Dokumentacja końcowa

Przedmiot: Analiza algorytmów
Temat: 13 (Plan studiów)
Autor: Adrian Nadratowski (293 163)

1) **Sprecyzowanie problemu**
   Tematem projektu jest opracowanie metody wygenerowania planu studiów (sekwencji przedmiotów w odpowiedniej kolejności) z zachowaniem podanych warunków, czyli zależności między przedmiotami (jeśli w danych wejściowych występuje linia "*a b*" - oznacza to, że aby przystąpić do przedmiotu *b*, należy uprzednio zaliczyć przedmiot *a*)
2) **Przyjęte założenia**
   a) Dane wejściowe są postaci:
      > ${liczba_przedmiotów}
      > *przedmiotA przedmiotB*
      > .
      > .
      > .
      > *przedmiotX przedmiotY*
   b) Grafy reprezentujące zależności między przedmiotami generowane są w dwóch trybach: gęsty (maksymalna liczba krawędzi w grafie nieskierowanym - |V| * (|V| - 1) / 2), rzadki (liczba krawędzi w grafie równa liczbie wierzchołków), natomiast interfejs generatora grafów pozwala na łatwą modyfikację programu w celu generowania grafów o dowolnej złożoności.
   c) Algorytm nie gwarantuje identycznego wyniku w przypadku dwóch zestawów danych identycznych pod względem zawartości, ale różniących się kolejnością podawanych par przedmiotów
   d) Przy pomiarach czasu wykonania algorytmu pominięte zostały kroki budowy grafu. Ten proces został wykonany przed rozpoczęciem pomiaru
3) **Metoda generowania losowych grafów**
   W celu przetestowania złożoności algorytmu należało opracować metodę generowania grafów losowych o zadanej liczbie krawędzi i rozmiarze. Metoda tworzenia losowego grafu nieskierowanego została oparta o wykorzystanie macierzy sąsiedztwa. Dla grafów nieskierowanych, macierz sąsiedztwa ma elementy o wartości 1 wpisane tylko w dolnym trójkącie macierzy (z wyłączeniem głównej przekątnej). Pozostałe elementy mają wartość 0. Jeśli w *i*-tym wierszu, w *j*-tej kolumnie, występuje element o wartości 1, oznacza to, że w grafie występuje krawędź z wierzchołka o numerze *j* do wierzchołka o numerze *i*.
4) **Opis rozwiązania, reprezentacji danych, wybranych struktur danych i używanych terminów**
   a) **Terminologia**
      > *indegree* - stopień wierzchołka określający ile krawędzi kończy się w danym wierzchołku

> *plan zajęć* - uporządkowana lista przedmiotów w kolejności, w jakiej powinien się na nie zapisywać
> *korzeń* - wierzchołek o indegree = 0 (reprezentuje przedmiot, dla którego nie ma już poprzedników, tzn. wszyscy poprzednicy zostali wpisani do planu zajęć)

b) **Reprezentacja danych**
Graf (skierowany) -> przedstawienie zależności między przedmiotami
Wierzchołek -> przedmiot
Krawędź -> zależność między przedmiotami. Wierzchołek, z którego wychodzi krawędź reprezentuje przedmiot-poprzenik dla przedmiotu reprezentowanego przez wierzchołek, do którego krawędź jest skierowana

c) **Użyte struktury danych**
Do przechowywania wierzchołków w grafie wykorzystywana jest struktura z języka C++ *std::unordered_map*. Taki wybór zapewnia optymalny czas wstawiania oraz usuwania z grafu wierzchołków.

d) **Opis rozwiązania**
Rozwiązanie bazuje na podziale wierzchołków na korzenie i wierzchołki zwykłe. Dopóki w grafie występują wierzchołki i problem da się rozwiązać (w grafie nie występują cykle), pobierany jest pierwszy wierzchołek z mapy korzeni, następnie dla każdego z jego sąsiadów indegree jest dekrementowane i jeśli któryś z nich stanie się wierzchołkiem reprezentującym niezależny przedmiot w danej chwili, dany sąsiad jest wpisywany do mapy korzeni. Ostatecznie, usuwa się przetwarzany w danej iteracji wierzchołek.

5) Ocena złożoności algorytmu

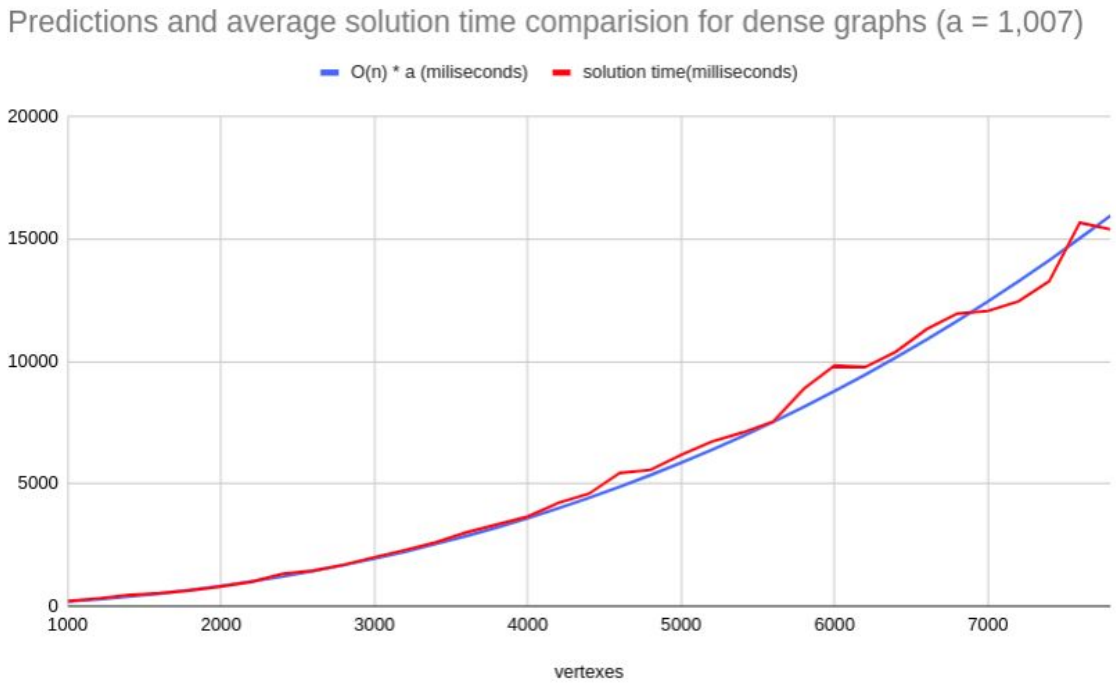
Algorytm działa dopóki w grafie są wierzchołki (o ile nie zostanie wcześniej przerwany przez wykryty w grafie cykl). Główna pętla algorytmu wykonuje się więc |V| razy. W każdej iteracji, z mapy korzeni pobierany jest pierwszy element (złożoność O(1)), następnie, dla każdego sąsiada, dekrementowane jest *indegree* (złożoność O(1) * *liczba_sąsiadów*). Jeśli któryś z sąsiadów stanie się korzeniem, należy dodac go do struktury *unordered_map* przechowującej korzenie (warto zauważyć, że każdy sąsiad zostanie dodany tylko raz, więc operacji wstawiania do mapy korzeni w całym algorytmie będzie dokładnie |V|. Koszt operacji wstawiania jest, w przypadku struktury *unordered_map,* średnio stały, natomiast w najgorszym przypadku - liniowy w stosunku do rozmiaru mapy). Na końcu każdej iteracji, z mapy korzeni oraz z mapy wierzchołków usuwany jest przetwarzany korzeń. Taka operacja ma, podobnie jak operacja wstawiania do struktury *unordered_map*, średni koszt O(1), natomiast w najgorszym przypadku O(n), gdzie n to rozmiar mapy. Na koniec, należy zaznaczyć, że sumaryczna liczba sąsiadów dla każdego wierzchołka w grafie, jest równa liczbie krawędzi w nim występujących.

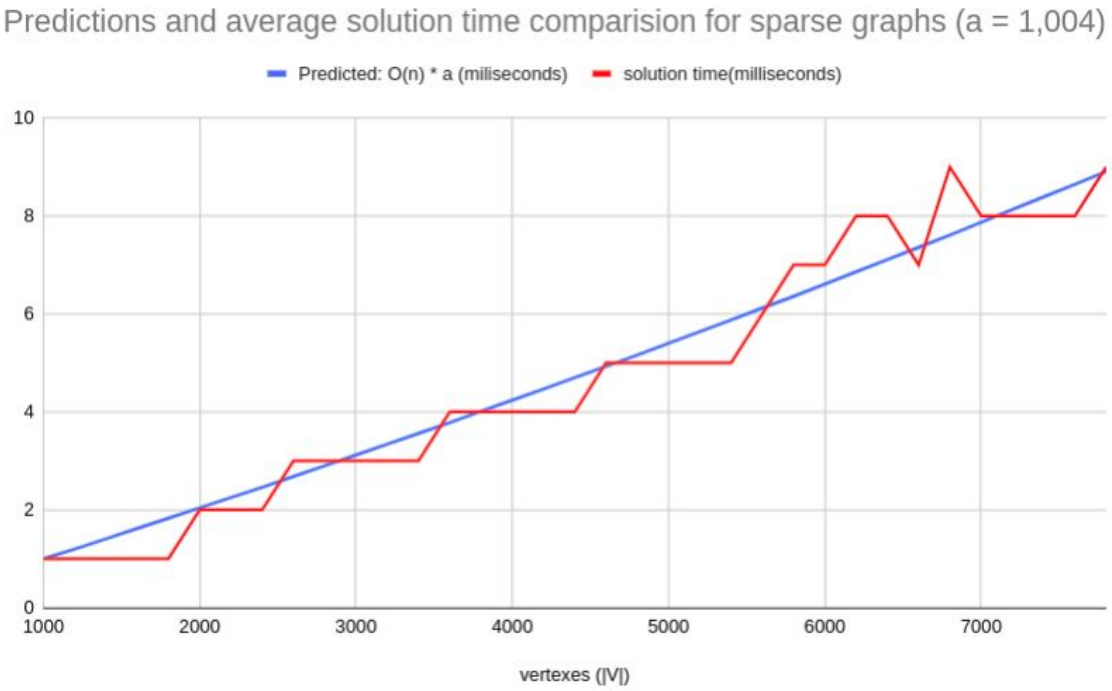
Słowem podsumowania, średni koszt algorytmu wyniesie O(|V|, |E|) = |V| * 3 + |E| * 5 + |V| * 1. Pierwszy człon dotyczy pobrania przetwarzanego korzenia i usunięcia go z dwóch map (3 operacje dla każdego przetwarzanego wierzchołka, 2 z nich w najgorszym przypadku liniowo zależne od rozmiaru mapy), drugi dotyczy procesu usunięcia z mapy sąsiadów przetwarzanego korzenia wszystkich występujących w niej wierzchołków (5 operacji, w tym 2, które w najgorszym przypadku są liniowo zależne od rozmiaru mapy), natomiast trzeci - wstawienia wierzchołka do mapy

korzeni. Ta operacja wykona się dokładnie raz dla każdego wierzchołka.

Można więc przyjąć złożoność jako O(|V|, |E|) = |V| + |E|.
Jeśli uprościmy formułę i przyjmiemy |V| = n, to dla grafów rzadkich, gdzie |E| = |V|, otrzymamy:
O(n) = 9*n (liniowa zależność od liczby wierzchołków)
a dla grafów gęstych (w przypadku tego zadania grafy z maksymalną liczbą krawędzi), gdzie |E| = |V| * (|V| - 1)/2, otrzymamy:
O(n) = 4*n + [5 * n * (n -1)] / 2 (kwadratowa zależność od liczby wierzchołków)

6) **Porównanie przewidywanej złożoności z otrzymanymi wynikami**
Na potrzeby zestawienia działania algorytmu z przewidywanymi rezultatami, wykonane zostały dwa testy (oddzielnie dla grafów rzadkich i gęstych) z takimi samymi parametrami. Każdy test polegał na wygenerowaniu po 10 instancji problemu danej wielkości (inicjalnie 1000 wierzchołków), utworzenia losowego grafu, zmierzenia czasu wykonania algorytmu i zwiększeniu rozmiaru problemu o zadany krok (200 wierzchołków). Generowane były instancje dla grafów o wielkościach 1000 - 7800 wierzchołków. Czasy wykonania zostały uśrednione i zestawione w tabelach w pliku *AAL.xslx*. Poniżej zestawienie przeskalowanych funkcji kwadratowej (dla grafów gęstych) i funkcji liniowej (dla grafów rzadkich):





Oszacowanie dla grafów gęstych wygląda obiecująco. Wykres dla grafów rzadkich wydaje się być bardzo chaotyczny, jednak oscyluje wokół pewnej funkcji liniowej. Fakt, że wykres wygląda chaotycznie wynika z bardzo małej liczby danych przetwarzanych przez algorytm (od 2000 do 15600 wierzchołków i krawędzi). Aby realnie sprawdzić oszacowanie dla grafów rzadkich, należałoby wygenerować bardzo duże grafy. Dla obu typów grafu jakość oszacowania *q(n)* oscyluje w granicach 1.