PROJEKTOWANIE ALGORYTMÓW SZTUCZNEJ INTELIGENCJI

PROJEKT 3 – GRAFY

06.05.2020

Sebastian Robak 248994

Prowadząca zajęcia: Mgr inż. Magda Skoczeń

Termin zajęć Środa 18:55

1. Wprowadzenie

Celem projektu było zbadanie złożoności obliczeniowej algorytmu rozwiązującego problem najkrótszej drogi algorytmu Djikstry. Badania zostały przeprowadzone dla dwóch reprezentacji grafu (macierzy i listy), dla różnych ilości wierzchołków (10, 50, 100, 200, 500) oraz dla różnych gęstości (25%, 50%, 75% oraz grafu pełnego). Dla każdego zestawu parametrów (reprezentacja grafu, liczba wierzchołków, gęstość) zostały wygenerowane po 100 losowych grafów a następnie obliczono średni czas działania danego algorytmu.

2. Opis badanego algorytmu

Algorytm Dijkstry polega na znalezieniu najkrótszej drogi w grafie ważonym, z wierzchołka startowego do wszystkich pozostałych wierzchołków. Polega on na metodzie relaksacji, która polega na sprawdzaniu czy poprzez przejście przez daną krawędź nie znajdziemy drogi krótszej niż dotychczasowa.

Algorytm Dijkstry używa kolejki priorytetowej i wybiera najbliższy wierzchołek, który jeszcze nie został użyty i wykonuje relaksacje na wszystkich wychodzących z niego krawędziach. Nie mogą w nim występować ujemne wagi.

Złożoność obliczeniowa to: O (|E| + |V| log|V|), gdzie V to liczba wierzchołków a E to liczba krawędzi.

Pseudokod algorytmu Dijkstry:

```
Dijkstra(G, w, s):
   dla każdego wierzchołka v w V[G] wykonaj
      d[v] := nieskończoność
      poprzednik[v] := niezdefiniowane
   d[s] := 0
   Q := V
   dopóki Q niepuste wykonaj
      u := Zdejmij_Min(Q)
      dla każdego wierzchołka v - sąsiada u wykonaj
         jeżeli d[v] > d[u] + w(u, v) to
            d[v] := d[u] + w(u, v)
            poprzednik[v] := u
```

Źródło: https://pl.wikipedia.org/wiki/Algorytm_Dijkstry

3. Omówienie przebiegu eksperymentów

Eksperyment polegał na liczeniu czasu znalezienia najkrótszej drogi za pomocą algorytmu Dijkstry dla każdego ze 100 grafów, a następnie policzenie średniego czasu wykonania algorytmu na danym zestawie. Takie badanie czasu zostało wykonane dla każdego zestawu parametrów (reprezentacja grafu, liczba wierzchołków, gęstość). Pomiar czas był wykonywany tylko dla algorytmu szukania drogi i nie uwzględniał innych procesów dla rzetelności wyników. Każdy graf był tworzony przy pomocy pseudolosowych liczb całkowitych.

Poniżej zostały zawarte wykresy zależności średniego czasu wykonania algorytmu od ilości wierzchołków. Są przedstawione wykresy dla każdej reprezentacji grafu oraz dla każdej gęstości grafu. W tabelkach zostały zawarte szczegółowe dane pokazujące średni czas wykonania algorytmu w sekundach oraz zaokrąglone do 4 miejsc po przecinku.

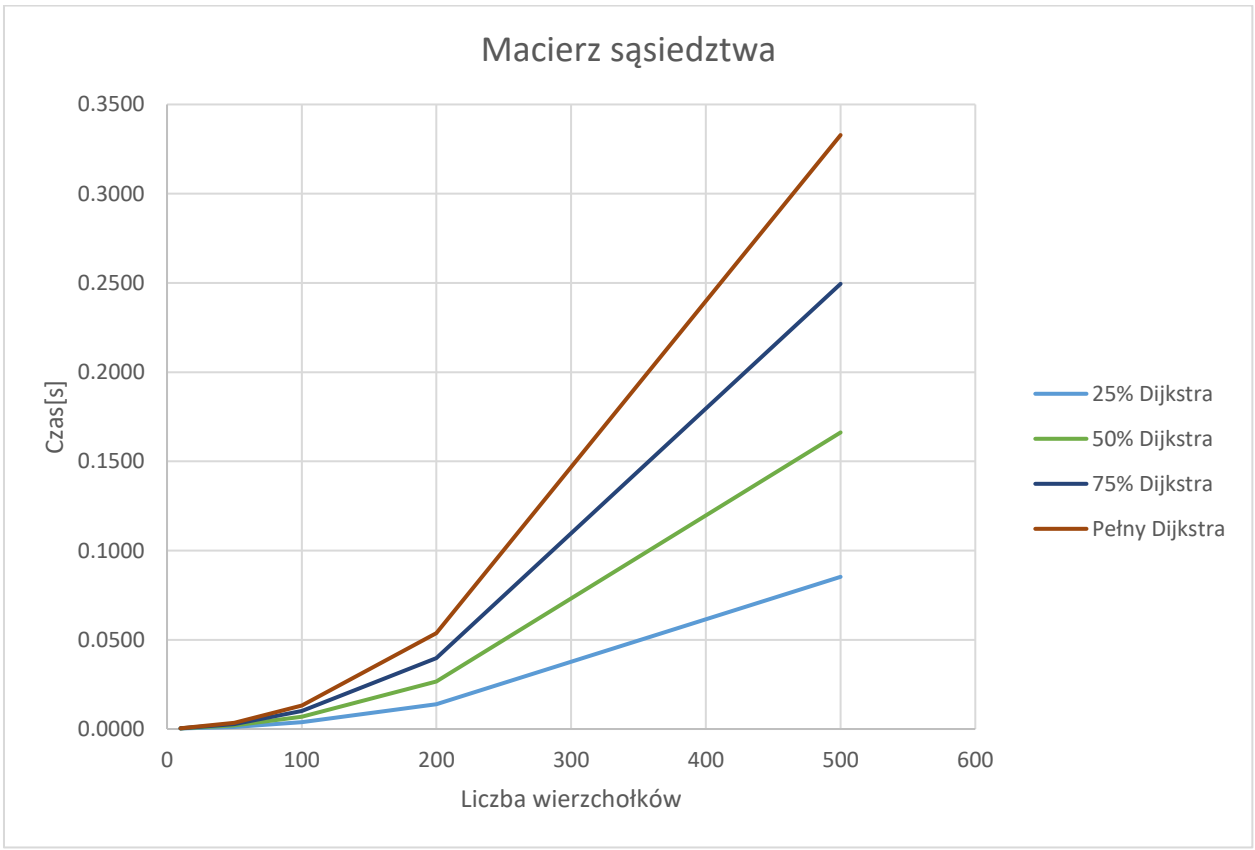
Tabela dla reprezentacji grafu za pomocą listy sąsiedztwa:

Gęstość	L. wierzcholkow	Czas [s]
0.25	10	0.0003
	50	0.0021
	100	0.0071
	200	0.0292
	500	0.1891
0.5	10	0.0005
	50	0.0044
	100	0.0139
	200	0.0588
	500	0.3696
0.75	10	0.0006
	50	0.0052
	100	0.0211
	200	0.0904
	500	0.5551
1	10	0.0005
	50	0.0070
	100	0.0284
	200	0.1248
	500	0.7475

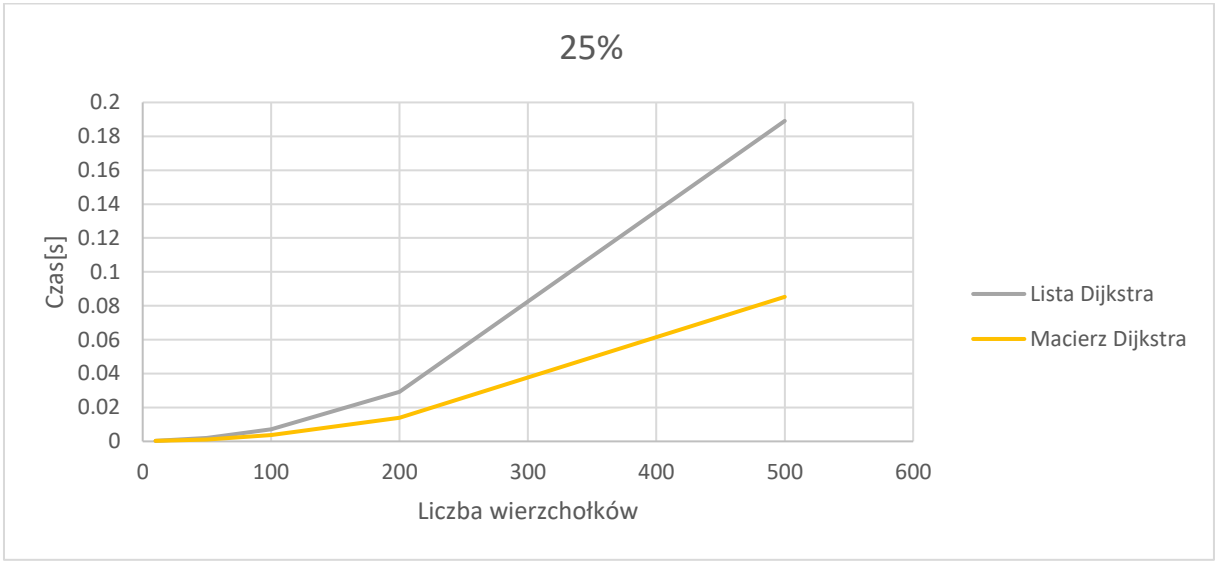
Tabela dla reprezentacji grafu za pomocą macierzy sąsiedztwa:

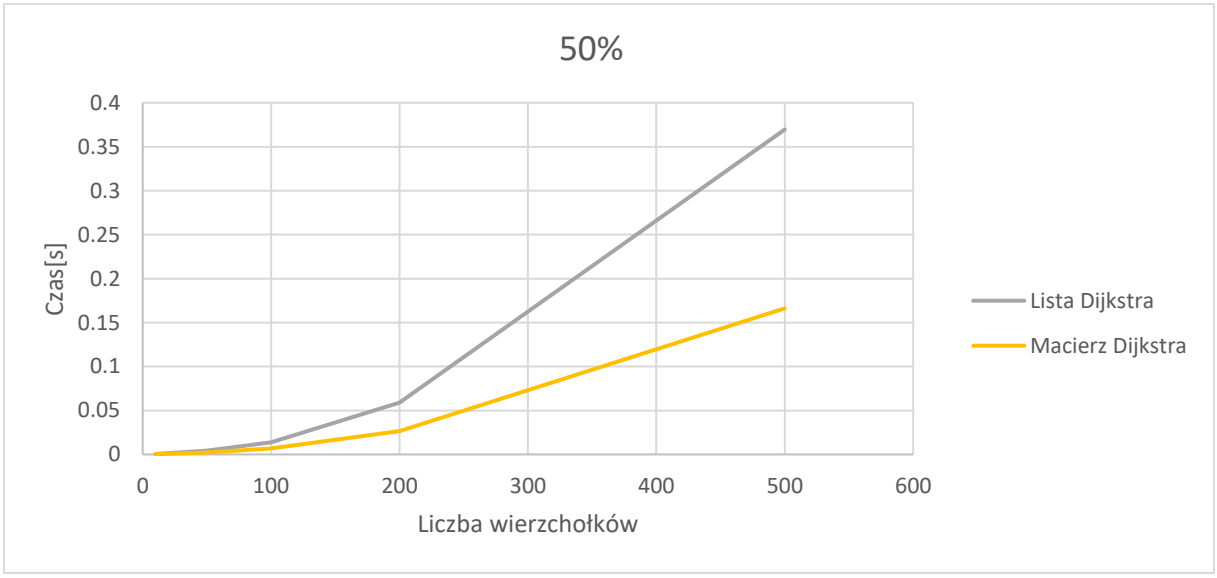
Gęstość	L. wierzcholkow	Czas [s]
0.25	10	0.0003
	50	0.0012
	100	0.0038
	200	0.0139
	500	0.0853
0.5	10	0.0004
	50	0.0020
	100	0.0069
	200	0.0266
	500	0.1662
0.75	10	0.0004
	50	0.0027
	100	0.0101
	200	0.0396
	500	0.2495
1	10	0.0004
	50	0.0035
	100	0.0132
	200	0.0537
	500	0.3329

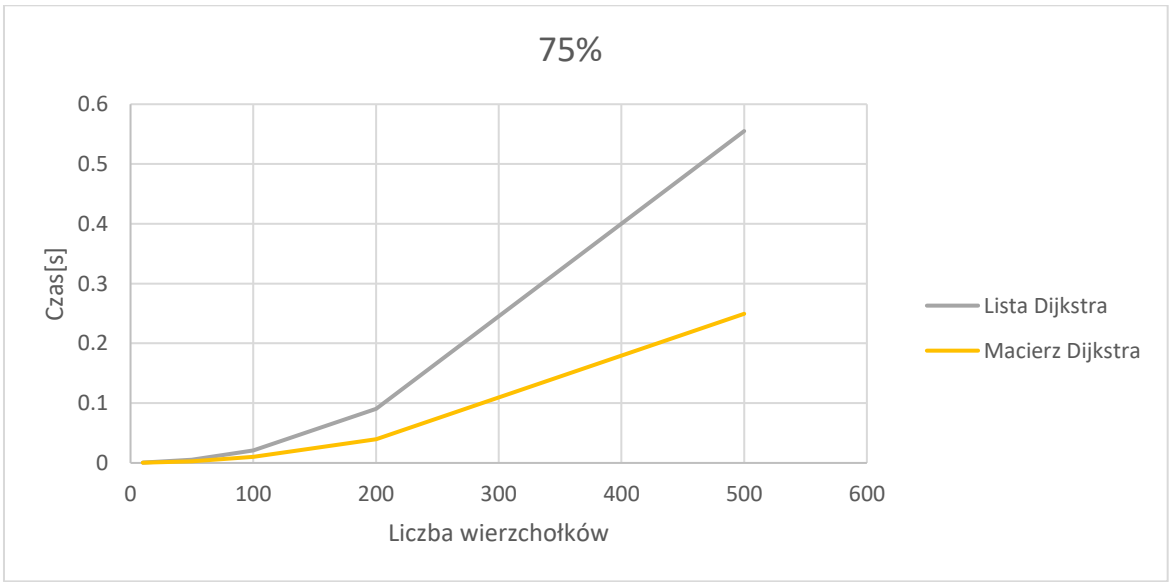
Wykresy dla różnych reprezentacji grafu:



Wykresy dla różnych gęstości:







4. Podsumowanie i wnioski

- Uzyskane wyniki pokazują, że dla większej gęstości grafu algorytm działa wolniej, co jest poprawne, ponieważ jest więcej obliczeń do wykonania
- Uzyskane wyniki wskazują także na złożoność obliczeniową algorytmu Dijkstry podobną do tej podanej w literaturze, czyli $O(|E| + |V| \log|V|)$
- Wyniki pokazały, że algorytm dla reprezentacji grafu za pomocą listy sąsiedztwa jest wolniejszy niż za pomocą macierzy sąsiedztwa, nie zgadza się to z literaturą według której wyniki powinny być odwrotne. Jest to spowodowane metodą implementacji macierzy sąsiedztwa oraz listy sąsiedztwa, przechodzenie po liście by dodać lub usunąć element w liście sąsiedztwa trwa dłużej, niż dostęp w macierzy sąsiedztwa.

5. Bibliografia

https://en.wikipedia.org/wiki/Graph_(abstract_data_type)

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

https://www.softwaretestinghelp.com/graph-implementation-cpp/

https://www.tutorialspoint.com/cplusplus-program-for-dijkstra-s-shortest-path-algorithm

https://eduinf.waw.pl/inf/alg/001_search/0138.php