Sprawozdanie nr 4
13.05.2022

ALGORYTMY I STRUKTURY DANYCH semestr letni, rok akademicki 2021/2022 piątek 9:45-11:15

1. Andrei Kartavik, Indeks: 153925
2. Ivan Kaliadzich, Indeks: 153936

Wstęp

Celem tego sprawozdania jest zapoznanie się z klasami złożoności algorytmów na przykładzie poszukiwania cyklu Eulera i cykli Hamiltona, oraz zapoznanie się z algorytmem z powracaniem, który był wykorzystany dla poszukiwania cykli Hamiltona w grafie.

Przy porównaniu zależności czasu poszukiwania cyklu Eulera (E), pojedynczego cyklu Hamiltona (H1) i wszystkich cykli Hamiltona (HA) od *n* liczby wierzchołków, dla *d* = 0.6, pomiary były wykonywane dla *n* z zakresu [10, 16], z krokiem co 1, ale gdy czas poszukiwania wszystkich cykli Hamiltona stał się zbyt duży, przerwaliśmy go testowanie i kontynuowaliśmy testowanie poszukiwania cyklu Eulera, oraz poszukiwania pojedynczego cyklu Hamiltona dla *n* z zakresu [100, 550] z krokiem co 50 (10 punków pomiarowych).
Przy wyodrębnionym testowaniu zależności czasu wyszukiwania cyklu Eulera od *n* wierzchołków i gęstości grafu *d* = 0.2 i *d* = 0.6; pomiary były wykonywane dla *n* z zakresu [50, 500], z krokiem co 50.
Podczas testowania zależności czasu wyszukiwania pojedynczego cyklu Hamiltona od *n* wierzchołków i gęstości grafu d = 0.2 i d = 0.6; pomiary były wykonywane dla *n* z zakresu [20, 65], z krokiem co 5.
Przy badaniu zależności czasu wyszukiwania wszystkich cykli Hamiltona od *n* wierzchołków i gęstości grafu d = 0.2 i d = 0.6; oraz badaniu liczby cykli Hamiltona w grafie, pomiary były wykonywane dla *n* z zakresu [10, 15], z krokiem co 1.
Wszystkie pomiary były wykonywane dla *m* liczby krawędzi, gdzie
m = ½dn(n-1).

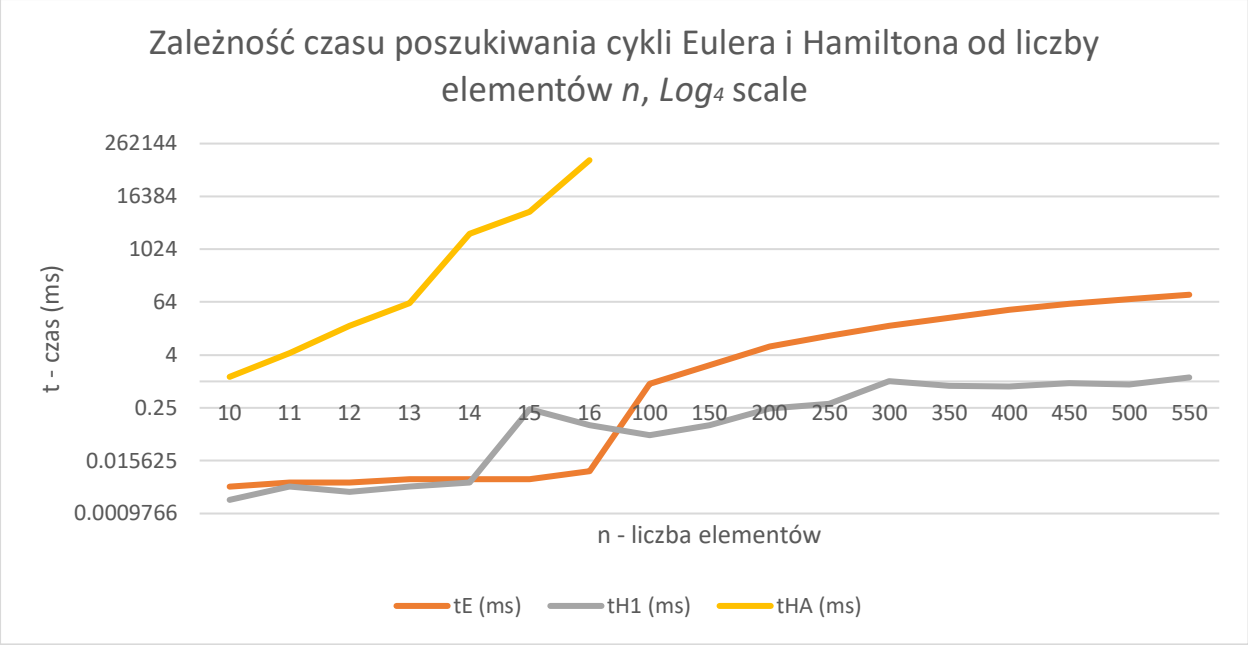
Generowanie grafów

Ponieważ warunkiem koniecznym i wystarczającym na to by spójny graf nieskierowany miał cykl Eulera jest parzystość stopni wszystkich wierzchołków, podjęliśmy następną strategię generowania potrzebnych nam grafów: Najpierw generujemy losowy graf nieskierowany, zatem przez pętle sprawdzamy parzystość stopnia każdego wierzchołka i w przypadku nieparzystości usuwamy, lub dodajemy jedną krawędź danego wierzchołka. Po przechodzeniu przez cały graf sprawdzamy go na spójność metodą przechodzenia w głąb DFS i jeżeli graf okazał się niespójnym, zaczynamy proces generowania grafu od nowa, i powtarzamy tą procedurę aż do momentu uzyskania potrzebnego nam grafu.

Porównanie czasu poszukiwania cyklu Eulera (E), pojedynczego cyklu Hamiltona (H1) i wszystkich cykli Hamiltona (HA) w grafie nieskierowanym

d = 0,6			
n	t_E (ms)	t_{H1} (ms)	t_{HA} (ms)
10	0,004	0,002	1,27
11	0,005	0,004	4,327
12	0,005	0,003	18,218
13	0,006	0,004	59,838
14	0,006	0,005	2270,14
15	0,006	0,23	7349,637
16	0,009	0,1	109564,4
100	0,89	0,06	
150	2,371	0,1	
200	6,296	0,241	
250	10,796	0,308	
300	18,437	1,019	
350	28,436	0,792	
400	42,092	0,764	
450	58,863	0,921	
500	75,031	0,841	
550	94,345	1,229	

Tablica 1: Zależność czasu poszukiwania cykli Eulera i Hamiltona od n, dla gęstości d = 0,6. (ms)

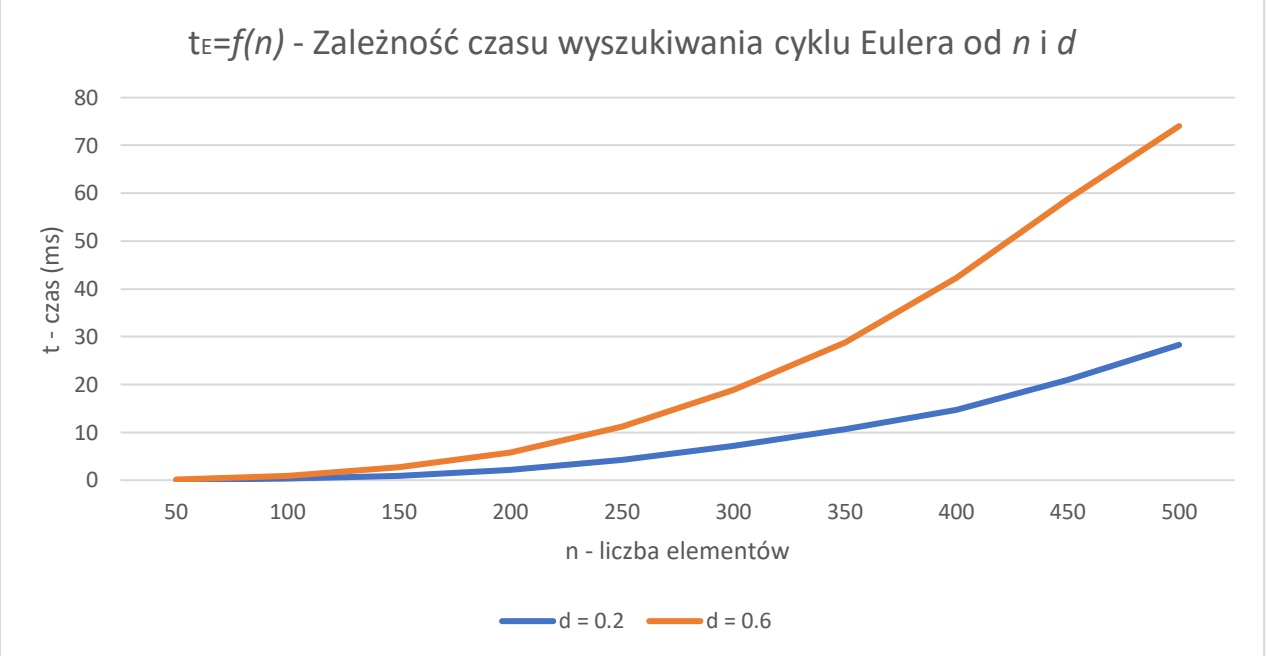


Problem znajdowania cyklu Eulera należy do klasy problemów łatwych (P – klasy problemów), czyli takich, które są rozwiązywalne przez Deterministyczną maszynę Turinga w wielomianowym czasie. Klasyfikacja ta została dokonana na podstawie tego, że istnieje algorytm, który znajduje cykl Eulera w czasie rzędu wielomianowego. Złożoność obliczeniowa tego algorytmu wynosi O(m) dla odpowiedniej reprezentacji grafu, a właśnie dla listy sąsiedztwa.
Problem znajdowania cyklu Hamiltona należy do klasy problemów trudnych (NP-zupełnych) czyli takich, które nie są rozwiązywalne przez Deterministyczną maszynę Turinga w czasie wielomianowym. To znaczy, że nie istnieje algorytmu poszukującego cyklu Hamiltona w czasie wielomianowym, a z tego wynika, że złożoność obliczeniowa poszukiwania pojedynczego, jak i wszystkich cykli Hamiltona jest wykładnicza, czyli O(n!). Dlatego, nawet dla 16 wierzchołków, jak możemy zobaczyć na powyższym wykresie, czas poszukiwania wszystkich cykli Hamiltona wynosi aż prawie 2 minuty.

Zależność czasu poszukiwania cyklu Eulera od *n* wierzchołków i różnych wartości *d*

n - liczba wierzchołków	d - gęstość	t_E=*f(n)* - Zależność czasu wyszukiwania cyklu Eulera od *n* i *d* (ms)
50	0,2	0,045
	0,6	0,136
100	0,2	0,318
	0,6	0,898
150	0,2	0,925
	0,6	2,69
200	0,2	2,195
	0,6	5,798
250	0,2	4,223
	0,6	11,161
300	0,2	7,198
	0,6	18,887
350	0,2	10,668
	0,6	28,83
400	0,2	14,739
	0,6	42,254
450	0,2	20,962
	0,6	58,725
500	0,2	28,303
	0,6	74,046

Tablica 2: Zależność czasu poszukiwania cyklu Eulera od n i d. (ms)

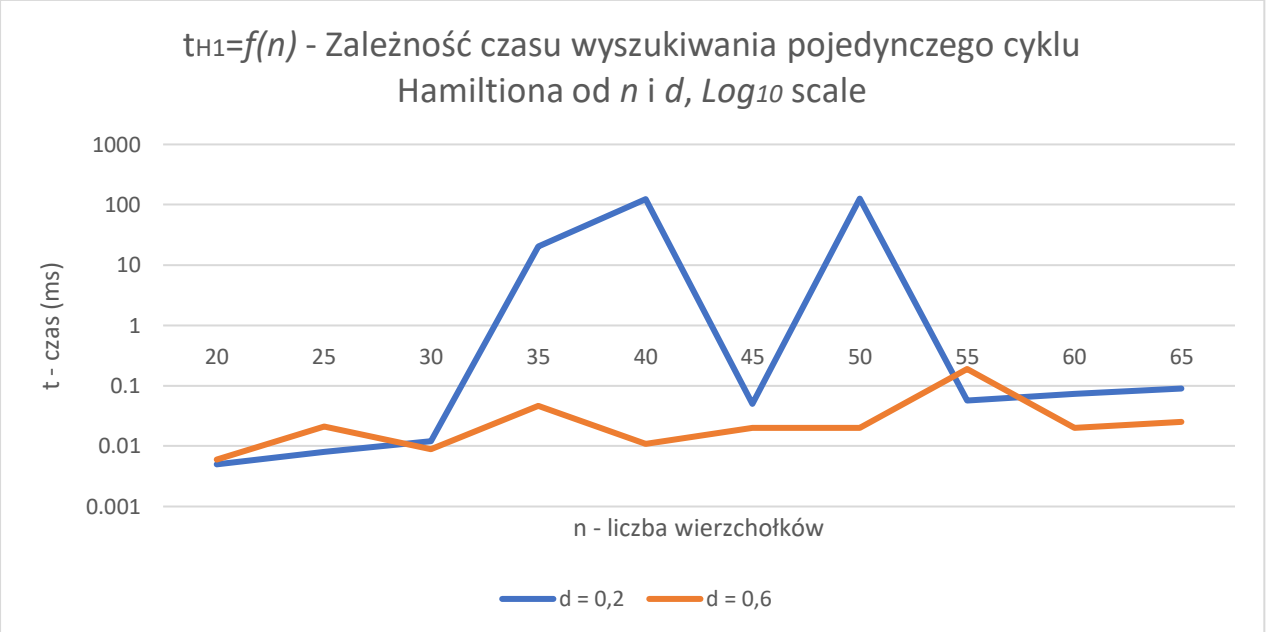


Cykl Eulera to taki cykl w grafie, który przechodzi przez każdą jego krawędź dokładnie jeden raz. Nasz algorytm znajdowania cyklu Eulera w grafie jest oparty na metodę przeszukiwania grafu w głąb (DFS), z tą różnicą, że analizowane są krawędzie zamiast wierzchołków.
Jak i w DFS, zaczynamy od dowolnego wierzchołka, kładziemy go na stos i przechodzimy do następnego wierzchołka, ale teraz, z powodu tego, że musimy przejść przez każdą krawędź dokładnie jeden raz, po przechodzeniu do następnego wierzchołka usuwamy tą krawędź, która łączy te wierzchołki. W przypadku, gdy przejdziemy do wierzchołka, który już nie ma krawędzi łączących go z innymi wierzchołkami, musimy zdjąć ten wierzchołek ze stosu i zapisać go w wynik, oraz cofnąć się do poprzedniego wierzchołku i kontynuować tą procedurę aż do momentu, kiedy stos okaże się pusty. Otrzymana kolejność wierzchołków które były zapisywane do wyniku i jest naszym cyklem Eulera.
Jak już było powiedziano, przy odpowiedniej reprezentacji grafu, czyli listy sąsiedztwa, złożoność znajdowania cyklu Eulera wynosi O(*m*), gdzie *m* – liczba krawędzi w grafie, i wynika to z tego, że musimy przejść przez każdą krawędź dokładnie raz – O(*m*), a złożoność poszukania następnika kolejnego odwiedzonego wierzchołka w grafie dla listy sąsiedztwa jest natychmiastowe – O(1), co razem nam daje O(*m*). Ale z powodu tego, że w algorytmie, który jest oparty na DFS, przy przechodzeniu do następnego wierzchołka musimy każdy raz usuwać krawędź, co dla listy sąsiedztwa wynosi O(*n*), bo musimy przejść przez list, żeby odnaleźć wierzchołek, który jest połączony daną krawędzią, co w wyniku nam daje złożoność czasową tego algorytmu dla listy sąsiedztwa - O(*n*m*), gdzie *n* – liczba wierzchołków, a *m* – liczba krawędzi.
I właśnie z tego powodu, do implementacji wybraliśmy inną reprezentację grafu, mianowicie macierz sąsiedztwa, dla której złożoność usuwania krawędzi w grafie jest natychmiastowe, czyli O(1)… Ale ma ona swoją wadę, bo złożoność czasowa poszukania następnika wierzchołka dla macierzy sąsiedztwa wynosi O(n). I w wyniku, musimy przejść przez każdy wierzchołek – O(n) i dla każdego wierzchołka znajdujemy go następnika - O(n), co razem nam daje $O(n^2)$, oraz musimy przejść przez każdą krawędź – O(m), więc złożoność takiego algorytmu dla macierzy sąsiedztwa wynosi $O(n^2 + m)$.
Możemy dojść do wniosku, że reprezentacja grafu ma wpływ na złożoność obliczeniową tej metody. Oraz mogliśmy zobaczyć na wykresie i w tablice 2, że gęstość grafu miała wpływ na czas wykonania algorytmu. Wynika to z tego, że im większe jest *d*, tym większa jest liczba krawędzi w grafie ($m = ½dn(n-1).$), więc musimy przejść przez większą liczbę krawędzi, żeby odnaleźć cykl Eulera, co i daje nam tą różnicę w czasie.

Poszukiwanie cykli Hamiltona

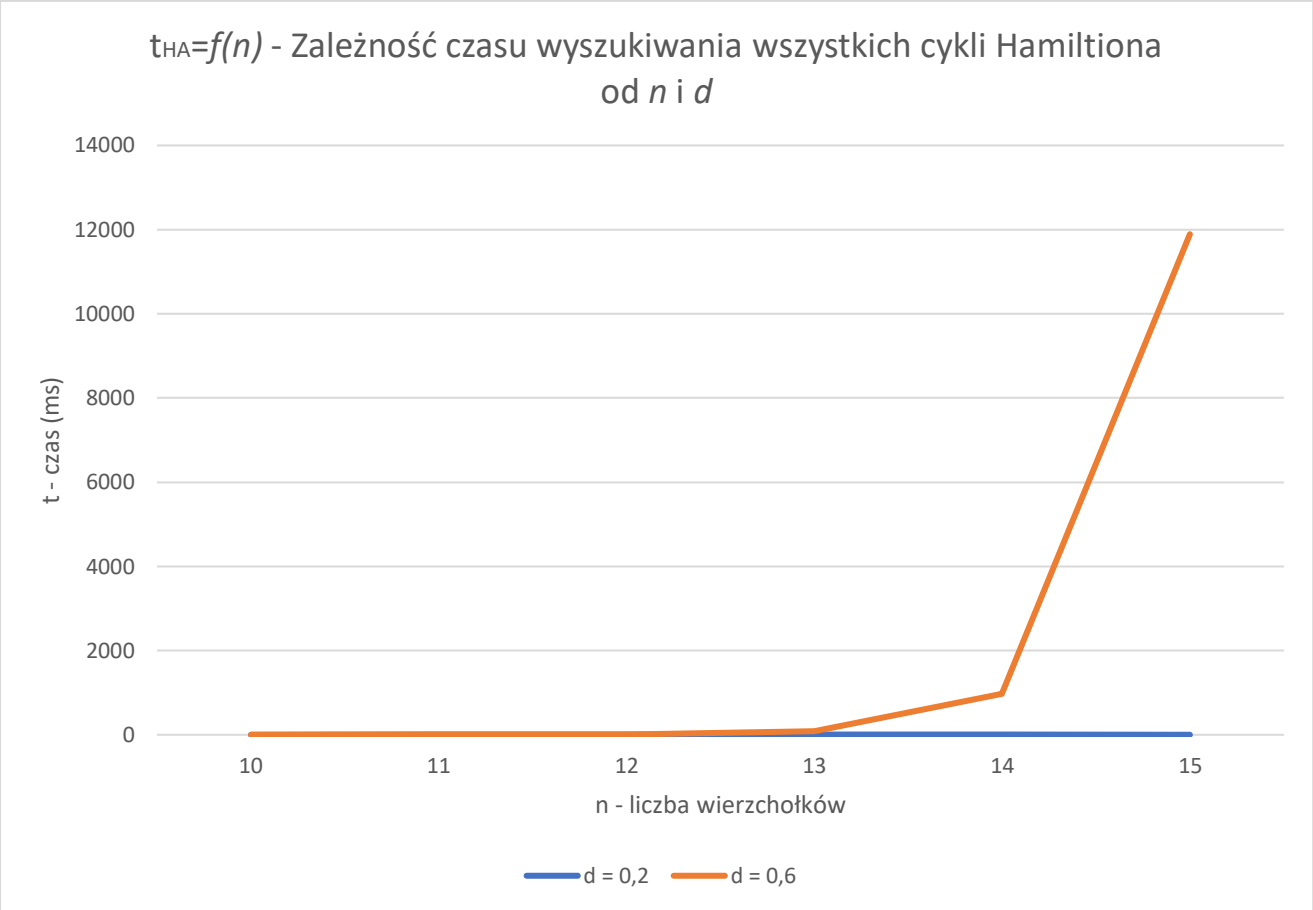
n	d	$t_{H1}=f(n)$ (ms)
20	0,2	0,005
	0,6	0,006
25	0,2	0,008
	0,6	0,021
30	0,2	0,012
	0,6	0,009
35	0,2	20,271
	0,6	0,046
40	0,2	125,279
	0,6	0,011
45	0,2	0,05
	0,6	0,02
50	0,2	126,626
	0,6	0,02
55	0,2	0,057
	0,6	0,19
60	0,2	0,073
	0,6	0,02
65	0,2	0,091
	0,6	0,025

Tablica 3: Zależność czasu wyszukiwania pojedynczego cyklu Hamiltona od n, dla różnych d. (ms)



n	d	$t_{HA}=f(n)$
10	0,2	0,006
	0,6	0,797
11	0,2	0,016
	0,6	2,08
12	0,2	0,042
	0,6	13,043
13	0,2	0,046
	0,6	86,103
14	0,2	0,306
	0,6	975,42
15	0,2	0,43
	0,6	11891,727

Tablica 4: Zależność czasu wyszukiwania wszystkich cykli Hamiltona od n, dla różnych d. (ms)



n - liczba wierzchołków	d - gęstość	C_H - liczba cykli Hamiltona w zależności od *n* i *d*
10	0,2	0
	0,6	672
11	0,2	0
	0,6	1256
12	0,2	1
	0,6	6011
13	0,2	0
	0,6	24437
14	0,2	3
	0,6	686509
15	0,2	1
	0,6	14236860

Tablica 5: liczba cykli Hamiltona w zależności do n i d

Cykl Hamiltona to taki cykl w grafie, w którym każdy wierzchołek grafu odwiedzany jest dokładnie raz (oprócz pierwszego wierzchołka). Jak już ustaliliśmy, poszukiwanie cyklu Hamiltona w grafie odnosi się do NP-zupełnych problemów, a to znaczy, że nie istnieje takiego algorytmu, który znalazłby rozwiązanie w czasie wielomianowym.
Dla rozwiązania tego problemu posłużyliśmy się algorytmem z powracaniem, ideą którego jest generowanie wszystkich przypadków z odrzucaniem tych, które nie spełniają wymagań. Do rozwiązania naszego problemu ten algorytm wygląda tak: Zaczynamy od dowolnego wierzchołka i zaznaczamy, że on został odwiedzony, oraz dodajemy go do naszego wyniku. Zatem przechodzimy do następnego nieodwiedzonego wierzchołka i ponownie zaznaczamy, że wierzchołek został odwiedzony, oraz dodajemy go do naszego wyniku. Gdy dojdziemy do takiego wierzchołka, dla którego wszystkie sąsiednie wierzchołki już zostały odwiedzone wcześniej, sprawdzamy, czy nasz wynik spełnia warunek cyklu Hamiltona, jeżeli tak, zaznaczamy, że to jest cykl Hamiltona, jeżeli nie – skreślamy ten wierzchołek z wyniku, zaznaczmy go jako nieodwiedzony, cofamy się do poprzedniego wierzchołka który został odwiedzony przed tym, na którym skończyliśmy i przechodzimy do innego nieodwiedzonego wierzchołka, kontynuując wykonanie naszego algorytmu. Jeżeli naszym celem jest znalezienie pojedynczego cyklu Hamiltona, to skończymy działanie algorytmu przy znalezieniu pierwszego takiego cyklu, a jeżeli naszym celem jest znalezienie wszystkich cykli Hamiltona, to będziemy kontynuować ten proces aż nie sprawdzimy wszystkie możliwe przypadki.

Jest to lepsze podejście niż banalny „brute force”, czyli generowanie wszystkich możliwych przypadków (wszystkich permutacji połączonych wierzchołków w grafie) i sprawdzania każdego z nich, bo przy wykonywaniu naszego algorytmu, jeżeli znaleziony wynik nie odpowiada naszemu warunku, nie musimy zaczynać generowanie permutacji od nowa, a możemy po prostu cofnąć się do poprzedniego wierzchołka i sprawdzić inny możliwy przypadek. Ale nadal, w najgorszym przypadku musimy sprawdzić wszystkie możliwe przypadki w celu znalezienia cyklu Hamiltona, co wynosi O(n!). Dlatego, niezależnie od reprezentacji grafu, będziemy mieli złożoność O(n!). Wybraliśmy macierz sąsiedztwa, bo jest ona prosta do implementacji i ma liniową złożoność poszukiwania następnika kolejnego wierzchołka.

Dla wyszukiwania pojedynczego i wszystkich cykli algorytm jest taki sam, jednak w przypadku poszukiwania pojedynczego cyklu, po jego znalezieniu algorytm kończy pracę, a w przypadku poszukiwania wszystkich cykli, algorytm sprawdzi wszystkie możliwe przypadki. Poszukiwanie pojedynczego cyklu nie jest ”stabilne”, ponieważ możemy znaleźć cykl Hamiltona na początku poszukiwań, lub możemy znaleźć cykl Hamiltona w końcu poszukiwań, co oczywiście ma wpływ na czas działania algorytmu i dlatego, jak możemy zobaczyć na wykresie, i pojawiają się te skoki . W przypadku poszukiwania wszystkich cykli Hamiltona czas działania algorytmu jest bardziej „stabilny”, bo musimy sprawdzić każdy możliwy przypadek w grafie i dlatego, im więcej mamy wierzchołków, oraz im większa jest gęstość (a to znaczy, że większa liczba krawędzi), tym więcej przypadków musimy sprawdzić, stąd i wynika ta różnica w czasie, którą możemy zobaczyć na wykresie i w tablice *4*.

Jak już było powiedziano, gęstość grafu ma wpływ na czas wyszukiwania wszystkich cykli Hamiltona, ale zarówno ma ona wpływ na czas wyszukiwania pojedynczego cyklu Hamiltona… Ponieważ mamy większą gęstość, to mamy i większą liczbę krawędzi, a to znaczy, że mamy większą liczbę cykli Hamiltona w grafie (co możemy zobaczyć w tablice *5*), bo z każdego wierzchołka będziemy mieli więcej możliwości przejścia do następnego, i w związku z tym, mamy większą szansę znaleźć cykl Hamiltona na początku poszukiwań. Stąd i wynika, że dla mniejszej gęstości czas wyszukiwania pojedynczego cyklu Hamiltona średnio jest większy, niż czas poszukiwania dla większej gęstości.

Podsumowanie

Przy rozwiązaniu problemu znajdowania cyklu Eulera w grafie, musieliśmy wygenerować takie spójne grafy nieskierowane, które spełniałyby warunek konieczny i dostateczny istnienia cyklu Eulera, a właśnie parzystość stopni każdego wierzchołka. Wybraliśmy nie samą efektywną strategię ze względu na złożoność czasową i pamięciową, ale mieliśmy pewność, że otrzymano potrzebnego nam grafu. Dokonaliśmy klasyfikacji tego problemu, a właśnie odnieśliśmy go do klasy problemów łatwych (P), bo istnieje taki algorytm, który jest w stanie rozwiązać ten problem w czasie wielomianowym. Zarówno mogliśmy zobaczyć wpływ gęstości grafu na czas poszukiwania cyklu Eulera, oraz upewniliśmy się, że złożoność czasowa algorytmu znajdowania cyklu Eulera oparty na DFS zależy od reprezentacji grafu.

Przy rozwiązaniu problemu znajdowania pojedynczego i wszystkich cykli Hamiltona w grafie stwierdziliśmy, że ten problem odnosi się do problemów trudnych (NP-zupełnych), ponieważ nie istnieje takiego algorytmu, który rozwiązałby ten problem w czasie wielomianowym. Ale mieliśmy do czynienia z dobrą strategią rozwiązywania NP-zupełnych problemów, mianowicie stosowanie algorytmu z powracaniem, który jest lepszym podejściem, niż generowanie i sprawdzanie wszystkich możliwych przypadków trudnego problemu. Zaobserwowaliśmy wpływ gęstości grafu na liczbę cykli Hamiltona, oraz na czas poszukiwania cykli Hamiltona i upewniliśmy się, że reprezentacja grafu nie ma wpływu na złożoność obliczeniową takiego problemu.