|  |  |
| --- | --- |
| 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, jednego i wszystkich cyklów Hamiltona,

- klasyfikacja problemów,

- wyznaczenie złożoności obliczeniowej algorytmów je rozwiązujących

- oraz zbadanie wpływu gęstości grafu na czas działania.

Wnioski przedstawiono na podstawie przeprowadzonych pomiarów czasu wyszukiwania jednego cyklu Eulera, jednego cyklu Hamiltona oraz 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. Następnie mierzy czas wyszukiwania cyklu Eulera za pomocą algorytmu opartego o algorytm przechodzenia wgłąb (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

Badanie polegało na wygenerowaniu losowych grafów złożonych z n wierzchołków, gdzie n należy do zbiorów {7, 8, 9, …, 12} oraz {13, 15, 17, …, 31}. Łącznie doświadczenie przeprowadzono dla 16 punktów pomiarowych. Następnie

W pierwszych siedmiu punktach pomiarowych (7-13) został zmierzony czas poszukiwania wszystkich cykli Hamiltona w grafie. Dla punktu pomiarowego o wartości n = 13 czas ten stał się zbyt duży (1073s), więc dla kolejnych punktów pomiarowych wykonywane były tylko pomiary czasu wykonania algorytmów poszukującego cykl Eulera oraz jednego cyklu Hamiltona. Dodatkowo n zwiększano z krokiem o 2, aby poszerzyć zakres wartości n.

# II Wyniki doświadczenia

# Generowanie macierzy

Jako reprezentację grafu wybrana została macierz sąsiedztwa. Zapewnia ona łatwą implementację oraz przejrzystość debugowania. Najlepsze pod względem złożoności czasowej byłoby wykorzystanie listy sąsiedztwa, która pozwala na znajdowanie szybsze sąsiadów, bez zbędnego sprawdzania istnienia krawędzi z każdym z wierzchołków. W najgorszym przypadku, poszukując sąsiadów danego wierzchołka algorytm przechodziłby przez wszystkie krawędzie w grafie, stąd złożoność znajdowania sąsiadów O(m), gdzie m – liczba krawędzi. Średnio znajdowanie sąsiadów trwa m/n operacji, przez losowy rozkład krawędzi między wierzchołkami. W macierzy sąsiedztwa sprawdzane jest kolejno, czy dany wierzchołek łączy się z innym, a więc wyszukiwanie sąsiadów ma złożoność O(n), n – liczba wierzchołków.

W każdym z punktów pomiarowych wygenerowana została macierz sąsiedztwa przez utworzenie statycznej tablicy dwuwymiarowej n na n, gdzie n oznacza liczbę wierzchołków, wypełnionej zerami. Następnie w macierzy generowane są krawędzie, prawdopodobieństwo utworzenia krawędzi określone jest przez gęstość – d. Następuje sprawdzenie, czy graf jest spójny, poprzez zastosowanie algorytmu Djikstry.

Należy upewnić się, że graf jest Eulerowski, to znaczy istnieje w nim cykl Eulera. Warunkiem koniecznym i dostatecznym istnienia cyklu Eulera w grafie spójnym jest to, że wszystkie wierzchołki mają stopień parzysty. Wykorzystamy właściwość grafu nieskierowanego - suma stopni wszystkich wierzchołków jest liczbą parzystą (ponieważ każda krawędź łączy dwa wierzchołki). Oznacza to, że jeśli istnieje jakikolwiek wierzchołek o stopniu nieparzystym, to istnieje również co najmniej jeden inny taki wierzchołek (suma liczb nieparzystych jest liczbą parzystą). W kolejnym kroku algorytmu 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 (jednak w najgorszym scenariuszu, gdy występuje więcej par jednego rodzaju, gęstość grafu się zmienia, ponieważ algorytm nie może wykonywać tych operacji naprzemiennie).

# Klasyfikacja problemów

Użyty w badaniu algorytm poszukiwania cyklu Eulera wykorzystuje do trawersowania grafu rekurencyjną procedurę przechodzenia w głąb (DFS, depth-first search). Wybierany jest wierzchołek startowy (w implementacji - wybierany jest wierzchołek o indeksie 0). Następnie rozpoczyna się przechodzenie grafu, podczas którego algorytm przechodzi do dowolnego (w implementacji - o najmniejszym indeksie) sąsiada. W trakcie przechodzenia struktury, usuwane są wykorzystane krawędzie. W przypadku dotarcia do wierzchołka, który nie ma sąsiadów, algorytm dodaje taki wierzchołek na stos oraz cofa się do poprzednio rozważanego wierzchołka. Mamy pewność, że w grafie znajduje się cykl Eulera (poprzez wcześniejsze sprawdzenie warunku koniecznego i dostatecznego). Powstały stos zawiera numery kolejnych wierzchołków w jednym z cykli Eulera. Posiada m elementów, tyle samo ile jest krawędzi w grafie.

Złożoność implementacji tego algorytmu jest zależna od wybranej reprezentacji grafu. W trakcie przechodzenia grafu, dla każdego wierzchołka należy znaleźć dowolnego sąsiada. Złożoność tego wyszukiwania można określić jako O(n\*x), gdzie x to złożoność znajdowania jednego sąsiada w reprezentacji grafu. W macierzy sąsiedztwa x = O(n), ponieważ kolejno sprawdzamy, czy istnieje połączenie z każdym wierzchołkiem. Dlatego złożoność znajdowania cyklu Eulera wynosi O(n\*O(m)) = O(n\*m).

Istnieje algorytm odnajdujący cykl Eulera w czasie wielomianowym, a więc problem ten można zaklasyfikować jako „łatwy”.

Na wykresach 1 oraz 2 można zaobserwować znaczącą różnicę w czasie wykonywania poszukiwania wszystkich cykli Hamiltona w grafie o gęstości 0,6 i reszty problemów. Przykładowo, dla grafu o 12 wierzchołkach, znalezienie przez program HA trwało ponad 17 minut, a cykl Eulera został znaleziony w ok. 0,01 sekundy. [Sposób znajdowania wszystkich cykli Hamiltona]

- interpretacja wyników

- klasyfikacja problemu (P, NP, NP- zupełność)

- jakie konsekwencje dla możliwej złożoności obliczeniowej algorytmu ta pociąga?

- uzasadnienie złożoności algorytmu, jaki to rodzaj złożoności ze względu na jego złożoność?

# Poszukiwanie cyklu Eulera

- złożoność obl ze względu na działanie algorytmu dfsEuler

- kon i dost warunek

- dlaczego w grafach istniał euler? Metoda generacji

- jaka jest najlepsza repr grafu i dlaczego?

- wpływ repr na złożoność

- jaki jest wpływ liczby wierzchołków, a jaki liczby krawędzi na czas działania metody (należy zanalizować wpływ n i m, bez przyjmowania założenia że m = O(n^2) )?

# Poszukiwanie cyklu Hamiltona