Grafy, algorytmy BFS i DFS

Czwartek 12:15-15:00

Mgr inż. Andrzej Wytyczak-Partyka

Szymon Płaneta

Spis treści

1. Zadanie

Wstęp

Zadaniem było utworzenie grafu i zaimplementowanie algorytmów przeszukiwania wszerz (BFS) i przeszukiwania w głąb (DFS). Następnie dokonanie ich porównania poprzez przetestowanie czasów przeszukiwania grafu.

2. Graf

Implementacja

Istnieją dwa standardowe sposoby reprezentowania grafu - za pomocą list sąsiedztwa lub macierzy sąsiedztwa. Oba sposoby mają swoje wady i zalety. Reprezentacja listowa umożliwia przedstawienie w zwarty sposób grafów rzadkich (liczba krawędzi znacznie mniejsza od kwadratu liczby wierzchołków). W przypadku, gdy graf jest gęsty, lub gdy istnieje potrzeba szybkiego stwierdzania, czy istnieje krawędź łącząca dwa dane wierzchołki, wtedy korzystniejsza będzie reprezantacja macierzowa.
W swoim programie zdecydowałem się na reprezentację listową, korzystającą z własnych struktur utworzonych we wcześniejszej fazie semestru.

3. Algorytm BFS

Implementacja

Przy starcie algorytmu tworzone są 3 tablice - jedna przechowuje informacje o kolorze wierzchołków, druga o odległości od źródła, trzecia o poprzeniku wierzchołka. Wierzchołek może mieć jeden z trzech kolorów - biały ('w'), gdy nie był jeszcze odwiedzony, szary ('g'), gdy był już odwiedzony, ale posiada białych sąsiadów lub czarny ('b') - był odwiedzony i nie posiada już nieodwiedzonych (białych) sąsiadów. Na początku wszystkie wierzchołki kolorowane są na biało, dystans od źródła i numer wierzchołka poprzedniego również na -1. Algorytm przeszukiwania wszerz tworzy kolejkę, w której przechowuje wierzchołki do przetworzenia. Obsługiwane są one zgodnie z rosnącą odległością od źródła. Niewątpliwie znaczną zaletą tego algorytmu w przypadku wyszukiwania ścieżki pomiędzy dwoma wierzchołkami jest to, że ścieżka taka będzie zawsze ścieżką najkrótszą.

Przeprowadzone testy

Wyniki przeprowadzonych testów zostaną przedstawione na wykresie w rozdziale 5. Porównanie.

4. Algorytm DFS

Implementacja

Przeszukiwaniw w głąb, jak sama nazwa wskazuje polega na sięganiu "głębiej" w graf, jeżeli jest to tylko możliwe. Przy przeszukiwaniu w głąb są badane krawędzie ostatnio odwiedzonego wierzchołka v, z którego wychodzą jeszcze niezbadane krawędzie. Gdy wszystkie krawędzie są zbadane, przeszukiwanie wraca do wierzchołka, z którego wierzchołek v został odwiedzony. Proces ten jest kontynuowany, aż wszystkie wierzchołki osiągalne z początkowego wierzchołka nie zostaną odwiedzone. Jeśli pozostaną nieodwiedzone wierzchołki, wybierany jest jeden z takich wierzchołków jako nowe źródło i algorytm powtarzany jest z tego miejsca. Całość powtarza się, dopóki wszystkie wierzchołki w grafie nie zostaną odwiedzone. Podobnie jak w przypadku algorytmu BFS, tworzone są tablice zawierające informacje o kolorze i poprzedniku wierzchołka. Różna jest natomiast tablica zachowująca informację o odległości. W wypadku tego algorytmu, zastosowano zmienną time, która zawiera informację o aktualnym kroku obliczeń. W tablicy d przechowywane są informacje o tym, w jakim kroku wierzchołek został pierwszy raz odwiedzony (i pokolorowany na szaro), a w tablicy f, kiedy każdy wierzchołek został przetworzony (i pokolorowany na czarno).

Przeprowadzone testy

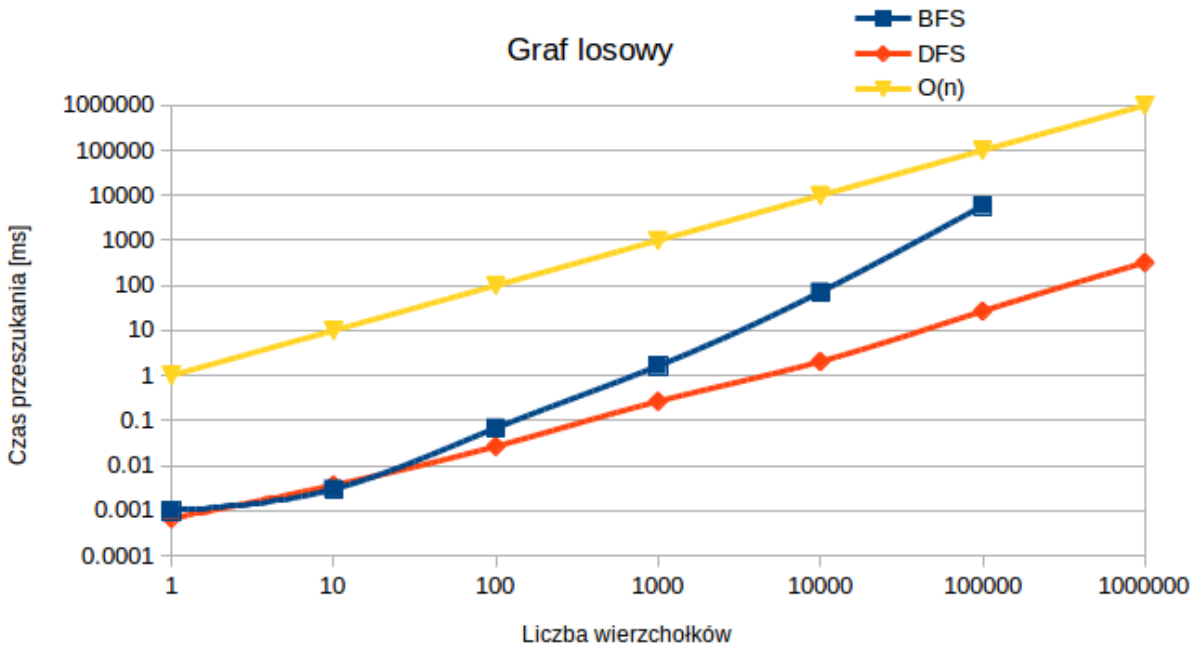
Wyniki przeprowadzonych testów zostaną przedstawione na wykresie w rozdziale 5. Porównanie.

5. Porównanie

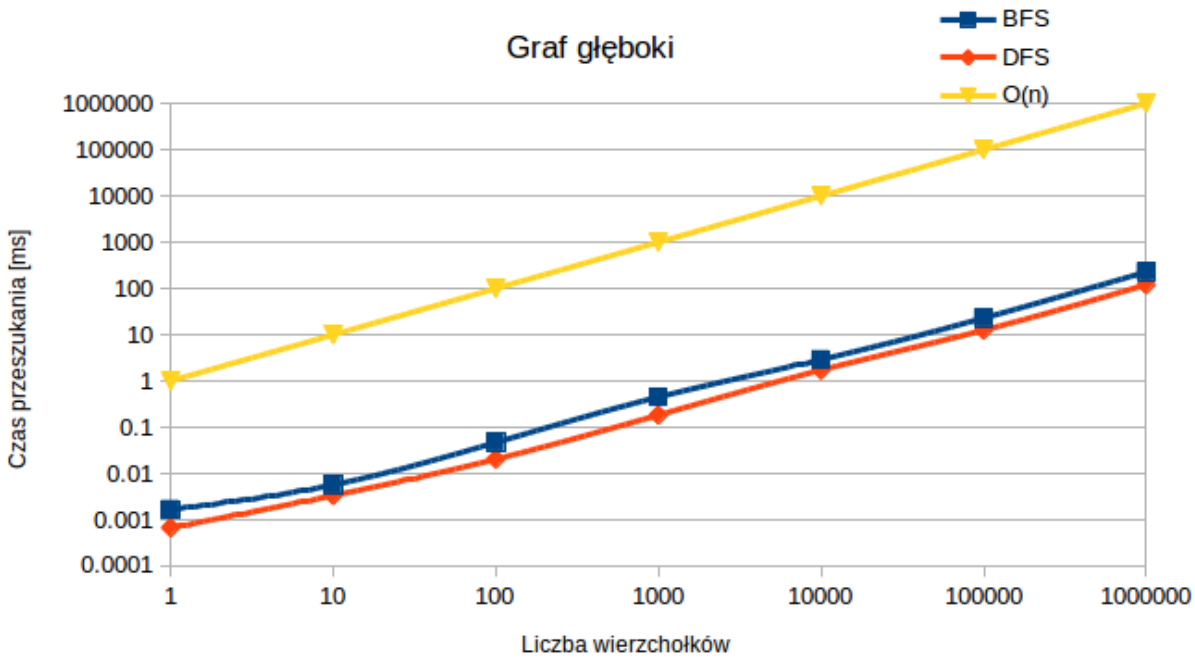
Dokonano porównania przeszukiwania całego grafu dla trzech przypadków:

- Graf spójny losowy (wierzchołek 0 połączony z 1, każdy kolejny wierzchołek i połączony z losową liczbą z przedziału [0:(i-1)]
- Graf głęboki (wierzchołek $0 \leftrightarrow 1, 1 \leftrightarrow 2, 2 \leftrightarrow 3, ..., (i-1) \leftrightarrow i$)
- Graf płytki (wierzchołek $0 \leftrightarrow 1, 0 \leftrightarrow 2, 0 \leftrightarrow 3, ..., 0 \leftrightarrow (i-1), 0 \leftrightarrow i$)

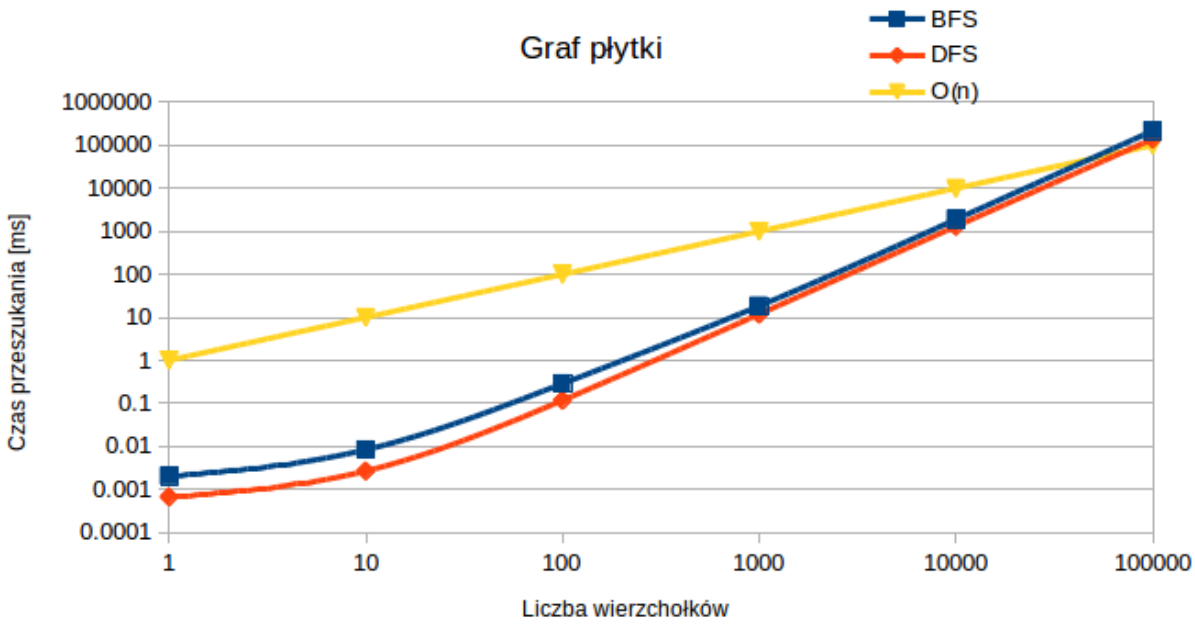
Poniżej przedstawiono czasy działania poszczególnych algorytmów na wspólnych wykresach.



Rys. 1: Czasy operacji przeszukania grafu losowego



Rys. 2: Czasy operacji przeszukania grafu głębokiego



Rys. 3: Czasy operacji przeszukania grafu płytkiego

6. Wnioski

Czas przeszukania grafu zależy od wykorzystanego algorytmu, liczby wierzchołków i krawędzi, oraz od właściwości danego grafu.
W każdym z testowanych przypadków, czasy przeszukiwania w głąb były krósze niż czasy przeszukiwania wszerz (zazwyczaj nieznacznie). Przewagą algorytmu BFS jest jednak znajdowanie zawsze najkrótszej ścieżki.
W ogólności, czas działania obu algorytmów powinien wynosić $O(V + E)$, gdzie V to liczba wierzchołków, a E to liczba krawędzi.
Różnice występują z różnych powodów - w algorytmie BFS czas dodawnia i usuwania z kolejki powinien wynosić $O(1)$, jednak z powodu sposobu implementacji kolejki (która zawiera wskaźnik jedynie na swój początek), czas dodawania rośnie do O(n), gdzie n to liczba elementów znajdujących się w kolejce. Zwiekszony jest również czas spędzony na przeglądaniu list sąsiedztwa - przy odwoływaniu się do każdego kolejnego elementu listy, jest ona przechodzona za każdym razem od początku. Jest to szczególnie widoczne dla przypadku grafu płytkiego - lista sąsiedztwa wierzchołka 0 zawiera V elementów, przechodzenie jej za każdym razem od początku powoduje znaczny wzrost czasów wykonania algorytmów zarówno BFS jak i DFS. W przypadku wykorzystania reprezentacji macierzowej grafu czasy te byłyby znacznie krótsze, ponieważ odwoływanie się do elementów następowałoby natychmiastowo. Przypadek grafu głębokiego, gdzie każdy wierzchołek ma 2 krawędzie, łączące go z wierzchołkiem poprzednim jak i następnym, przebiegło zgodnie z oczekiwaniami. W przypadku losowym, algorytm BFS wypadł znacznie słabiej z uwagi na wspomniany wcześniej czas wstawiania elementów do kolejki. Rozwiązaniem byłaby modyfikacja kolejki, aby zawierała również wskaźnik na swój koniec, co skróciłoby czasy wstawiania z $O(n)$ do $O(1)$.