Opracowanie algorytmów grafowych

Szymon Kaczmarek, nr indeksu 148056

1 Wprowadzenie

W niniejszym opracowaniu zostaną przedstawione trzy reprezentacje grafu oraz dwa algorytmy sortowania topologicznego grafu. Również poruszę problem generowania „DAG'ów". Wszystkie algorytmy zostały zaimplementowane w języku Python (link do repozytorium GitHub).

2 Reprezentacje grafu

W tej części zostaną omówione różne reprezentacje grafu.

2.1 Lista krawędzi

Ta reprezentacja grafu polega na zapisaniu listy wszystkich występujących krawędzi w grafie. Krawędzie są zapisane w postaci v -> u. Jest to bardzo prosta reprezentacja grafu, jednak złożoność podstawowych operacji pozostawia wiele do życzenia. Dla n wierzchołków oraz m krawędzi złożoności są następujące:

- Złożoność pamięciowa: $O(m)$
- Sprawdzenie czy dana krawędź istnieje: $O(m)$
- Wyszukanie wszystkich następników danego wierzchołka: $O(m)$

2.2 Lista następników

W tej reprezentacji przechowujemy informacje tylko o następnikach danego wierzchołka. W tablicy o wielkości rzędu grafu, w komórkach odpowiadających poszczególnym wierzchołkom są umieszczane kolejne następniki danego węzła. Na przykład dla krawędzi 1 -> 2, 1 -> 4, lista będzie wyglądać 1 -> 2 -> 4, gdzie 1 to indeks w tablicy. Dla n wierzchołków oraz m krawędzi złożoności są następujące:

- Złożoność pamięciowa: $O(n+m)$
- Sprawdzenie czy dana krawędź istnieje: średnio $O(m/n)$, pesymistycznie $O(n)$
- Wyszukanie wszystkich następników danego wierzchołka: $O(1)$

2.3 Macierz sąsiedztwa

Ostatnia reprezentacja polega na stworzeniu macierzy $n \times n$ (gdzie n to rząd grafu) oraz zaznaczeniu w odpowiednich miejscach:

- $macierz_{i,j} = 0$ - Jeżeli krawędź i -> j nie istnieje
- $macierz_{i,j} = 1$ - Jeżeli krawędź i -> j istnieje
- $macierz_{i,j} = -1$ - Jeżeli krawędź j -> i istnieje

Dla n wierzchołków oraz m krawędzi złożoności są następujące:

- Złożoność pamięciowa: $O(n^2)$
- Sprawdzenie czy dana krawędź istnieje: $O(1)$
- Wyszukanie wszystkich następników danego wierzchołka: $O(n)$

3 Generowanie losowych instancji grafowych

Sortowanie topologiczne, które będzie niżej omówione, można wykonać jedynie na tak zwanych „DAG'ach". „DAG" to, z angielskiego, acykliczny graf skierowany, którego generacją trzeba było się zająć przed sprawdzaniem poprawności algorytmów sortowania. Podejście, które podjąłem, to losowanie nowych krawędzi grafu oraz sprawdzanie na bieżąco czy graf po dodaniu danej krawędzi zawiera cykl. Jeżeli taki cykl by się pojawił, to należy usunąć nowo dodaną krawędź i wylosować inną. Maksymalna ilość krawędzi w „DAG'u" wynosi $(N-1)(N)/2$, gdzie N to rząd grafu. W kodzie jest przyjęta granica $\frac{(N-1)(N)/2}{2}$ celem otrzymania „rozsądnie" wypełnionego grafu.

4 Algorytmy sortowania topologicznego

W tej części zostaną omówione dwa algorytmy sortowania topologicznego grafu.

4.1 Sortowanie topologiczne Kahn'a

To sortowanie polega na obraniu za punkt początkowy wierzchołka o stopniu krawędzi wchodzących równym zero. Następnie usuwamy dany węzeł oraz wszystkie krawędzie wychodzące i szukamy kolejnego wierzchołka o stopniu zero. Kroki te powtarzamy aż do wyczerpania wierzchołków w grafie, bądź przetworzenia n wierzchołków, gdzie n jest rządem grafu.

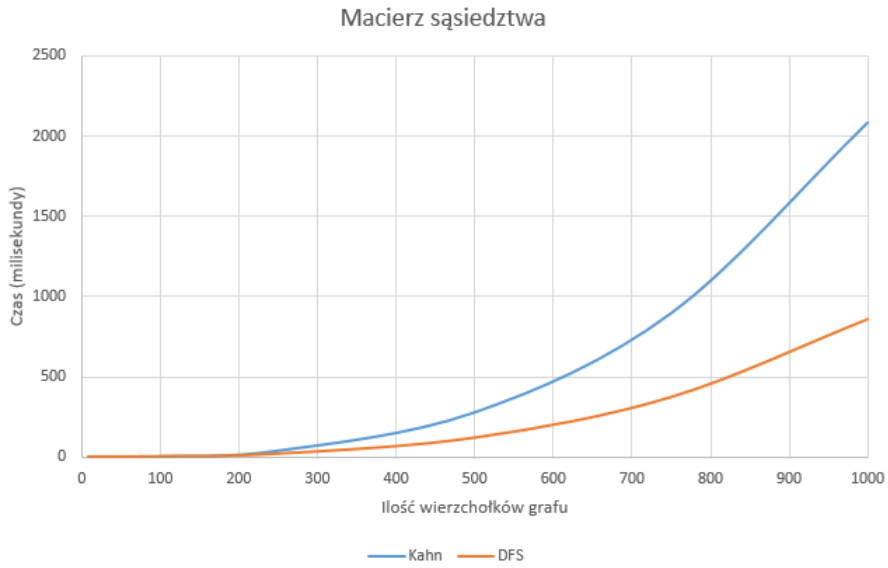
4.2 Sortowanie topologiczne z wykorzystaniem DFS

W tym sortowaniu wykorzystuje się algorytm DFS czyli przeszukiwania w głąb. Jest algorytm jest bardzo podobny do standardowego DFS'a z wyjątkiem kroku na którym zapisuje się wierzchołek przetworzony. W standardowym DFS'ie wierzchołki spisuje się od razu po wejściu do nich, za to w wersji sortującej graf - przed wyjściem z nich. Otrzymane w ten sposób uporządkowanie należy wypisać od tyłu.

5 Zestawienie czasów wykonywania

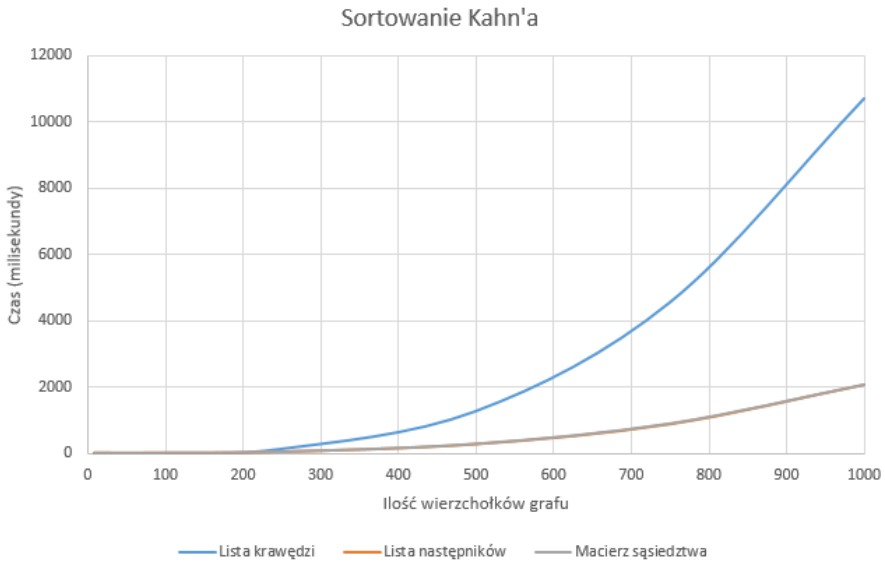
Niżej znajdują się wykresy czasów wykonywania algorytmów sortowania w zależności od rodzaju reprezentacji grafu oraz jego rzędu.

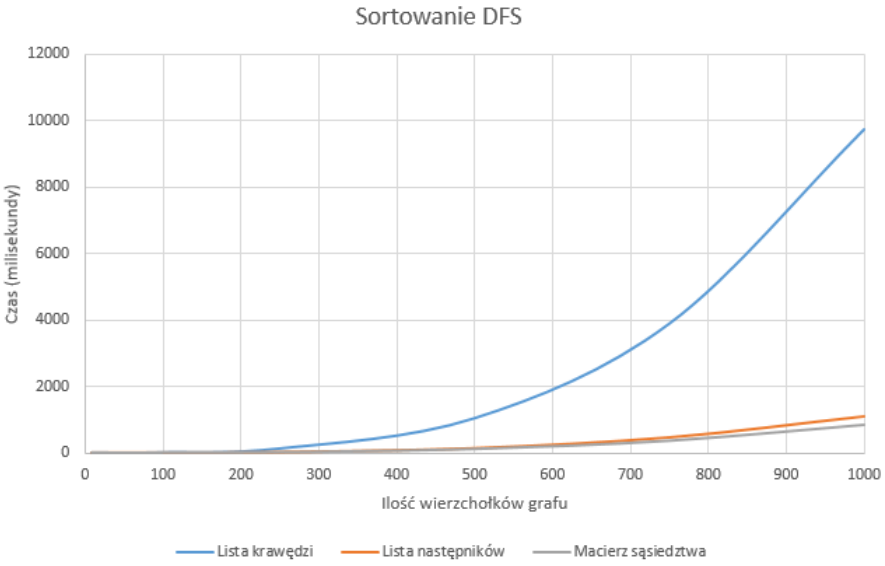
5.1 Sortowania w danych reprezentacjach



Na powyższych wykresach można zauważyć różnicę czasu potrzebnego na wykonanie się sortowania metodą Kahn'a oraz z wykorzystaniem DFS'a. Sortowanie Kahn'a jest sporo wolniejsze od alternatywy. Wyjątkiem jest przypadek listy krawędzi, gdzie wyszukanie wszystkich następników węzła jest bardzo wolne przez co oby dwa algorytmy wydają się działać w podobnym czasie.

5.2 Reprezentacje w danych sortowaniach





Z na powyższych wykresach z łatwością można zauważyć nieefektywność listy krawędzi. Powolne wyszukiwanie następników danego wierzchołka sprawia, że ta reprezentacja jest nieużywana w tych celach. Macierz sąsiedztwa oraz lista następników działają w podobnych czasach, lecz przy grafach rzadkich można by było zauważyć zwiększoną prędkość listy następników. Fakt, że oprócz wyszukiwania następników węzła należy przez nie przeiterować sprawia, że szybkość tej operacji jest prawie nieistotna.

6 Podsumowanie

Zostały przedstawione i omówione trzy reprezentacje grafu, wraz z analizą szybkości działania każdej z nich. Ponadto, zostały przybliżone dwa algorytmy sortowania topologicznego, dla których również zostało sporządzone zestawienie szybkości działania. Przy okazji, poruszona została kwestia losowania grafów.

Spis treści

1 Wprowadzenie **1**

2 Reprezentacje grafu **1**
2.1 Lista krawędzi 1
2.2 Lista następników 1
2.3 Macierz sąsiedztwa 2

3 Generowanie losowych instancji grafowych **2**

4 Algorytmy sortowania topologicznego **2**
4.1 Sortowanie topologiczne Kahn'a 2
4.2 Sortowanie topologiczne z wykorzystaniem DFS 3

5 Zestawienie czasów wykonywania **3**
5.1 Sortowania w danych reprezentacjach 4
5.2 Reprezentacje w danych sortowaniach 6

6 Podsumowanie **7**