Sprawozdanie

algorytmy z powracaniem

Jan Cieśla 155945

Dane wejściowe

Grafy hamiltonowskie tworzone są poprzez utworzenie macierzy sąsiedztwa, aby zapewnić obecność cyklu hamiltona łączymy w cykl losową permutację wierzchołków, po tej operacji stopnie wierzchołków są parzyste a graf jest już spójny. Następnie wypełniamy graf do pożądanego współczynnika wypełnienia losowymi 3-węzłowymi cyklami co zapewnia zachowanie parzystych stopni wierzchołków, powstaje graf również eulerowski. Finalnie przekształcamy macierz do listy następników.

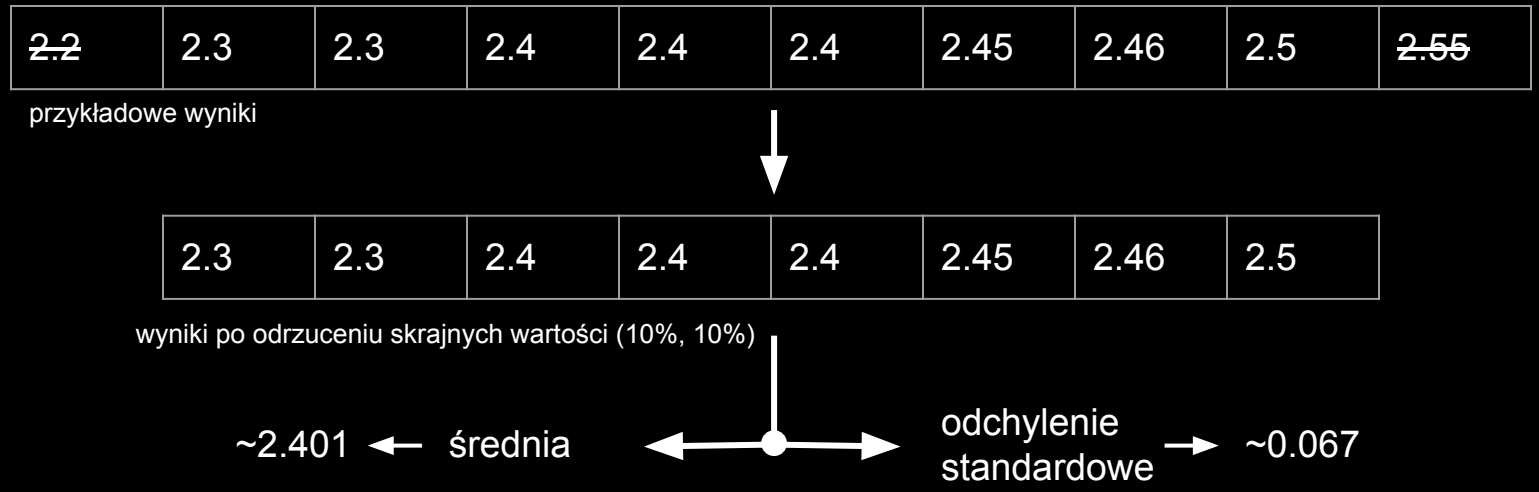
grafy nie-hamiltonowskie tworzone są poprzez utworzenie grafu hamiltonowskiego identycznego jak w poprzednim akapicie i wyizolowanie jednego wierzchołka.

Do testów algorytmu wyszukiwania cyklu hamiltona przyjęliśmy n'y zaczynające się od 10 z krokiem 5 do 55 włącznie i współczynniki wypełnienia 0.3 i 0.7 dla grafów hamiltonowskich oraz n'y zaczynające się od 10 z krokiem 1 do 17 włącznie przy współczynniku równym 0.5 dla grafów nie-hamiltonowskich.

Do testów algorytmu wyszukiwania cyklu eulera przyjęliśmy n'y zaczynające się od 100 z krokiem 100 do 1000 włącznie i współczynniki wypełnienia 0.3 i 0.7.

Testy algorytmów

W ramach testów algorytmów pięciokrotnie zmierzyliśmy czas ich działania dla każdej kombinacji wymaganej algorytm - współczynnik wypełnienia. Odrzuciliśmy skrajne wyniki (20% najwyższych i 20% najniższych), a z pozostałych wyników obliczyliśmy średnią oraz odchylenie standardowe - te dane są przedstawione na wykresach. W opisach algorytmów n będzie oznaczało ilość wierzchołków, a m ilość krawędzi. Czas mierzony jest za pomocą modułu time na platformie linux, dokładność pomiaru to +- 1us.



Reprezentacja grafu - Lista następników

Lista następników to struktura danych, która zawiera informacje o bezpośrednich sąsiadach każdego wierzchołka w grafie.

Złożoność pamięciowa listy następników zależy od liczby krawędzi w grafie oraz liczby wierzchołków. Jeśli graf ma n wierzchołków i m krawędzi, to pamięć potrzebna do przechowywania listy następników wynosi O(n + m).

Znalezienie pojedynczej krawędzi wymaga sprawdzenia całej listy następników określonego wierzchołka, z którego może wychodzić n-1 krawędzi, dlatego złożoność wynosi O(n-1) => O(n).

Znalezienie wszystkich następników danego wierzchołka również wymaga przejrzenia całej listy dotyczącej odpowiadającego wierzchołka, stąd złożoność O(n-1) => O(n).

Wybór tej reprezentacji został podyktowany przez to, że algorytmy znajdowania cyklu Eulera i cyklu Hamiltona opierają się na znajdowaniu następników przetwarzanego węzła, a lista następników zapewnia w tym przypadku najlepszą możliwą złożoność obliczeniową. Ważnym kryterium była również prostota implementacji.

```
0: [1, 2, 3, 4, 5, 6, 7, 8, 9]
1: [3, 5, 6]
2: [6, 9]
3: [5, 6, 9]
4: [5]
5: [8]
6: [7]
7: [9]
8: []
9: []
```

Uwagi i obserwacje

Znajdowanie cyklu hamiltona:

- ilość obecnych cykli hamiltona w grafie jest proporcjonalna do jego rozmiaru oraz współczynnika wypełniania
- Algorytm zatrzymuje się po znalezieniu pierwszego cyklu, zatem dla grafów nie-hamiltonowskich wymagane będzie przejrzenie wszystkich możliwych ze względu na krawędzie i punkt startowy permutacji węzłów.

  Gdy graf jest hamiltonowski czas działania uzależniony jest od tego jak szybko znajdziemy cykl, czego charakter jest losowy ale zależny od współczynnika wypełnienia grafu. (Wykres jest poszarpany właśnie z tego powodu)

- Duży współczynnik wypełnienia (0.7) sprawia, że algorytm jest w stanie znaleźć cykl w większym grafie, oraz znacząco przyspiesza średni czas znalezienia cyklu w stosunku do współczynnika małego (0.3).

Uwagi i obserwacje

Znajdowanie cyklu eulera:

- Algorytm zaimplementowany przez nas (wersja rekurencyjna) działa wydajnie, lecz w przypadku większych grafów będzie on przekraczał systemowy limit rekurencji oraz pamięć stosu.
- Algorytm działa szybciej gdy wypełnienie grafu jest mniejsze, jest to spowodowane mniejszą ilością krawędzi do przetworzenia w rzadszym grafie.

Znajdowanie cyklu eulera

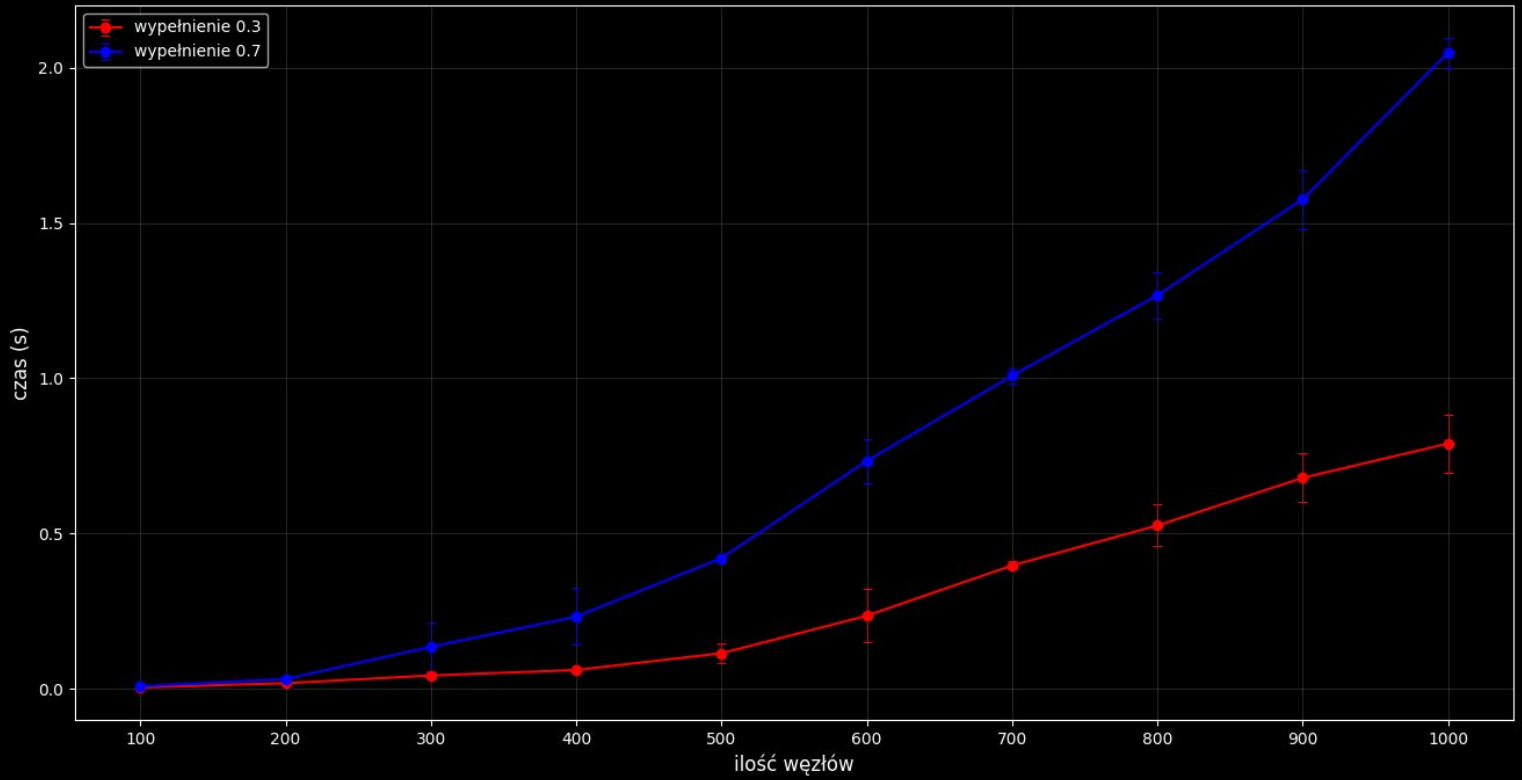
Zanim rozpocznie się faktyczny proces szukania algorytm sprawdza parzystość stopni wierzchołków aby sprawdzić czy cykl eulera jest obecny w grafie, jeśli nie nie przechodzimy do dalszej części. Proces ten nie wymaga dodatkowej pamięci a jego złożoność obliczeniowa to O(m).

Właściwy algorytm opiera się na wykorzystaniu stosu i rekurencyjnego przejścia post order po grafie. Zaczynamy od wybranego wierzchołka (jego wybór nie ma znaczenia ze względu na to, że szukamy cyklu [tylko w przypadku grafów spójnych w przeciwnym wypadku musimy wybrać wierzchołek o dodatnim stopniu]) i dla każdego następnika jeżeli krawędź nie została wykorzystana oznaczamy ją jako użytą i wywołujemy algorytm rekurencyjnie. Po przetworzeniu wszystkich sąsiadów odkładamy wierzchołek na stos wynikowy. Po zakończeniu wszystkich wywołań cykl jest przechowywany na stosie wynikowym.

Krawędzie oznaczamy jako zużyte zamiast usuwać je z grafu w celu poprawy wydajności. Są one przechowywane jako zbiór ze stałym czasem dostępu. Wymaga to O(m) dodatkowej pamięci w zamian redukując złożoność obliczeniową usunięcia krawędzi z O(n) do O(1).

Procedura przechodzi przez każdą krawędź w grafie czas - O(m), wszystkie krawędzie zostaną oznaczone jako usunięte - czas O(m) dodatkowa pamięć - O(m), odłożenie wierzchołka na stos w każdym wywołaniu czas - O(m) dodatkowa pamięć - O(m). Sumarycznie Algorytm działa w czasie liniowo zależnym od ilości krawędzi i ze złożonością pamięciową O(m).

Znajdowanie cyklu Eulera



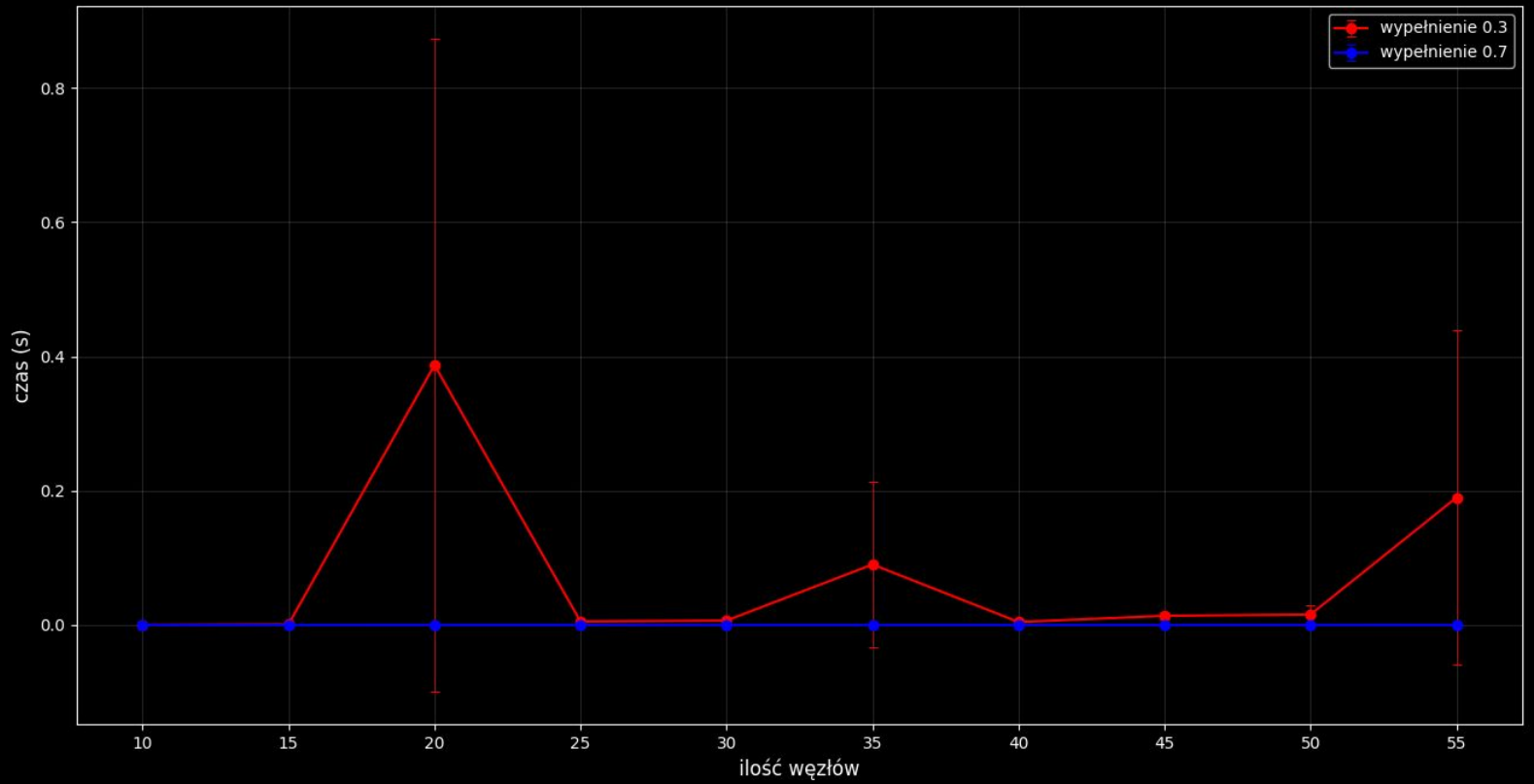
Znajdowanie cyklu hamiltona

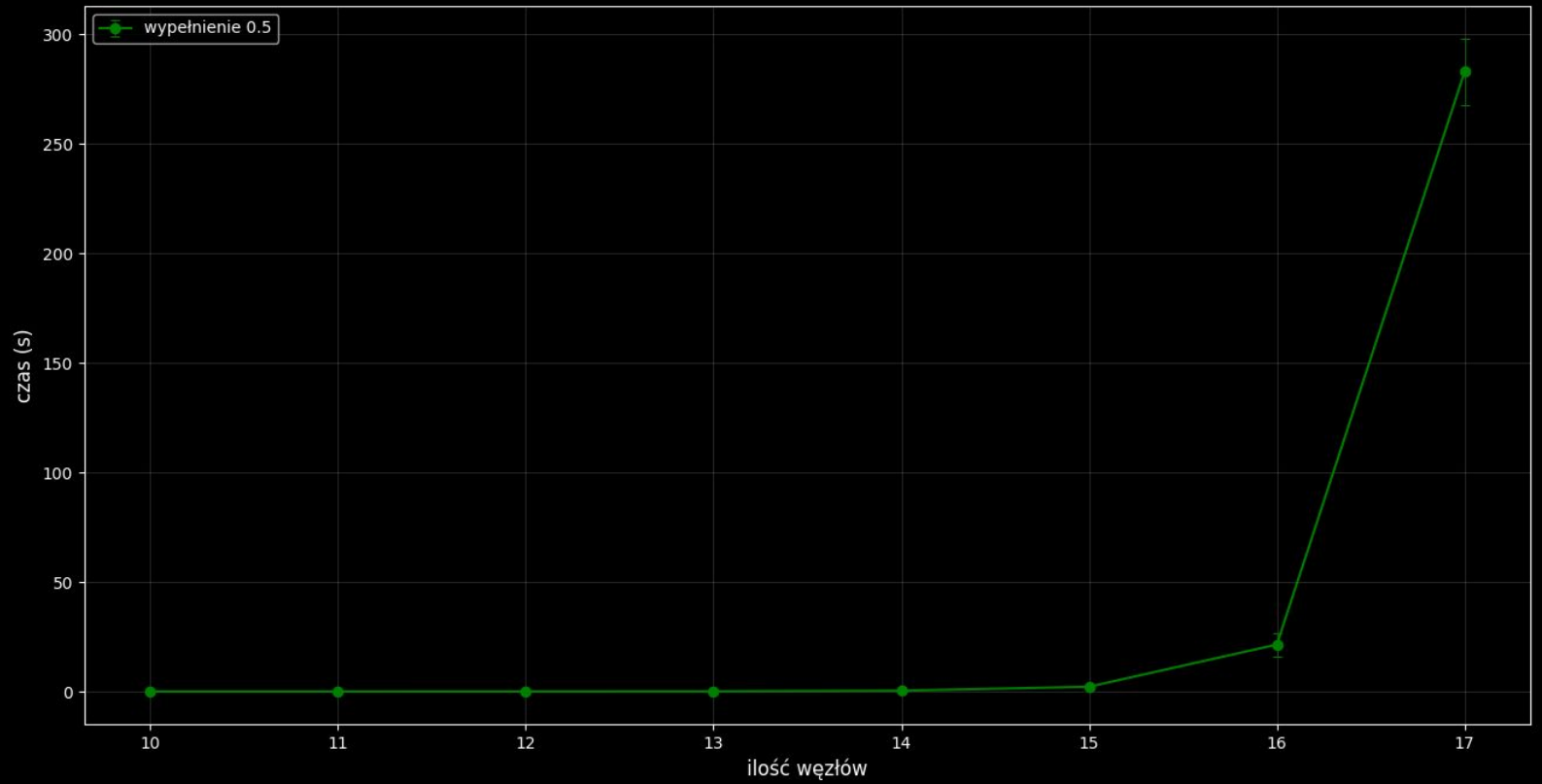
Algorytm poszukujący pojedynczego cyklu Hamiltona opiera się na wykorzystaniu algorytmu z powracaniem, główna idea tego algorytmu polega na rekurencyjnym przeszukiwaniu drzewa rozwiązań, przy czym w każdym kroku wybierany jest kolejny wierzchołek do przejścia, gdy osiągnięcie celu nie jest możliwe algorytm wycofuje się na wyższy poziom rekursji “zapominając” wierzchołki z niższych poziomów.

Złożoność obliczeniowa algorytmu poszukującego pojedynczego cyklu Hamiltona to O(n!), gdzie n to liczba wierzchołków. Złożoność ta wynika z faktu, że w najgorszym przypadku musimy sprawdzić każdą możliwą permutację wierzchołków. Z kolei algorytm poszukujący wszystkie cykle Hamiltona posiada złożoność czasową O(n*n!), ponieważ musimy znaleźć wszystkie permutacje wierzchołków oraz dla każdej z nich sprawdzić, czy tworzy ona cykl Hamiltona.

Różnica między tymi dwoma algorytmami polega na tym, że algorytm poszukujący pojedynczego cyklu Hamiltona kończy działanie po znalezieniu pierwszego cyklu, podczas gdy algorytm poszukujący wszystkich cykli Hamiltona kontynuuje działanie w celu znalezienia wszystkich możliwych cykli. Liczba wierzchołków oraz liczba krawędzi mają wpływ na czas działania algorytmów poszukujących cykli Hamiltona, ponieważ czym większa jest liczba wierzchołków, tym więcej możliwych permutacji należy sprawdzić.

Gęstość grafu ma wpływ na liczbę cykli Hamiltona w grafie oraz na czas działania obu algorytmów - im większa jest gęstość grafu, tym więcej możliwych cykli Hamiltona może istnieć, co wpływa na czas potrzebny do ich znalezienia.


Znajdowanie cyklu Hamiltona w grafie hamiltonowskim
wypełnienie 0.3
wypełnienie 0.7
czas (s)
0.0
0.2
0.4
0.6
0.8
ilość węzłów
10
15
20
25
30
35
40
45
50
55


Znajdowanie cyklu Hamiltona w grafie nie hamiltonowskim
wypełnienie 0.5
czas (s)
ilość węzłów
0
50
100
150
200
250
300
10
11
12
13
14
15
16
17