|  |  |
| --- | --- |
| L3 – środa 13:30 | |
| Aleksander Kamiński | 155840 |
| Olaf Hofman | 155974 |

Obraz zawierający tekst, zegar

Opis wygenerowany automatycznie

Zadanie 4 – Algorytmy z powracaniem

## Wprowadzenie

Celem sprawozdania jest analiza problemów poszukiwania cyklu Eulera oraz cyklu Hamiltona, klasyfikacja problemów, wyznaczenie algorytmów je rozwiązujących oraz ich złożoności obliczeniowej, zbadanie wpływu gęstości grafu na czas działania.

Wnioski przedstawiono na podstawie przeprowadzonych pomiarów czasu wyszukiwania jednego cyklu Eulera, jednego cyklu Hamiltona oraz znajdowania wszystkich cyklów Hamiltona w grafie. Wyniki doświadczenia przedstawiono w formie tabel oraz wykresów.

Aby zbadać czas obliczeń wykorzystano program napisany w języku Python 3.10. Platforma testowa składała się z komputera z procesorem Intel Core i7-9850H oraz 32 GB RAM.

Program generuje losowy spójny nieskierowany graf cykliczny z określoną liczbą wierzchołków oraz gęstością (prawdopodobieństwo istnienia krawędzi między każdymi dwoma wierzchołkami), reprezentowany przez macierz sąsiedztwa, mierzy czas wyszukiwania cyklu Eulera za pomocą algorytmu opartego o DFS oraz wyszukiwania wszystkich cykli Hamiltona algorytmem z powracaniem. Implementacja wykorzystuje kilka funkcji pomocniczych, w celu zwiększenia czytelności kodu. Pomiary badania były przetwarzane w pliku Excel.

Zaimportowano biblioteki *random* (generowanie liczb losowych), *time* (mierzenie czasu) oraz *copy* (tworzenie kopii struktury grafu).

# I Przebieg doświadczenia

Przeprowadzono test dla łącznie 16 punktów pomiarowych dla liczby n należącej do zakresów odpowiednio {7, 8, 9, …, 12} oraz {13, 15, 17, …, 31}, w których wygenerowano losowo graf złożony z n wierzchołków.

W każdym punkcie pomiarowym wygenerowana została macierz sąsiedztwa przez utworzenie statycznej tablicy dwuwymiarowej n na n. Nie jest to rozwiązanie optymalne czasowo, jeśli chodzi o wyszukiwanie cykli w celu ich znalezienia w dużych grafach, ale jest wystarczające dla pomiarów czasu, a zapewnia stosunkowo łatwą implementację i przejrzystość debugowania. Po utworzeniu wszystkie komórki i,j (i,j należą do zbioru [0, 1, 2, …, n]) zawierają wartość 0 (oznaczającą brak krawędzi). Następnie macierz jest trawersowana po wierszach i kolumnach i dla każdej zawierającej 0 komórki generowana jest losowa liczba należąca do przedziału [0, 1], która jest porównywana z wartością gęstości (odpowiednio 0,2 albo 0,6) i jeśli wygenerowana liczba jest mniejsza lub równa względem gęstości, to wartość komórek i,j oraz j,i jest zmieniana na 1 (oznaczające istnienie krawędzi). Aby upewnić się, że graf jest Eulerowski, trzeba sprawić, by wszystkie wierzchołki miały stopień parzysty. Wykorzystujemy właściwość grafu nieskierowanego taką, że suma stopni wierzchołków jest liczbą parzystą. Oznacza to, że jeśli istnieje jakikolwiek wierzchołek o stopniu nieparzystym, to istnieje co najmniej jeden inny taki wierzchołek (suma liczb nieparzystych jest liczbą parzystą). W związku z tym znajdowane są wszystkie wierzchołki o stopniu nieparzystym, a następnie ich lista jest przeglądana w poszukiwaniu na przemian pary, która ma wspólną krawędź, a następnie takiej, która jej nie ma. W pierwszym przypadku wspólna krawędź jest usuwana, a w drugim dopisywana. Dzięki temu, że odbywa się to naprzemiennie, zmiana w liczbie krawędzi, a zatem również w gęstości grafu jest minimalna.

W pierwszych sześciu punktach pomiarowych został zmierzony czas wykonania sprawdzenia czy w grafie istnieje cykl Eulera, sprawdzenia czy w grafie istnieje cykl Hamiltona oraz znalezienia i zliczenia wszystkich cykli Hamiltona w grafie. Dla punktu pomiarowego o wartości n = 13 czas wyszukiwania wszystkich cykli Hamiltona stał się zbyt duży, więc dla punktów pomiarowych od siódmego do szesnastego (n = 13, 15, 17, …, 31) włącznie różnica między kolejnymi wartościami n została zwiększona do 2 oraz były wykonywane tylko pomiary czasu wykonania sprawdzenia czy w grafie istnieje cykl Eulera oraz sprawdzenia czy w grafie istnieje cykl Hamiltona.

Sprawdzanie czy w grafie istnieje cykl Hamiltona i szukanie cyklu Hamiltona działają podobnie z tą różnicą, że sprawdzenie czy istnieje cykl Hamiltona zatrzymuje się po znalezieniu jednego cyklu i zwraca informację, że cykl istnieje, a szukanie wszystkich cykli Hamiltona w grafie znajduje wszystkie cykle i dopiero wtedy się zatrzymuje. Wyszukiwanie cyklu polega na rozpoczęciu od wierzchołka 0 (zawsze 0 dla spójności wyników) i dopisaniu go do cyklu. Na każdym etapie wykonywania sprawdzenia czy istnieje cykl Hamiltona oraz znajdowania wszystkich cykli Hamiltona część „bound” algorytmu „branch and bound” pełni funkcja wykorzystująca algorytm Djikstry do sprawdzenia czy jest nadal możliwe dojście do wszystkich nieodwiedzonych do tej pory wierzchołków grafu na podstawie obliczania długości ścieżki prowadzącej od ostatniego dopisanego do cyklu wierzchołka do wszystkich pozostałych wierzchołków bez odwiedzania wierzchołków już odwiedzonych, która odcina daną gałąź rozwiązania, jeśli przestała istnieć możliwość odwiedzenia wszystkich wierzchołków. Po dopisaniu danego wierzchołka do cyklu sprawdzana jest długość cyklu i jeśli jest ona równa liczbie wierzchołków oraz ostatni wierzchołek ma wspólną krawędź z ostatnim, to cykl jest dopisywany do listy cykli. W przeciwnym razie funkcja wywołuje się rekurencyjnie dopisując przy każdym wywołaniu innego sąsiada ostatniego wierzchołka. Dzieje się tak dopóki nie zostanie znaleziony cykl (sprawdzanie czy istnieje) lub nie zakończą się wszystkie wywołania rekurencyjne (szukanie wszystkich cykli.

# II Wyniki doświadczenia

# III Klasyfikacja problemów

# IV Poszukiwanie cyklu Eulera

# V Poszukiwanie cyklu Hamiltona