|  |  |
| --- | --- |
| 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.

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.

Czas na wykresach jest liczony w sekundach.

# II Wyniki doświadczenia

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Gęstość 0,2 | | | | | | |
| liczba\_wierzchołków | 7 | 8 | 9 | 10 | 11 | 12 |
| euler | 0,00797 | 0,00651 | 0,00754 | 0,00691 | 0,00767 | 0,0102 |
| hamilton\_1 | 0,00511 | 0,00582 | 0,01464 | 0,00294 | 0,00749 | 0,00852 |
| hamilton\_all | 0,00917 | 0,01012 | 0,00797 | 0,04009 | 0,15374 | 0,59861 |
| liczba cykli | 0 | 2 | 2 | 24 | 240 | 640 |
| liczba\_wierzchołków | 13 | 15 | 17 | 19 | 21 | 23 |
| euler | 0,00753 | 0,00653 | 0,00802 | 0,00776 | 0,01066 | 0,00757 |
| hamilton\_1 | 0,01017 | 0,00828 | 0,03552 | 0,02378 | 0,01631 | 0,02229 |
| Liczba\_wierzchołków | 25 | 27 | 29 | 31 |  |  |
| 0,00757 | 0,00858 | 0,00666 | 0,01055 | 0,00968 |  |  |
| 0,02229 | 0,0239 | 0,02307 | 0,03011 | 0,07699 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Gęstość 0,6 | | | | | | |
| liczba\_wierzchołków | 7 | 8 | 9 | 10 | 11 | 12 |
| euler | 0,00898 | 0,00265 | 0,00712 | 0,00697 | 0,01015 | 0,00815 |
| hamilton\_1 | 0,00718 | 0,00745 | 0,00886 | 0,001455 | 0,00799 | 0,00943 |
| hamilton\_all | 0,00771 | 0,09554 | 2,72758 | 8,32987 | 87,8485 | 1073,41 |
| liczba\_cykli | 28 | 360 | 11568 | 30336 | 327936 | 3528096 |
| Liczba\_wierzchołków | 13 | 15 | 17 | 19 | 21 | 23 |
| euler | 0,00757 | 0,00892 | 0,00754 | 0,00863 | 0,00777 | 0,00765 |
| Hamilton\_1 | 0,00878 | 0,01001 | 0,00832 | 0,00731 | 0,01032 | 0,00759 |
| Liczba\_wierzchołków | 25 | 27 | 29 | 31 |  |  |
| euler | 0,00813 | 0,00737 | 0,00861 | 0,00858 |  |  |
| Hamilton\_1 | 0,00794 | 0,01951 | 0,0243 | 0,02355 |  |  |

(Wykres 1)

(Wykres 2)

(Wykres 3)

(Wykres 4)

(Wykres 5)

(Wykres 6)

# 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).

# Poszukiwanie cyklu Eulera

Wykres 3 przedstawia wahania w czasie wykonania algorytmu poszukującego jednego cyklu Eulera. 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, tj. graf jest spójny i wszystkie jego wierzchołki mają stopień parzysty). 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”.

# Poszukiwanie cyklu Hamiltona

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.

Wyszukiwanie cyklu polega na rozpoczęciu od dowolnego wierzchołka (w implementacji wierzchołek o indeksie 0). Rozpoczyna się przechodzenie grafu algorytmem DFS, podczas którego przechodzi się do dowolnego (w implementacji - o najmniejszym indeksie) sąsiada. Każdy przeglądany wierzchołek dopisuje się na stos (stos ten będzie zawierał aktualnie przeglądaną gałąź drzewa rozwiązań, czyli ścieżkę od wierzchołka startowego do aktualnie przeglądanego). Jeżeli ten stos zawiera wszystkie n wierzchołków, oznacza to że algorytm znalazł rozwiązanie. Aby upewnić się, że rozważamy jedynie sensowne ścieżki, w następnym kroku algorytm sprawdza, czy da się dotrzeć do nieodwiedzonych wierzchołków. Tą funkcję pełni procedura oparta o algorytm Djikstry. Jeżeli z którymś z wierzchołków nie da się znaleźć połączenia, algorytm powraca do poprzedniego punktu, w którym można jeszcze szukać rozwiązań.

Jeżeli stos zawiera wszystkie n wierzchołków, oznacza to że algorytm znalazł rozwiązanie. W takim wypadku, algorytm kończy działanie, jeśli szukał jednego rozwiązania, lub cofa się w poszukiwaniu innych rozwiązań. Wyszukiwanie H1 i szukanie wszystkich cykli HA różnią się tym, że algorytm poszukujący H1 zatrzymuje się po znalezieniu pierwszego rozwiązania, a drugi algorytm „powraca” w celu znalezienia wszystkich możliwych rozwiązań problemu.

Warunek istnienia cyklu Hamiltona w grafie, który byłby jednocześnie konieczny i wystarczający, nie został odnaleziony. Z tego powodu, aby odpowiedzieć na problem decyzyjny (czy istnieje), należy znaleźć rozwiązanie problemu poszukiwania (znaleźć cykl Hamiltona). Zwiększa to też prawdopodobieństwo, że problem należy do klasy „trudnych”, tzn. że nie posiada algorytmu rozwiązującego go w czasie wielomianowym. Jednak brak znalezionego rozwiązania nie udowadnia, że takie rozwiązanie nie istnieje.

Wierzchołek startowy może posiadać maksymalnie n-1 sąsiadów, gdzie n to liczba wierzchołków. Po przejściu do kolejnego wierzchołka, liczba nieodwiedzonych wierzchołków, a więc i potencjalnych sąsiadów, spada do n-2. Podczas przechodzenia po grafie, algorytm ma więc n-1 możliwości wybrania pierwszego wierzchołka, (n-1)\*(n-2) możliwości wybrania drugiego itd. Pojawia się więc maksymalnie (n-1)! ścieżek do rozważenia. Jednak każda ścieżka ma swój odpowiednik, będący tym samym rozwiązaniem przeglądanym w odwrotnej kolejności. Stąd maksymalna liczba potencjalnych rozwiązań wynosi (n-1)!/2. Tak więc złożoność obliczeniowa algorytmu poszukiwania wszystkich cykli Hamiltona w grafie wynosi O((n-1)!/2) = O(n!).

Podczas porównywania wykresów 5 i 6, można zauważyć korelację między wzrastającą liczbą cykli Hamiltona oraz czasem działania funkcją je znajdującą.