Wtorki 16:50
Grupa I3
Kierunek Informatyka
Wydział Informatyki
Politechnika Poznańska

Algorytmy i struktury danych

Sprawozdanie z zadania w zespołach nr. 4
prowadząca: dr hab. inż. Małgorzata Sterna, prof PP

Algorytmy z powracaniem

autorzy:
Piotr Więtczak nr indeksu 132339
Tomasz Chudziak nr indeksu 136691

22 maja 2018

1 Opis implementacji

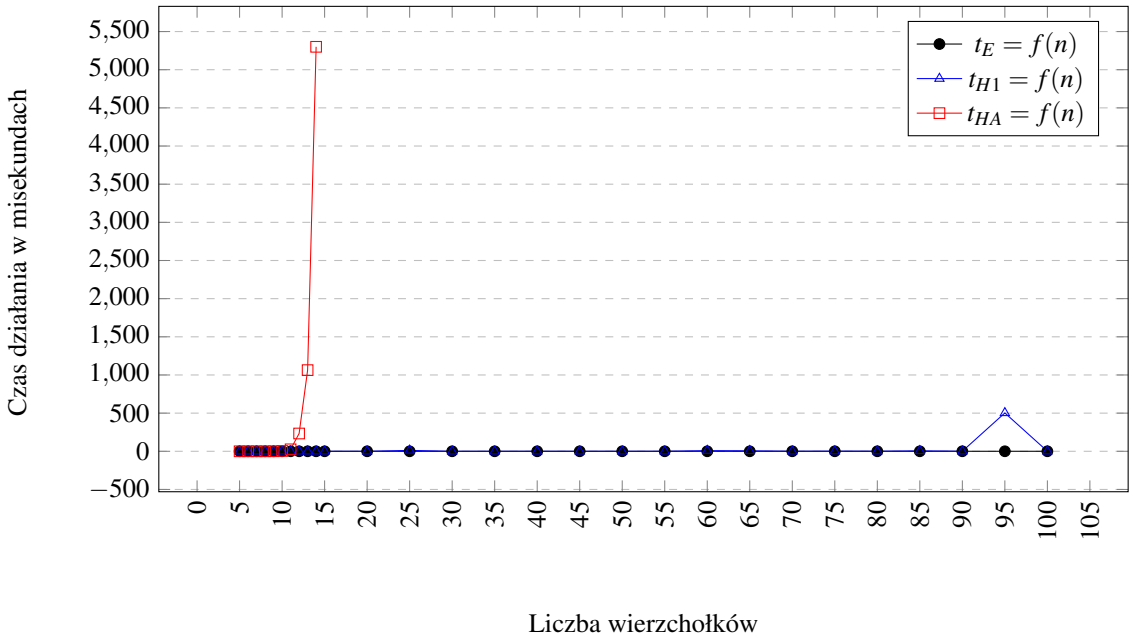
Do implementacji algorytmów poszukujących cyklu Eulera (E), pojedynczego cyklu Hamiltona $(H1)$ i wszystkich cykli Hamiltona użyliśmy języka C++. Do pomiarów czasu wykorzystaliśmy klasę $std :: chrono :: high\_resolution\_clock$ z biblioteki *chrono*. Do reprezentacji grafu zastosowaliśmy macierz sąsiedztwa, ze względu na intuicyjne działanie oraz łatwość implementacj.

2 Czasy działania algorytmów

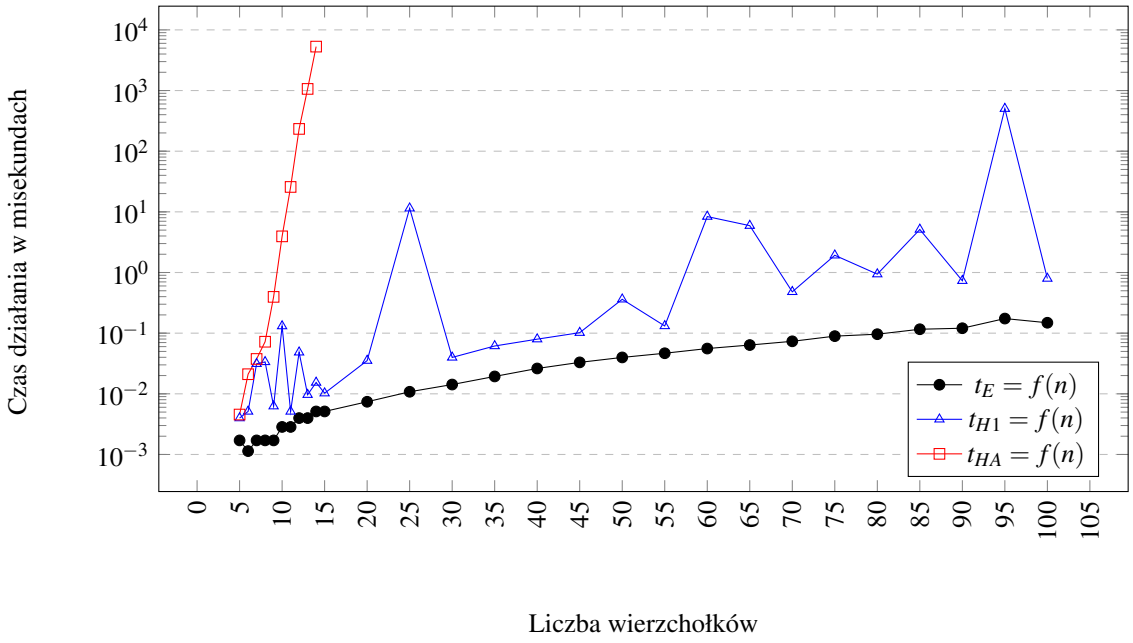
Tabela przedstawiająca czasy działania algorytmów

Liczba wierzchołków	t_E dla $d = 0.6$ $[ms]$	t_{H1} dla $d = 0.6$ $[ms]$	t_{HA} dla $d = 0.6$ $[ms]$
5	0.001	0.003	0.004
6	0.001	0.005	0.021
7	0.001	0.031	0.037
8	0.001	0.033	0.072
9	0.001	0.006	0.396
10	0.002	0.130	3.945
11	0.002	0.005	25.734
12	0.003	0.048	232.968
13	0.003	0.009	1063.510
14	0.005	0.015	5300.210
15	0.005	0.010	*przerwano*
20	0.007	0.035	*przerwano*
25	0.010	11.446	*przerwano*
30	0.014	0.039	*przerwano*
35	0.019	0.061	*przerwano*
40	0.026	0.079	*przerwano*
45	0.032	0.101	*przerwano*
50	0.039	0.361	*przerwano*
55	0.046	0.131	*przerwano*
60	0.055	8.360	*przerwano*
65	0.063	5.921	*przerwano*
70	0.073	0.480	*przerwano*
75	0.089	1.922	*przerwano*
80	0.096	0.939	*przerwano*
85	0.116	5.121	*przerwano*
90	0.120	0.731	*przerwano*
95	0.174	502.357	*przerwano*
100	0.148	0.795	*przerwano*

Wykres przedstawiający czasy działania algorytmów dla $d = 0.6$



Wykres przedstawiający czasy działania algorytmów dla $d = 0.6$ skala logarytmiczna



Problemy znajdowania cyklu Eulera i cyklu Hamiltona dotyczą przeszukiwania grafu.

Znajdowanie cyklu Eulera należy do klasy problemów łatwych (P), czyli takich dla których potrafimy znaleźć algorytm rozwiązujący ten problem w czasie wielomianowym.

Złożoność obliczeniowa algorytmu znajdowania cyklu Eulera, przy zastosowaniu listy sąsiedztwa, wynosi $O(m)$, gdzie m - liczba krawędzi, ponieważ podczas przeszukiwania grafu trzeba przejść po wszystkich krawędziach, jest to złożoność wielomianowa.

Znajdowanie cyklu Hamiltona należy do problemów NP-zupełnych, które są podklasą problemów trudnych

(NP), dla problemów które należą do klasy NP i NP-zupełnych nie znamy rozwiązań działających w czasie wielomianowym lub mniejszym, czyli są to zadania o o złożoności co najmniej wykładniczej. Do problemów NP-zupełnych transformują się wielomianowo wszystkie problem z klasy NP. Rozwiązując problem NP-zupełny rozwiązujemy wszystkie problemy z tej podklasy, dlatego znajdując rozwiązanie jednego takiego problemu w czasie wielomianowym, znajdziemy rozwiązanie wielomianowe dla wszystkich problemów NP-zupełnych.

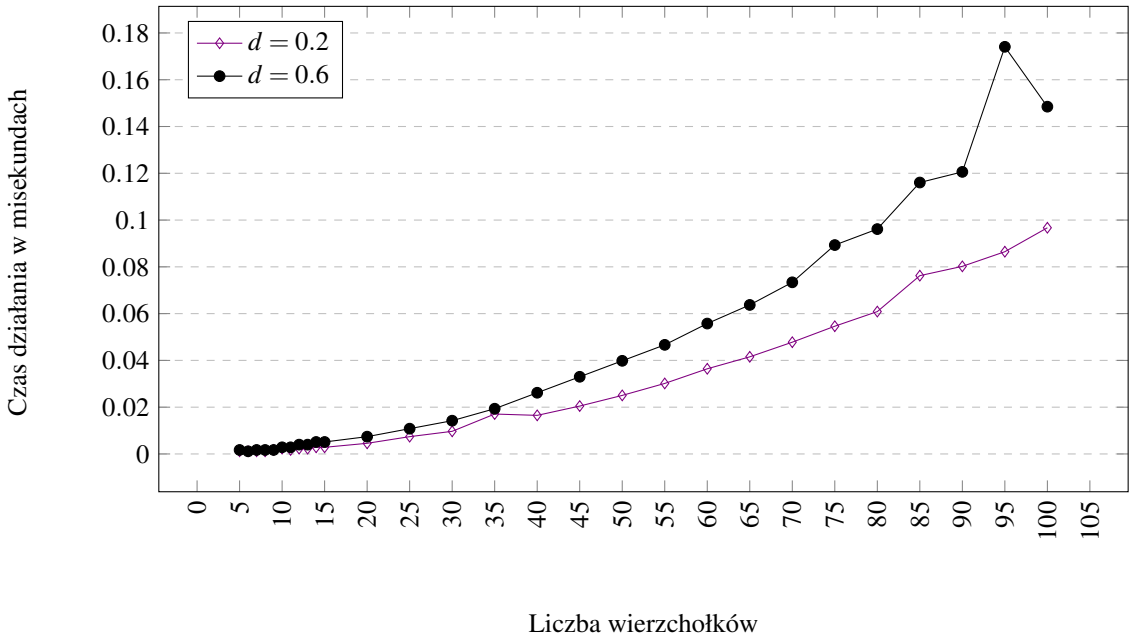
Złożoność obliczeniowa algorytmu znajdowania pojedynczego i wszystkich cykli Hamiltona wynosi $O(n!)$, gdzie n - liczba wierzchołków, ponieważ w najgorszym przypadku należy sprawdzić wszystkie możliwe permutacje, jest to złożoność wykładnicza.

3 Czasy poszukiwania cyklu Eulera dla różnych wartości d

Tabela przedstawiająca T_E dla różnych wartości d

Liczba wierzchołków	t_E dla $d = 0.2 \quad [ms]$	t_E dla $d = 0.6 \quad [ms]$
5	0.001	0.001
6	0.001	0.001
7	0.001	0.001
8	0.001	0.001
9	0.001	0.001
10	0.002	0.002
11	0.001	0.002
12	0.002	0.003
13	0.002	0.003
14	0.002	0.005
15	0.002	0.005
20	0.004	0.007
25	0.007	0.010
30	0.009	0.014
35	0.017	0.019
40	0.016	0.026
45	0.020	0.032
50	0.025	0.039
55	0.030	0.046
60	0.036	0.055
65	0.041	0.063
70	0.047	0.073
75	0.054	0.089
80	0.060	0.096
85	0.076	0.116
90	0.080	0.120
95	0.086	0.174
100	0.096	0.148

Wykres przedstawiający T_E dla różnych wartości d



Metoda poszukiwania cyklu Eulera oparta jest na algorytmie DFS (przeszukiwanie w głąb), z tą różnicą że przegląda krawędzie zamiast wierzchołków. Do przedstawienia grafu użyto macierzy sąsiedztwa. Uznaliśmy, że sprawi ona nam najmniej problemów związanych z implementacją. Niestety, wybrana przez nas struktura nie jest najwydajniejsza do tego typu zadania, szybsza byłaby lista następników. Macierz sąsiedztwa posiada jednak przewagę w krótkim czasie i łatwości usuwania krawędzi.

Metoda poszukująca cyklu Eulera przechodzi przez każdą krawędź co najmniej raz. Dlatego dla listy następników złożoność wynosi $O(m)$. Do implementacji z macierzą sąsiedztwa trzeba dodać czas wyszukiwania następnika dla wszystkich wierzchołków, stąd złożoność obliczeniowa dla tej implementacji wynosi $O(n^2+m)$.

Algorytm rozpoczyna działanie od sprawdzenia, czy wszystkie wierzchołki są parzystego stopnia, w wypadku gdy nie są, kończy swoje działanie i zwraca odpowiednią wartość (w tym wypadku -1). Na tym etapie nie jest sprawdzana spójność grafu, ten warunek jest uwzględniany podczas jego tworzenia. Jeżeli struktura przejdzie ten test, wybierany jest wierzchołek startowy (w tej implementacji wierzchołek o najniższym możliwym indeksie). Jest on kładziony na stos. Następnie wybierany jest wierzchołek o jak najmniejszym numerze, do którego istnieje połączenie z obecnej lokalizacji, przechodzi do niego, dodaje go na stos i usuwa połączenie między nimi. Algorytm powtarza to tak długo, aż nie znajdzie się w wierzchołku, z którego nie ma wyjścia. W ten czas zaczyna zdejmować ze stosu, wykonuje tę czynność tak długo, aż nie wróci do wierzchołka, w którym istnieje niewykorzystane połączenie. Jeżeli je znajdzie schemat postępowania powtarza się. Algorytm kończy się, gdy nie zostanie już żadna krawędź. Kolejność zdejmowania ze stosu jest tu bardzo istotna, to właśnie ona tworzy cykl Eulera.

Warunek konieczny i dostateczny istnienia cyklu Eulera w grafie:

- graf jest spójny,
- dla grafu nieskierowanego, wszystkie wierzchołki są stopnia parzystego,
- dla grafu skierowanego, taka sama liczba krawędzi wchodzących i wychodzących dla każdego wierzchołka.

W testowanych grafach istniał cykl Eulera ponieważ zostały one wygenerowane odpowiednią metodą. Opierała się ona na tworzeniu klik o rozmiarze 3, po stworzeniu pierwszej wybierany był losowy należący do grafu wierzchołek, oraz losowano dwa nienależące do grafu, z tych trzech wierzchołków do grafu dołączana była nowa klika. Dołączanie nowych klik trwało aż do osiągnięcia pożądanej gęstości.

Wraz z zwiększaniem liczby wierzchołków rośnie czas działania algorytmu poszukującego cyklu Eulera, tak samo dla grafów o tych samych rozmiarach ale o większej gęstości czas jest dłuższy, co oznacza że algorytm zachowuje się naturalnie.

4 Czasy poszukiwania pojedynczego i wszystkich cykli Hamiltona dla różnych wartości d

Tabela prezentująca t_{H1} i t_{HA} dla różnych wartości d

Liczba wierzchołków	$d = 0.2$		$d = 0.6$	
	t_{H1} $[ms]$	t_{HA} $[ms]$	t_{H1} $[ms]$	t_{HA} $[ms]$
5	0.001	0.002	0.003	0.004
6	0.002	0.002	0.005	0.021
7	0.001	0.001	0.031	0.037
8	0.002	0.003	0.033	0.072
9	0.001	0.001	0.006	0.396
10	0.008	0.010	0.130	3.945
11	0.000	0.001	0.005	25.734
12	0.009	0.010	0.048	232.968
13	0.012	0.013	0.009	1063.510
14	0.039	0.043	0.015	5300.210
15	0.062	0.069	0.010	*przerwano*
20	18.071	19.434	0.035	*przerwano*
25	1.196	12220.800	11.446	*przerwano*

Wykres przedstawiający t_{H1} dla różnych wartości d

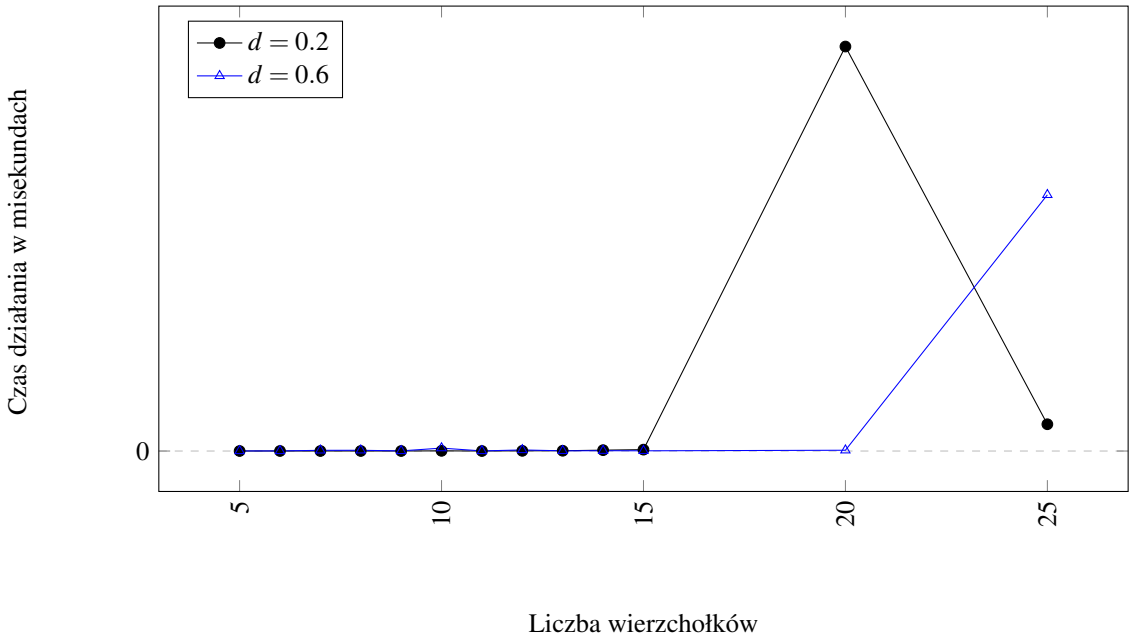


Tabela prezentująca t_{HA} dla różnych wartości d

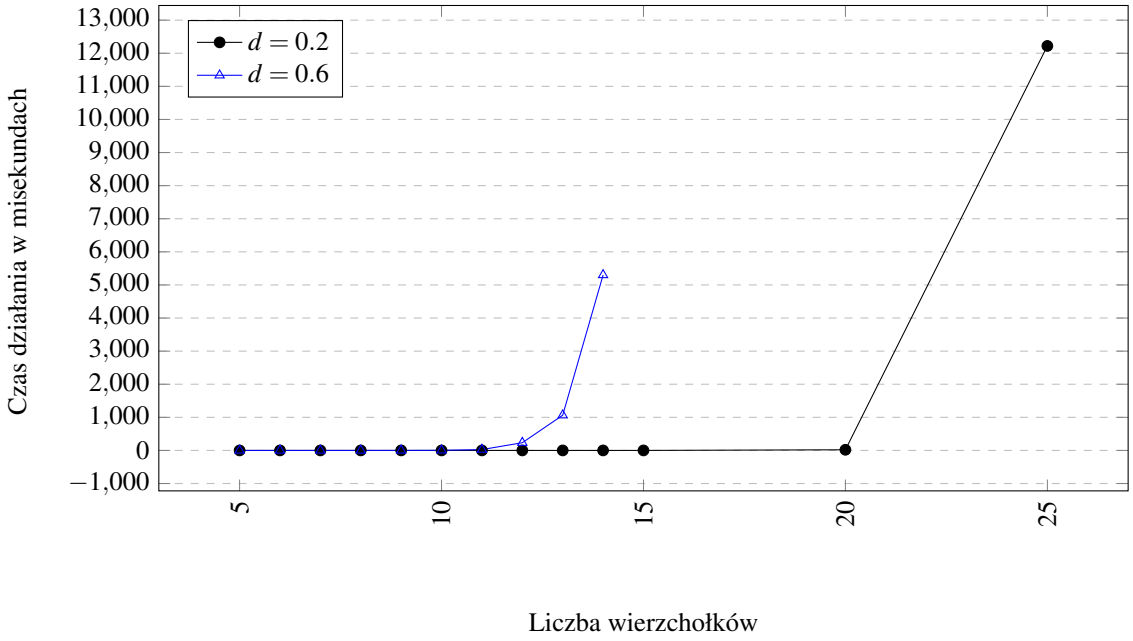


Tabela prezentująca liczbę cykli Hamiltona dla różnych wartości d

Liczba wierzchołków	Liczba cykli Hamiltona dla $d = 0.2$	Liczba cykli Hamiltona dla $d = 0.6$
5	0	0
6	0	2
7	0	0
8	0	2
9	0	4
10	0	128
11	0	3372
12	0	11964
13	0	66680
14	0	346018
15	0	*przerwano*
20	0	*przerwano*
25	0	*przerwano*

Do znajdowania

Spis treści

1 Opis implementacji **1**

2 Czasy działania algorytmów **1**

3 Czasy poszukiwania cyklu Eulera dla różnych wartości d **3**

4 Czasy poszukiwania pojedynczego i wszystkich cykli Hamiltona dla różnych wartości d **5**