Graf, przeszukiwanie DFS i BFS

218582

7 maja 2016

Spis treści

Spis rysunków

1 Wstęp

Zadanie polegało na zaimplementowaniu grafu oraz algorytmów jego przeszukiwania: DFS (przeszukiwanie wgłąb) i BFS (przeszukiwanie wszerz) a także zbadaniu czasu trwania wykonywania się przeszukiwań dla zadanej ilości krawędzi.

2 Wybór sposobu reprezentacji grafu

Powiązania między wierzchołkami grafu reprezentowane są za pomocą listy sąsiedztwa - tablicy dynamicznie rozszerzalnej, która przechowuje wskaźniki na kolejne tablice tego typu.

Pierwsza wymieniona tablica identyfikuje wierzchołek. W drugiej przechowywane są indeksy wierzchołków, które mają wspólną krawędź z danym wierzchołkiem.

Takie rozwiązanie zajmuje mniej pamięci niż macierz powiązań, o ile ilość krawędzi jest niewielka. W liście sąsiedztwa przechowywanych jest:

$$N = 2 \cdot (\text{ilość krawędzi})$$

elementów, ponieważ każde powiązanie jest zapisane w tablicach obu wierzchołków.

Jeśli ilość krawędzi dąży do maksymalnej (każdy wierzchołek połączony z każdym), to taka tablica przechowuje tyle samo elementów co macierz powiązań (tj. (ilość wierzchołków)^2).

Należy zwrócić uwagę, że tablica dynamicznie rozszerzalna powiększa zaalokowaną przestrzeń dwa razy gdy jej brakuje. To działa na niekorzyść takiej reprezentacji grafu dla dużych gęstości krawędzi, jeśli chodzi o wykorzystaną pamięć. W takiej sytuacji, macierz powiązań jest lepszym rozwiązaniem.

Użyta implementacja pozwala natomiast na szybkie znalezienie sąsiadów danego wierzchołka - wystarczy zwrócić tablicę pod odpowiednim indeksem.

Dla niewielkiej ilości krawędzi unika się również alokowania natychmiast tablicy o wielkości (ilość wierzchołków)^2.

3 Generowanie grafu spójnego

Graf spójny uzyskuje się w wyniku działania następującego (w przybliżeniu) algorytmu:

- dodaj wierzchołek do grafu,
- dodaj krawędź między ostatnio dodanym wierzchołkiem a losowym wierzchołkiem spośród wcześniej dodanych.

Takie rozwiązanie pozwala na stworzenie minimalnego grafu spójnego.

Aby zwiększyć liczbę krawędzi, stosuje się następujący algorytm:

- Jeśli krawedź między dwoma wylosowanymi wierzchołkami nie istnieje lub oba wylosowane wierzchołki są różne: dodaj krawędź między dwoma losowymi wierzchołkami.
- W przeciwnym wypadku losuj wierzchołki jeszcze raz.

Takie rozwiązanie wymusza stworzenie określonej ilości krawędzi.

Przy ilości krawędzi zmierzającej do maksymalnej, którą to ilość można obliczyć w następujący sposób:

$$E_{max} = \sum_{V=0}^{V-1} V_i$$

powyższe rozwiązanie staje się nieefektywne, gdyż pętla `while`, w której losowane są dodatkowe wierzchołki może wykonywać się bardzo długo w końcowej fazie (aż trafi na pozostałą parę wierzchołków).

Stąd też utworzono specjalną metodę do tworzenia grafu dla maksymalnej ilości krawędzi. W tym wypadku łączy się elementy każdy z każdym.

Wybrana reprezentacja grafu (2) powoduje jednak, że tworzenie takiego drzewa zajmuje więcej pamięci w stosunku do innych implementacji. W testach unikać się będzie przeprowadzania ich dla dużej liczby krawędzi i dużej liczby wierzchołków jednocześnie.

4 Oczekiwane rezultaty

Zarówno algorytm BFS jak i DFS charakteryzują się teoretyczną złożonością $O(V + E)$, gdzie V - liczba wierzchołków, E - liczba krawędzi.

Należy pamiętać, że w przypadkach niekorzystnych dla danej implementacji grafu, złożoność może znacząco odbiegać od teoretycznej.

5 Testy

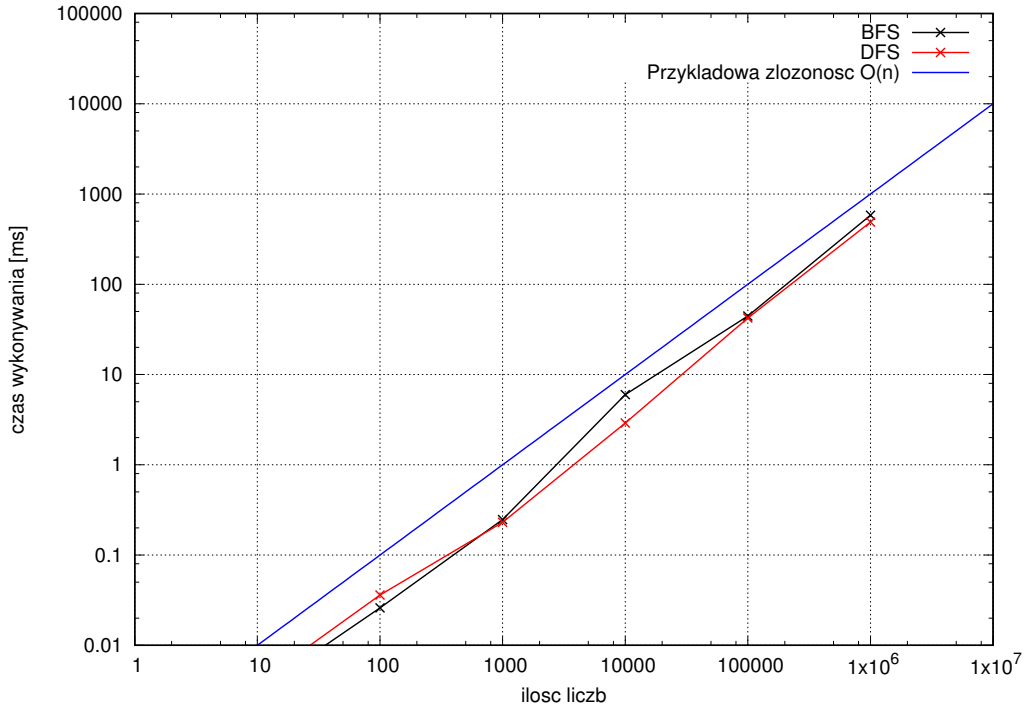
W celu porównania jaki wpływ na czas wykonywania się operacji przeszukania grafu wzdłuż i wszerz mają poszczególne jego parametry, wykonano następujące testy:

1. Czas przeszukiwań DFS i BFS dla kolejnych wielkości grafu z:
   (a) najmniejszą liczbą połączeń potrzebną do stworzenia grafu spójnego
   (b) z liczbą połączeń dwukrotnie większą niż powyższa
   (c) z liczbą krawędzi 20-krotnie mniejszą niż maksymalna liczba krawędzi (ale nie mniej niż najmniejsza wymagana do grafu spójnego)
2. Czas przeszukiwań DFS i BFS dla 1000 elementów w grafie:
   (a) dla najmniejszej ilości krawędzi
   (b) dla 25% gęstości grafu (25% wszystkich możliwych krawędzi)
   (c) dla 50% gęstości grafu (50% wszystkich możliwych krawędzi)
   (d) dla 75% gęstości grafu (75% wszystkich możliwych krawędzi)
   (e) dla grafu z największą możliwą ilością połączeń

6 Wyniki

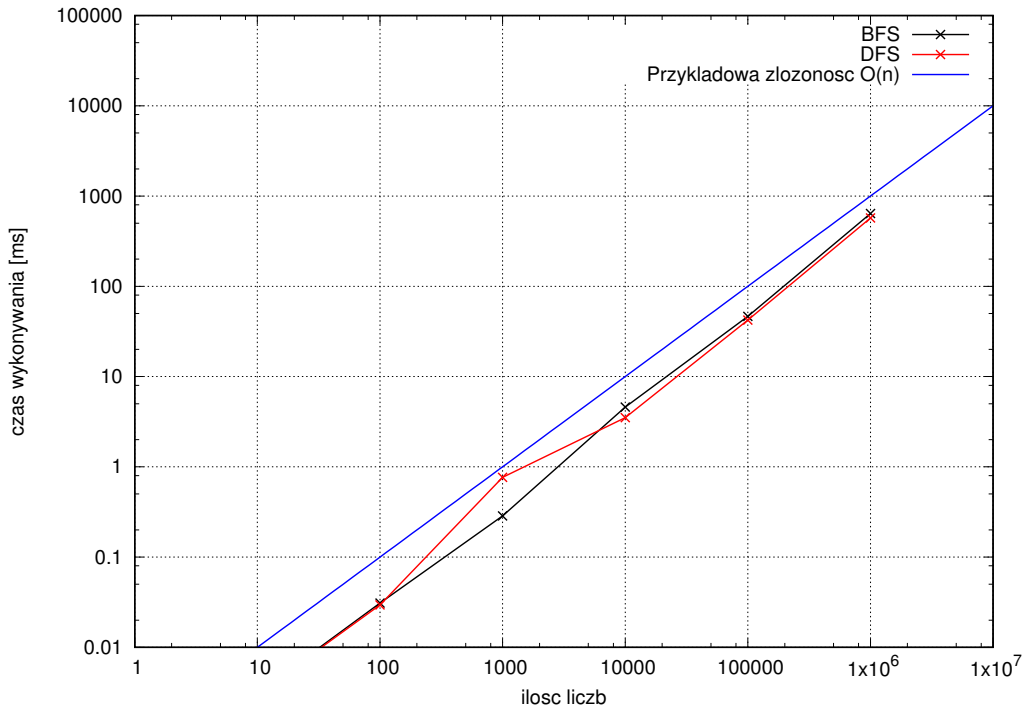
6.1 Czas przeszukiwań DFS i BFS dla kolejnych wielkości grafu

6.1.1 Graf z najmniejszą liczbą połączeń potrzebną do stworzenia grafu spójnego



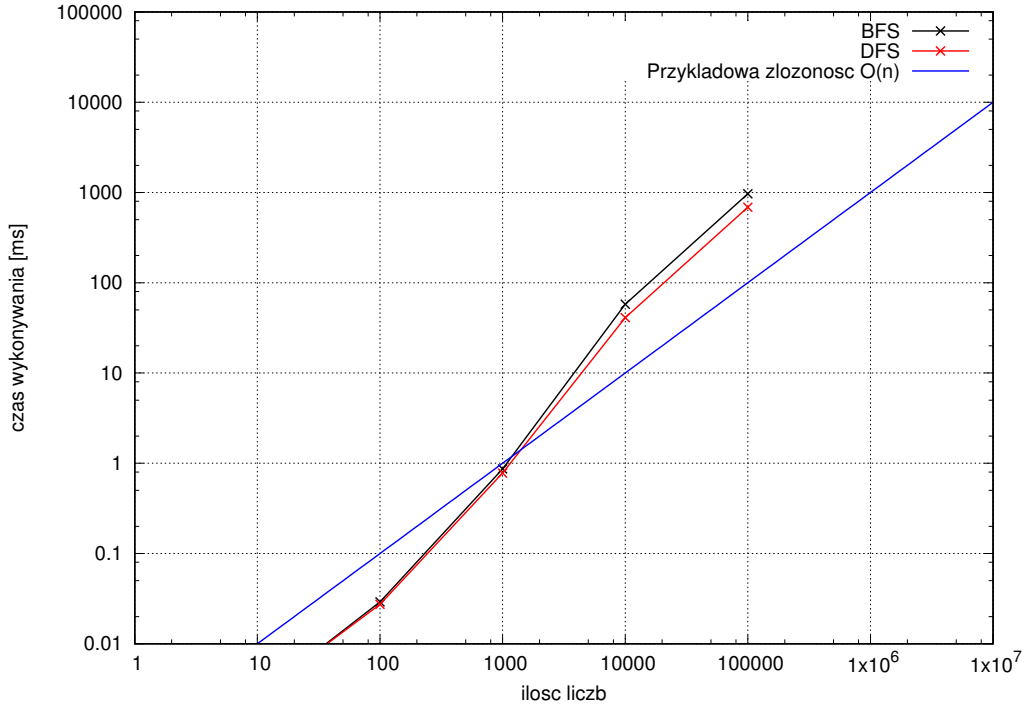
Rysunek 1: Przeszukiwanie DFS i BFS w grafie - najmniejsza liczba połączeń, graf spójny

6.1.2 Graf z liczbą krawędzi dwukrotnie większą od najmniejszej liczby połączeń potrzebnej do stworzenia grafu spójnego



Rysunek 2: Przeszukiwanie DFS i BFS w grafie - 2x najmniejsza liczba połączeń, graf spójny

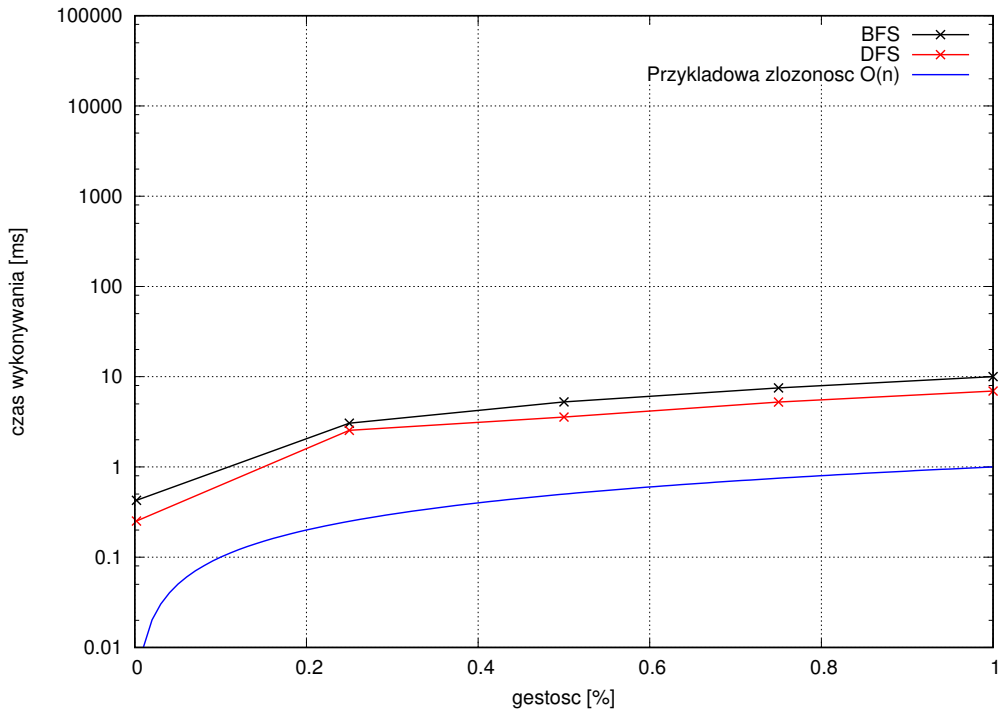
6.1.3 Graf z liczbą krawędzi 20-krotnie mniejszą niż maksymalna liczba krawędzi



Rysunek 3: Przeszukiwanie DFS i BFS w grafie - liczba krawędzi 20x mniejsza niż maksymalna, graf spójny

6.2 Czas przeszukiwań DFS i BFS dla 1000 elementów w grafie dla różnych gęstości grafu

Maksymalna liczba krawędzi dla 1000 wierzchołków wynosi 499500, minimalna do grafu spójnego - 999 (gęstość: 0,2%).



Rysunek 4: Przeszukiwanie DFS i BFS w grafie - liczba krawędzi 1000, zmieniająca się gęstość krawędzi, graf spójny

7 Wnioski

- Zarówno z wykresu 1 jak i 2 wynika, że dla zmieniającej się ilości wierzchołków w grafie, złożoność obu badanych algorytmów w przybliżeniu wynosi $O(n)$. Odchylenia od tej złożoności w niektórych fragmentach wykresu prawdopodobnie wynikają z użytych rozwiązań w implementacji algorytmów czy reprezentacji grafu.

- Dla coraz większej ilości wierzchołków i dużej ilości krawędzi staje się wyraźna różnica między złożonością teoretyczną a empiryczną (Rys. 3). Prawdopodobnie wynika ona z faktu, że wykorzystana reprezentacja krawędzi grafu źle radzi sobie przy dużej ilości krawędzi.

- Z wykresu na rysunku 4 można wywnioskować, że zmiana gęstości krawędzi przy stałej liczbie wierzchołków również daje możliwość odczytania złożoności na poziomie $O(n)$.

- Zarówno ilość wierzchołków jak i krawędzi wpływa na złożoność algorytmów DFS i BFS. Badanie wpływu pojedynczego składnika w obu przypadkach dało złożoność na poziomie $O(n)$.

Tablica 1: Zbiorcza tablica wyników - testy dla zmiany ilości wierzchołków

Najmniejsza liczba krawędzi, spójny graf		
Algorytm	Ilość wierzchołków	Czas średni [ms]
DFS	10	0.0038051
	100	0.0360605
	1000	0.230574
	10000	2.90005
	100000	42.2752
	1000000	489.777
BFS	10	0.0030559
	100	0.0260175
	1000	0.247106
	10000	5.98665
	100000	44.3966
	1000000	582.96
Najmniejsza liczba krawędzi x 2, spójny graf		
Algorytm	Ilość wierzchołków	Czas średni [ms]
DFS	10	0.0030504
	100	0.0295455
	1000	0.763592
	10000	3.50782
	100000	42.3198
	1000000	571.041
BFS	10	0.0030984
	100	0.031002
	1000	0.286247
	10000	4.59653
	100000	46.402
	1000000	641.062
Liczba krawędzi 20x mniejsza od największej, spójny graf		
Algorytm	Ilość wierzchołków	Czas średni [ms]
DFS	10	0.0026508
	100	0.0273721
	1000	0.777923
	10000	41.0882
	100000	685.996
BFS	10	0.0025765
	100	0.0291032
	1000	0.866566
	10000	57.7701
	100000	967.993

Tablica 2: Zbiorcza tablica wyników - testy dla zmiany gęstości krawędzi

1000 wierzchołków, graf spójny		
Algorytm	Gęstość krawędzi [%]	Czas średni [ms]
DFS	0.2	0.251416
	25	2.54407
	50	3.57055
	75	5.24478
	100	6.92155
BFS	0.2	0.425972
	25	3.05527
	50	5.25072
	75	7.51591
	100	10.0098