PAMSI - Projekt nr.3

Grafy

Dawid Krekora 254003

10 czerwca 2022

Spis treści

1 Wprowadzenie

Zadanie projektowe polegało na napisaniu dwóch implementacji grafów:

- Za pomocą listy sąsiedztwa
- Za pomocą macierzy sąsiedztwa

Następnie należało zaimplementować dwa algorytmy operujące na grafach, zajmujące się problemami znajdowania najkrótszej ścieżki w grafie. Algorytmy te zostały poddane testowi efektywności w zależności od gestości grafu oraz ilości krawędzi. Grafy które zostały poddane testom były grafami ważonymi, skierowanymi.

2 Opis badanych algorytmów

W zadaniu projektowym poddano badaniom dwa algorytmy wyszukiwania najkrótszej ścieżki. Algorytmy te zostały bliżej przedstawione poniżej.

2.1 Algorytm Dijkstry

Algorytm Dijkstry jest najpopularniejszym algorytmem (ma szerokie spektrum zastosowań) zajmującym się badaniem problemu najkrótszej drogi. Spowodowane jest to głównie jego efektywnością działania. Szacuje się, że złożoność obliczeniowa algorytmu Dijkstry wynosi $\mathbf{O}(Elog_V)$ gdzie:

- E - liczba krawędzi grafu
- V - liczba wierzchołków w grafie

Wymieniona złożoność dotyczy implementacji tablicy kosztów dojścia do wierzchołków w postaci kolejki priorytetowej. W naszym przypadku użyto zwykłej iteracji tablicowej która jest mniej efektywna. Działanie algorytmu Dijkstry opiera się zależnościach między wierzchołkami incydentnymi aktualizując w każdej iteracji drogę potrzebną do dotarcia do sąsiadów wybranego wierzchołka. Sam wierzchołek natomiast jest przenoszony do zbioru wierzchołków przetworzonych. Dzieje się tak do momentu aż wszystkie wierzchołki nie zostaną przetworzone.

2.2 Algorytm Bellmana-Forda

Algorytm Bellmana-Forda posiada podobne działanie do algorytmu Dijkstry - podobne dlatego, że również możemy za jego pomocą znaleźć drogę do wszystkich wierzchołków i określić jej koszt dojścia. W praktyce jednak rozszerza nam działanie algorytmu Dijstry ponieważ jest w stanie wykryć negatywne cykle (cykle w których wraz z każdym przejściem koszt dojścia do wierzchołka maleje), które fałszują wyniki końcowe. W takim przypadku algorytm wykryje taką nieprawidłowość i nas o tym poinformuje. Niestety funkcjonalność ta jest obarczona dodatkową złożonością obliczeniową która szacowana jest na $\mathbf{O}(E * V)$. Wynika to przede wszystkim z konieczności sprawdzania wszystkich krawędzi grafu w każdej z iteracji (których suma jest nie mniejsza niż całkowita liczba wierzchołków w grafie). Jeżeli zostanie wykryty negatywny cykl algorytm zwróci wartość 1. Natomiast w przypadku poprawnego grafu wartość 0, co oznacza paradoksalnie: wszystko jest poprawne, nie wykryto (wartość 0) negatywnych cykli. Następnie uzyskamy rezultat działań identyczny z algorytmem Dijkstry.

3 Przebieg eksperymentów

Do eksperymentów użyto funkcjonalność bibloteki *chrono* z pakietu biblotek C++. Dla każdej ilości wierzchołków (10,50,100,150,200) i każdej gęstości grafu (0.25,0.5,0.75,1) wywołano 100 losowych instancji grafów ważonych, skierowanych. Czas działania każdej instancji dla każdego testu (łącznie 20 testów) zarejestrowano za pomocą wspomnianej wcześniej bibloteki, dodano całowity przebieg czasowy, a następnie określono wartość średnią dla każdego testu. Wyniki przedstawione w tabeli (rys.1) zostały naniesione według różnych kryteriów zawartych w tytułach wykresów (rys.2-7).

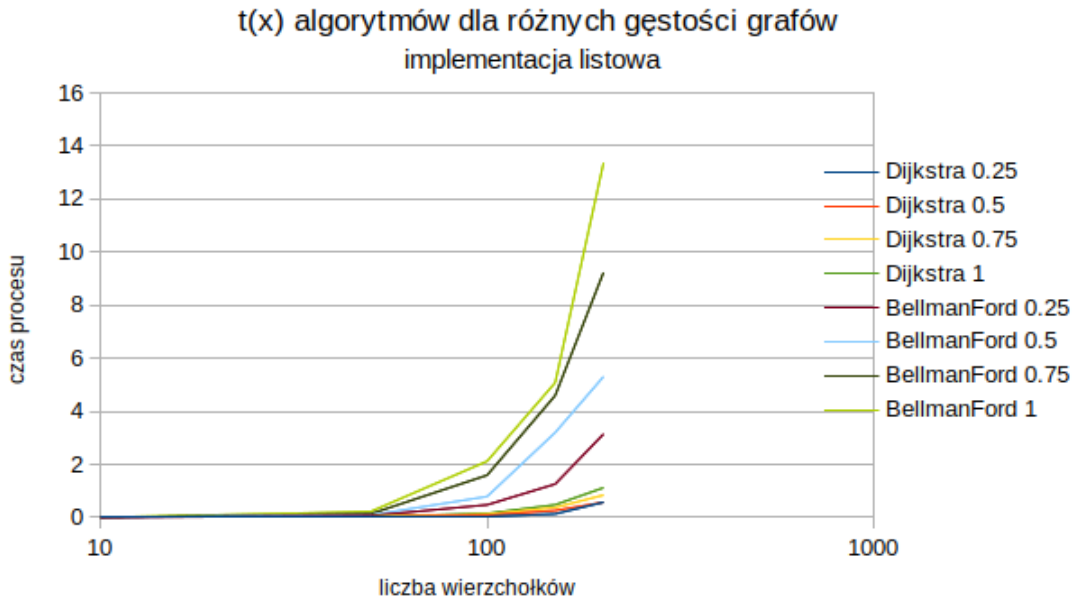
	dijkstra lista			
ilosc V \ gestosc	0,25	0,5	0,75	1
10	0,000257534	0,000299463	0,000569488	0,000411942
50	0,00702808	0,0114114	0,0172912	0,0226372
100	0,0416554	0,0783902	0,119738	0,151215
150	0,13351	0,261492	0,373265	0,475792
200	0,571321	0,571321	0,840752	1,11873

	dijkstra macierz			
ilosc V \ gestosc	0,25	0,5	0,75	1
10	0,000161824	0,000237683	0,000304991	0,000366376
50	0,00747518	0,0112332	0,0190721	0,0215441
100	0,0427588	0,0808704	0,111519	0,156546
150	0,135219	0,244089	0,357583	0,468814
200	0,287225	0,576563	0,831475	1,14297

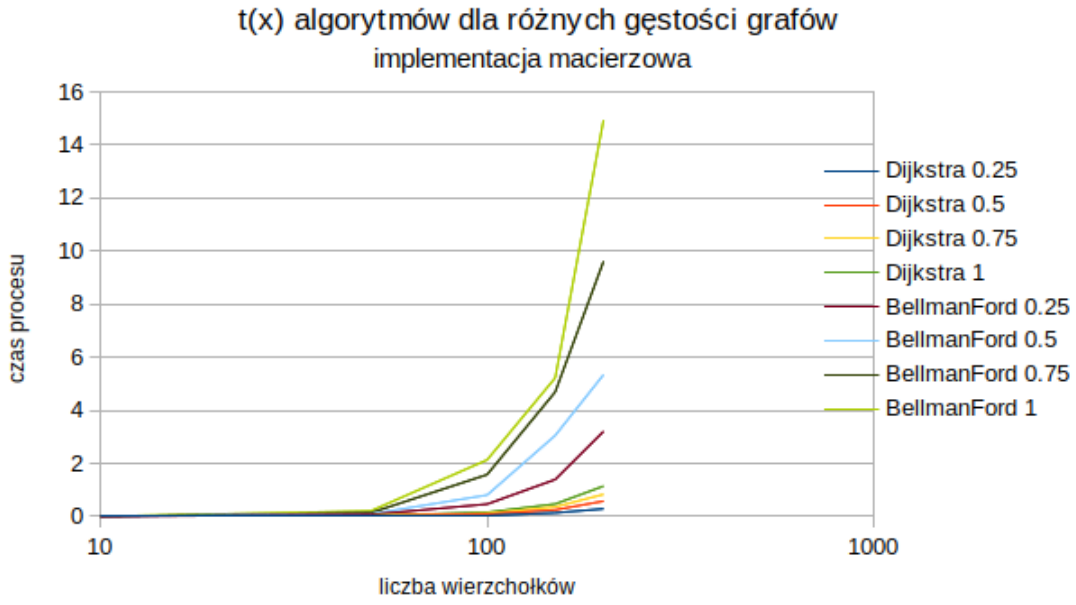
	bellmanFord lista			
ilosc V \ gestosc	0,25	0,5	0,75	1
10	0,000653168	0,000851569	0,000920317	0,00160233
50	0,0620042	0,0892149	0,131915	0,218895
100	0,473852	0,786601	1,59265	2,11801
150	1,25718	3,20279	4,59827	5,08305
200	3,14662	5,3088	9,23627	13,366

	bellmanFord macierz			
ilosc V \ gestosc	0,25	0,5	0,75	1
10	0,00069346	0,00076096	0,00141458	0,00165594
50	0,0686844	0,0902758	0,133177	0,203749
100	0,463102	0,805735	1,57692	2,1314
150	1,39212	3,05203	4,69363	5,2234
200	3,20626	5,34356	9,61478	14,9454

Rysunek 1: Tabela wyników czasu działania algorytmów (wartości średnie)

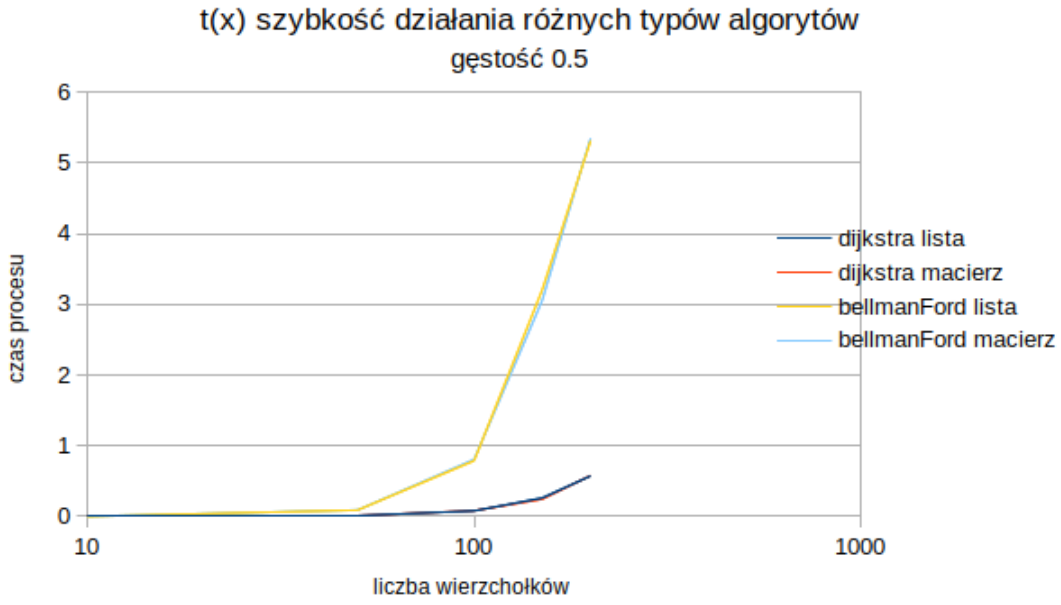


Rysunek 2: Funkcja czasu działania algorytmu od ilości jego wierzchołków - implementacja listowa

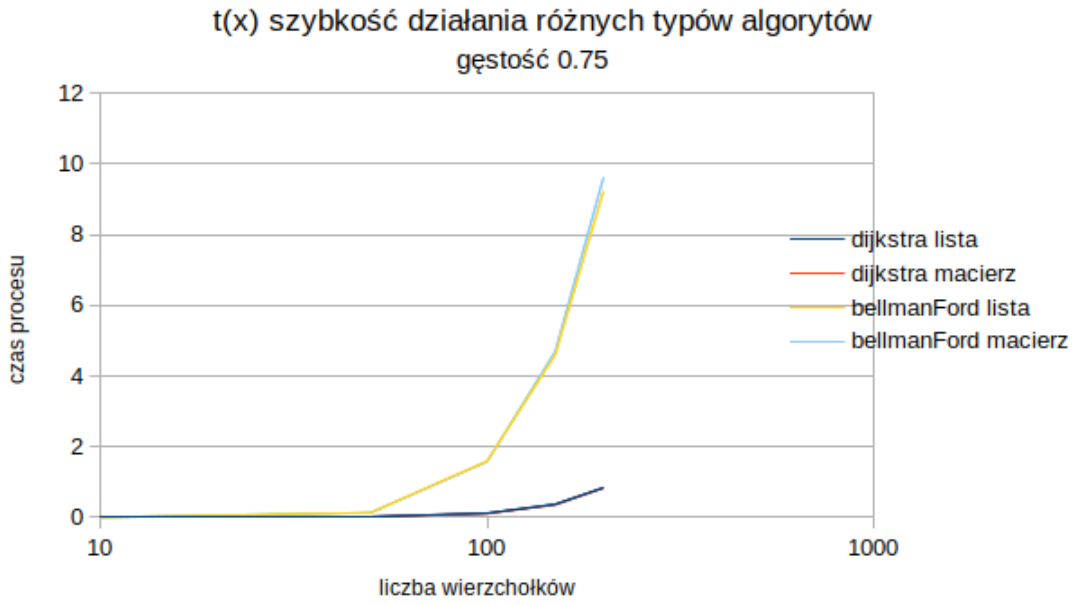


Rysunek 3: Funkcja czasu działania algorytmu od ilości jego wierzchołków - implementacja macierzowa

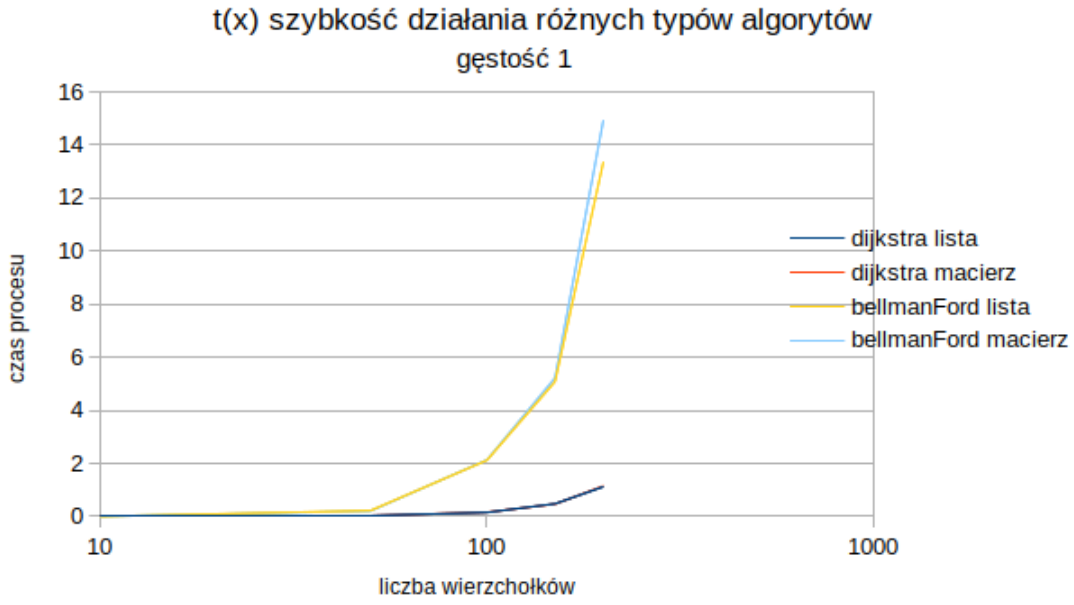
Rysunek 4: Funkcja czasu działania algorytmu od ilości jego wierzchołków - gestość 0.25 dla różnych implementacji



Rysunek 5: Funkcja czasu działania algorytmu od ilości jego wierzchołków - gestość 0.5 dla różnych implementacji



Rysunek 6: Funkcja czasu działania algorytmu od ilości jego wierzchołków - gestość 0.75 dla różnych implementacji



Rysunek 7: Funkcja czasu działania algorytmu od ilości jego wierzchołków - gestość 1 dla różnych implementacji

4 Podsumowanie i wnioski

- Czas działania algorytmów był zgodny z naszymi założeniami. Dla małej ilości wierzchołków nie miała znaczenia ani gęstość grafu, ani sposób jego implementacji.
- Wraz ze wzrostem liczby wierzchołków można było zauważyć znaczące różnice w długości działania. Algorytm Bellmana-Forda okazał się niezwykle konsumpcyjny. Dla maksymalnych gęstości czas jego działania był ponad 10 krotnie większy od czasu działania algorytmu Dijkstry. Sposób implementacji nie miał w tym wypadku większego znaczenia (różnice czasowe były niewielkie - na korzyść w jedną lub drugą stronę). Z najbardziej złożonym problemem najlepiej poradziła sobie implementacja listowa.
- Zdecydowania lepszym algorytmem znajdowania najkrótszej ścieżki okazał się algorytm Dijkstry i to on jest używany w przytłaczającej przewadze nad algorytmem Bellmana-Forda. Wynika to z faktu, że rzadko mamy do czynienia z reprezentacją danych w której występuje negatywny cykl. Zwycięski algorytm znalazł zastosowanie m.in. w wyznaczaniu drogi na mapie - oczywiście z nieco efektywniejszej formie.

5 Bibliografia

- Piotr Wróblewski, W: Helion, Ąlgorytmy, struktury danych i techniki programowania", 2009
- https://baeldung.com/cs/bellman-ford
- https://eduinf.waw.pl/inf/alg/001 search/index.php
- Michael T.G,Roberto T., David M.,W: John Wiley and Sons Inc. , "Data structures and algorithms in $C++$", 2009