8)  5(3)  6(6)  7(5)
5. 0(9)  1(1)  2(2)  3(7)  4(3)  6(6)  7(3)
6. 0(9)  1(1)  2(8)  3(1)  4(6)  5(6)  7(3)
7. 0(5)  1(4)  2(5)  3(1)  4(5)  5(3)  6(3)

Dijkstra
Distances from source
0. 0
1. 8
2. 7
3. 6
4. 9
5. 8
6. 7
7. 5

Bellman-Ford
0. 0
1. 8
2. 7
3. 6
4. 9
5. 8
6. 7
7. 5
```

5 Wyniki pomiarów

Tabele zawierają uśrednione wyniki czasów realizacji algorytmów w sekundach.

Dijkstra, macierz sąsiedztwa:

		Ilość wierzchołków				
		10	50	100	250	500
Gęstość grafu	0,25	0,00000976	0,000163	0,00062	0,003773	0,015081
	0,5	0,00001037	0,000176	0,000654	0,004001	0,01599
	0,75	0,00001028	0,000174	0,000651	0,003927	0,0157
	1	0,00001006	0,000169	0,000628	0,003869	0,015333

Dijkstra, lista sąsiedztwa:

		Ilość wierzchołków				
		10	50	100	250	500
Gęstość grafu	0,25	0,00000665	0,00025	0,000901	0,005486	0,022023
	0,5	0,00001162	0,000339	0,001286	0,008607	0,034941
	0,75	0,00001459	0,000402	0,001696	0,011352	0,04614
	1	0,00001649	0,000479	0,002007	0,013316	0,056391

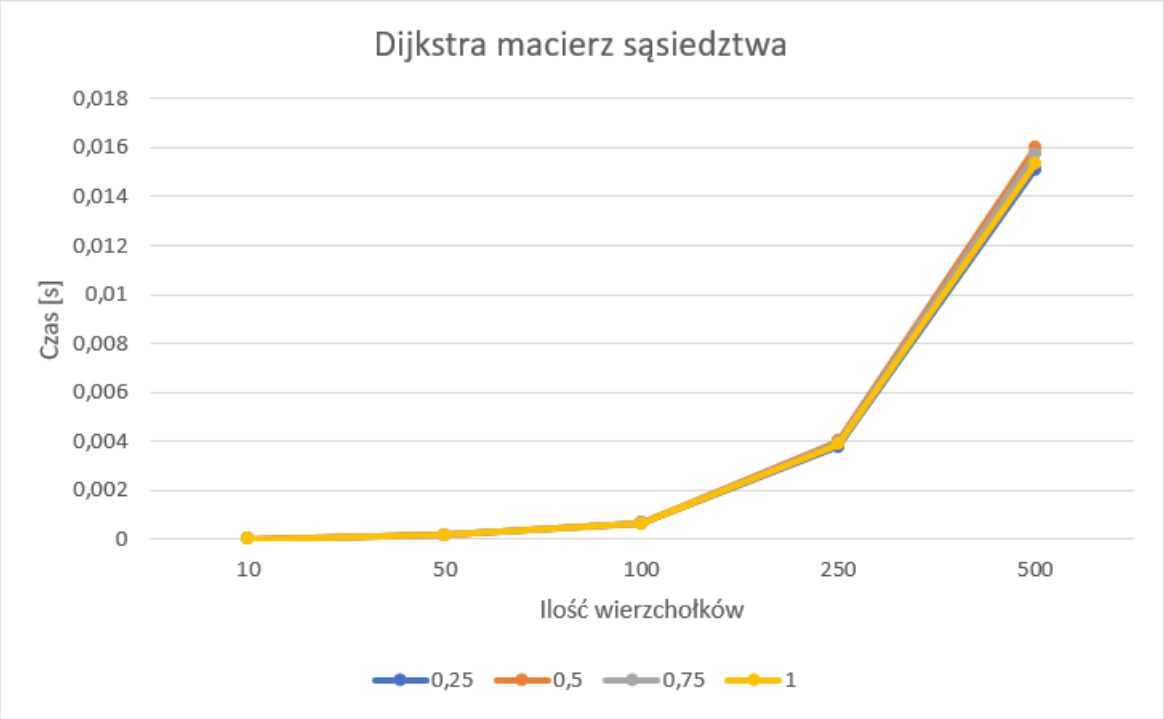
Bellman-Ford, macierz sąsiedztwa:

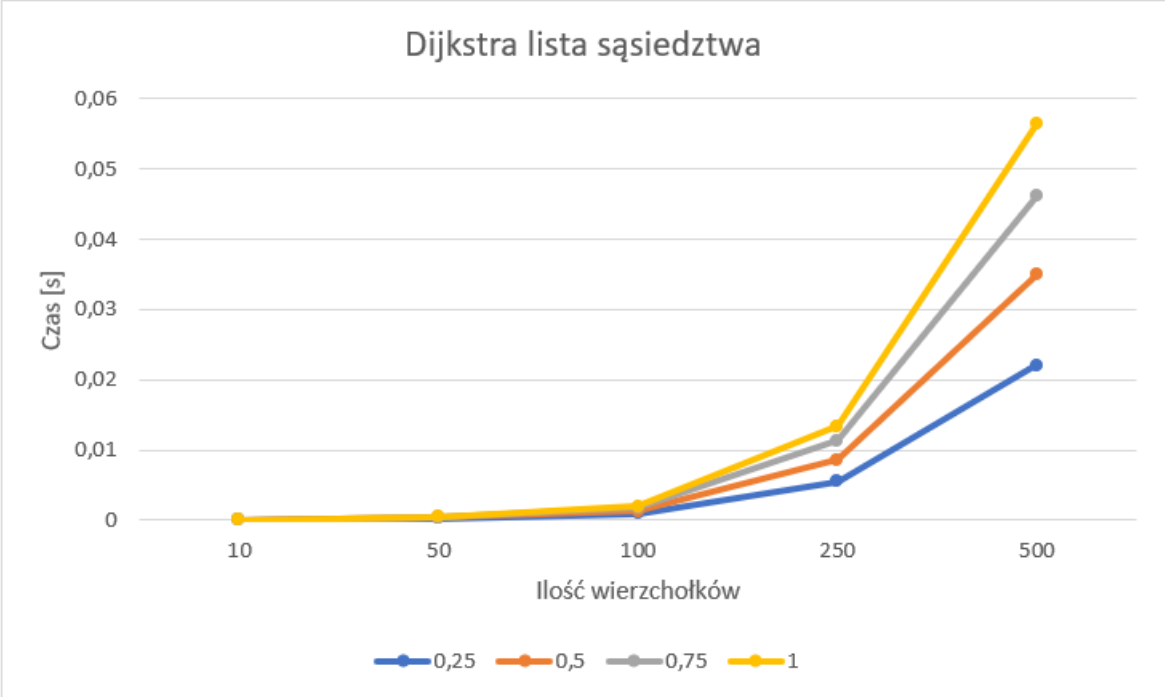
		Ilość wierzchołków				
		10	50	100	250	500
Gęstość grafu	0,25	0,00001705	0,002355	0,018866	0,291269	2,27466
	0,5	0,00001885	0,002143	0,01692	0,256993	1,99682
	0,75	0,00001778	0,001841	0,014316	0,222266	1,70514
	1	0,00001507	0,00151	0,011936	0,185224	1,50413

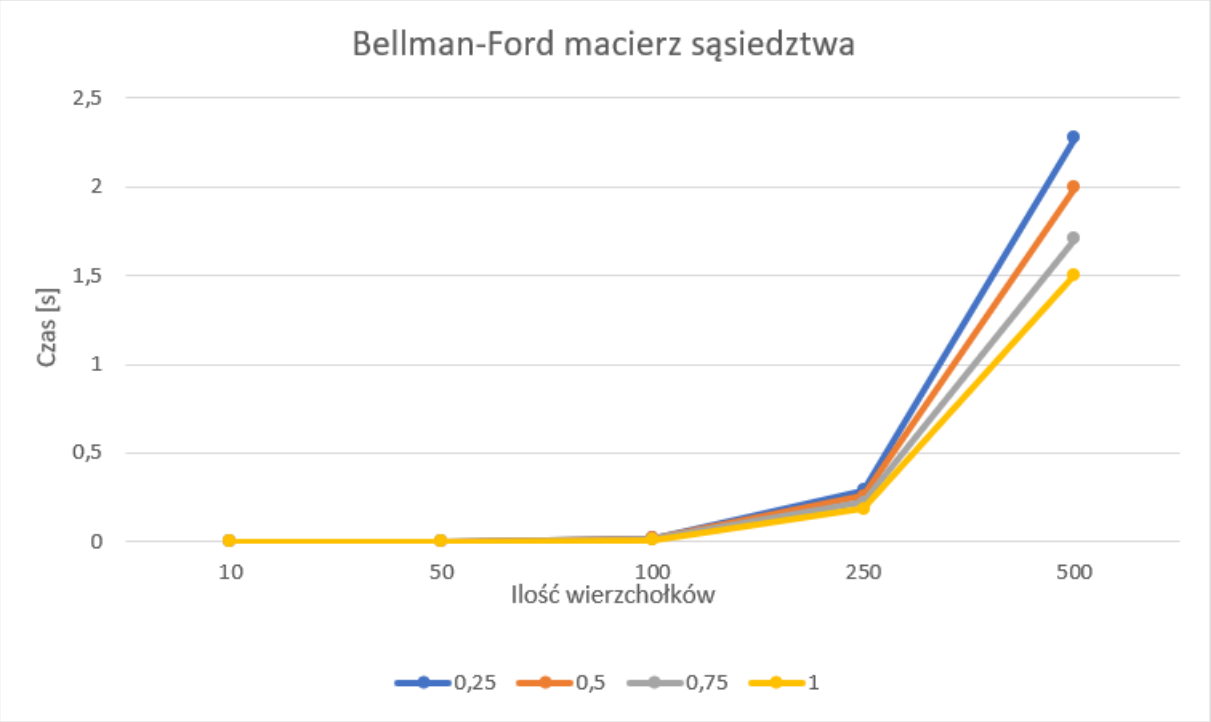
Bellman-Ford, lista sąsiedztwa:

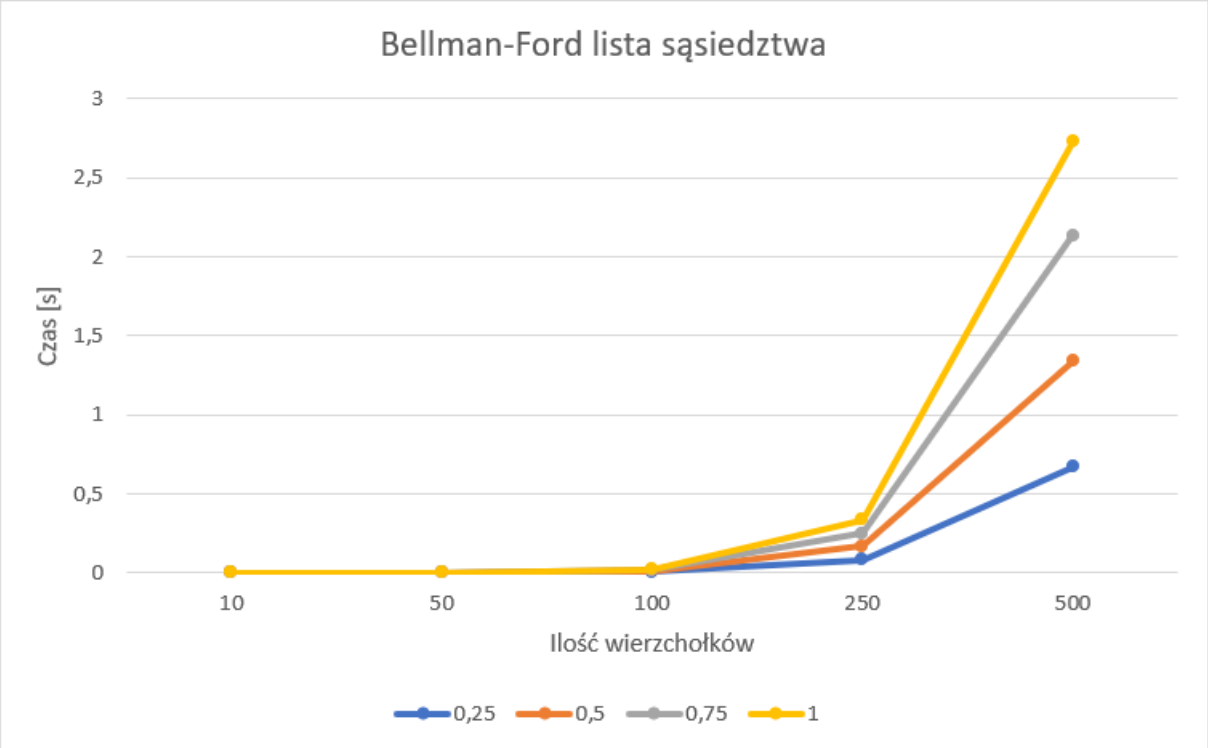
		Ilość wierzchołków				
		10	50	100	250	500
Gęstość grafu	0,25	0,00000803	0,000731	0,005515	0,084615	0,67088
	0,5	0,00001286	0,001363	0,011075	0,168142	1,33843
	0,75	0,00001768	0,002014	0,016389	0,249796	2,13605
	1	0,00002332	0,002675	0,021279	0,333209	2,72544

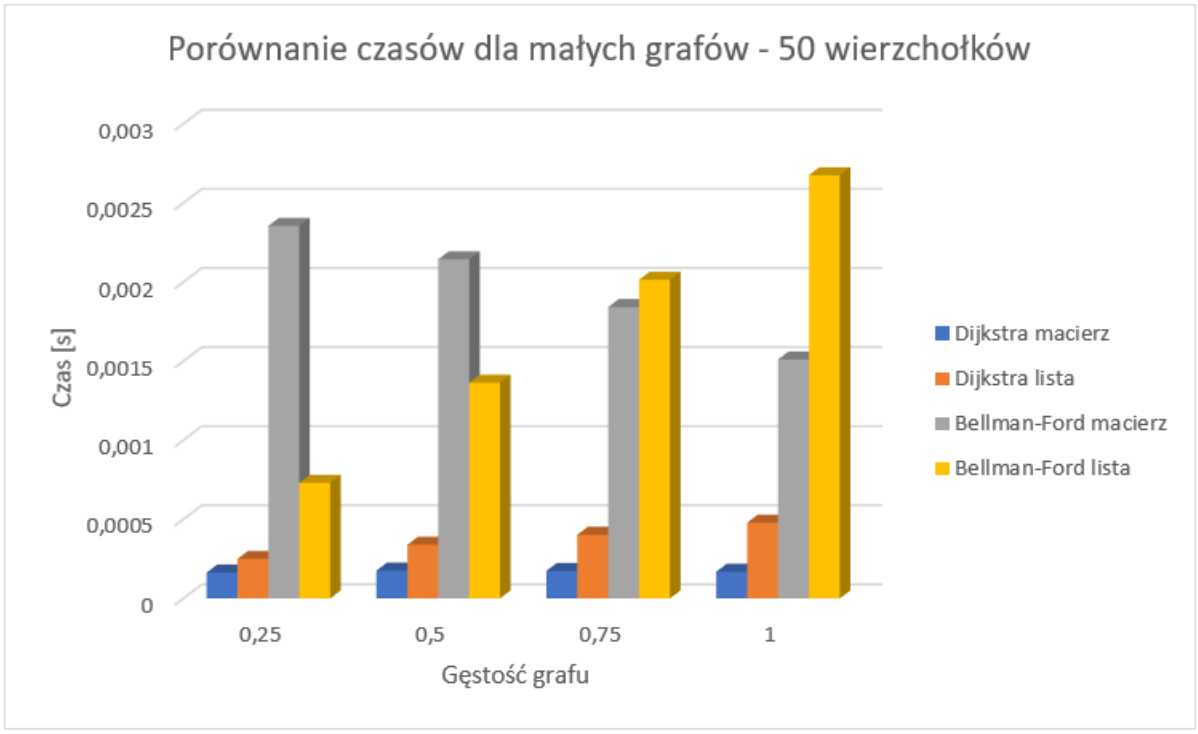
6 Wykresy

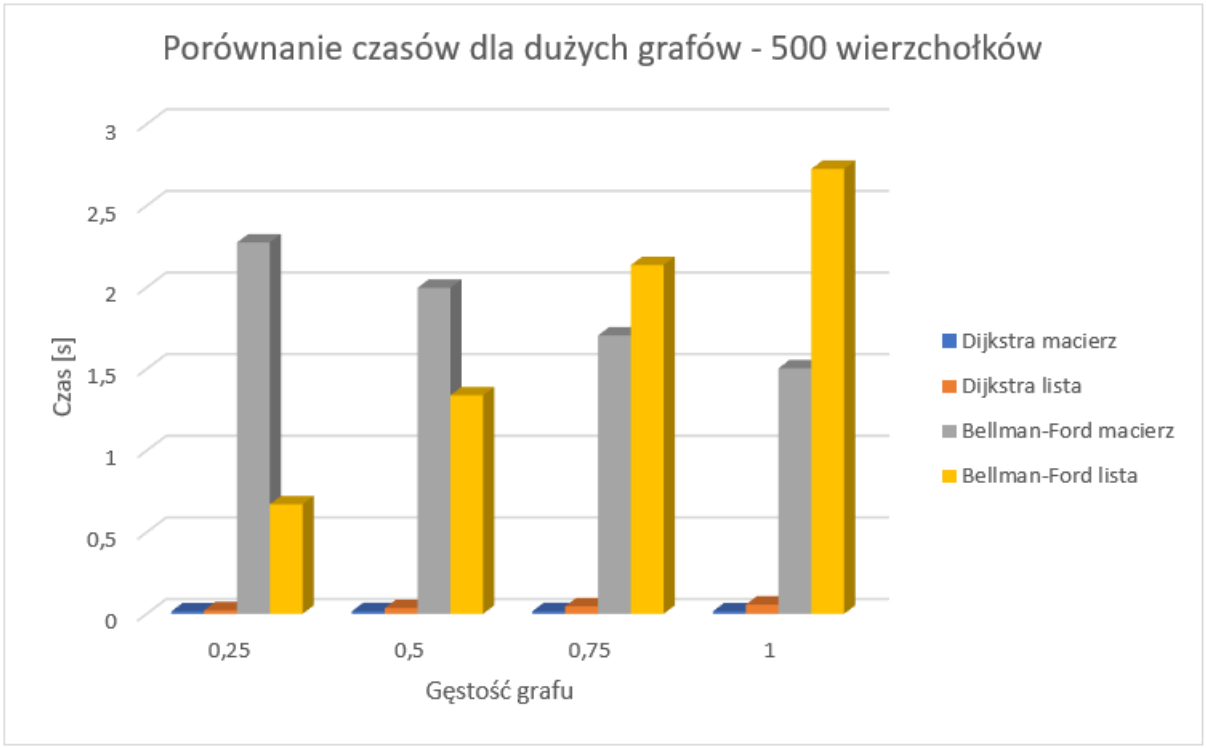

Dijkstra macierz sąsiedztwa
0,018
0,016
0,014
0,012
0,01
0,008
0,006
0,004
0,002
0
Czas [s]
10
50
100
250
500
Ilość wierzchołków
0,25
0,5
0,75
1


Dijkstra lista sąsiedztwa
0,06
0,05
0,04
0,03
0,02
0,01
0
Czas [s]
10
50
100
250
500
Ilość wierzchołków
0,25
0,5
0,75
1


Bellman-Ford macierz sąsiedztwa
2,5
2
1,5
1
0,5
0
Czas [s]
10
50
100
250
500
Ilość wierzchołków
0,25
0,5
0,75
1


Bellman-Ford lista sąsiedztwa
3
2,5
2
1,5
1
0,5
0
Czas [s]
10
50
100
250
500
Ilość wierzchołków
0,25
0,5
0,75
1


Porównanie czasów dla małych grafów - 50 wierzchołków
0,003
0,0025
0,002
0,0015
0,001
0,0005
0
Czas [s]
0,25
0,5
0,75
1
Gęstość grafu
Dijkstra macierz
Dijkstra lista
Bellman-Ford macierz
Bellman-Ford lista


Porównanie czasów dla dużych grafów - 500 wierzchołków
3
2,5
2
1,5
1
0,5
0
Czas [s]
0,25
0,5
0,75
1
Gęstość grafu
Dijkstra macierz
Dijkstra lista
Bellman-Ford macierz
Bellman-Ford lista

7 Wnioski

- Algorytm Dijkstry okazuje się dużo szybszy niż algorytm Bellmana-Forda.
- Dla badań bez ujemnych wag między wierzchołkami, można usunąć część kodu odpowiadającą za szukanie negatywnego cyklu w algorytmie Bellmana-Forda, przyspieszając jego działanie.
- W tej implementacji macierzy i listy, algorytm Dijkstry szybciej działa na macierzy, ponieważ nie musi sięgać po dane zawarte w strukturze dla każdego pola tablicy dwuwymiarowej, tak jak w liście.
- Porównując czasy działania algorytmów dla macierzy i listy, przy dużej ilości wierzchołków i różnych gęstościach grafów, można zauważyć dużo większą rozbieżność wyników działań na grafach przechowywanych w macierzach, aniżeli w listach.
- Wraz ze wzrostem gęstości grafu, spada szybkość działania algorytmu Dijkstry oraz Bellmana-Forda na liście.

8 Literatura

- `https://pl.wikipedia.org/wiki/Algorytm_Dijkstry`
- `https://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda`
- `https://eduinf.waw.pl/inf/alg/001_search/0124.php`