Projekt nr 16

Porównanie implementacji algorytmu Forda Fulkersona w językach programowania C/C++ i C#

Autorzy:

Petrykowski Daniel

Wojciechowski Grzegorz

Sprawozdanie nr 1

1. Cel projektu

Celem projektu jest zaimplementowanie algorytmu Forda Fulkersona do wyznaczania ścieżki pozwalającej na maksymalny przepływu w sieci przepływowej. Zostanie on zaimplementowany w C/C++ i C#, a następnie zostanie zbadany czas trwania obliczeń dla tych dwóch implementacji przy takich samych parametrach wejściowych.

Sprawozdanie nr 2

1. Opis algorytmu Forda Fulkersona

Algorytm Forda Fulkersona pozwala na wyznaczenie maksymalnego przepływu w sieci przepływowej. Jednak, że przed przedstawieniem zasady działania algorytmu należy wprowadzić potrzebne pojęcia:

1. sieć przepływowa – jest grafem skierowanym G = (V, E) w którym przepływ odbywa się w kierunku wyznaczonym przez zwrot krawędzi grafu, każda krawędź jest skojarzona z parametrem określającym jej przepustowość.
2. przepustowość - jest definiowana jako funkcja c(u,v), gdzie u i v są wierzchołkami. Jeżeli pomiędzy dwoma wierzchołkami istnieje krawędź to wartość przepustowość będzie wynosić c(u,v) >= 0, jeżeli taka krawędź nie istnie to wartość przepustowość będzie równa c(u,v) = 0.
3. źródło - wierzchołek s, który jest źródłem z którego zaczyna się przepływ
4. ujście - wierzchołek t będący ujściem przepływu
5. przepływ f(u,v) - określa jaki przepływ odbywa się w kanale. Przy czym przepływ pomiędzy dwoma węzłami sieci nie może być większy niż wartość przepustowości pomiędzy nimi.
6. maksymalna przepustowość sieci pomiędzy źródłem i ujściem - jest ona określona przez funkcje:

obrazek

Czyli suma przepływów ze źródła s do wszystkich pozostałych wierzchołków sieci. (1)

1. sieć rezydualna – jest grafem indukowanym z pierwotnej sieci przepływu. Nowo powstała sieć rezydualna posiada tą samą liczbę wierzchołków i krawędzie określające przepustowość rezydualną pomiędzy nimi.
2. przepustowość rezydualna - jest to różnica przepustowości i przepływu w sieci przepływowej. (Jakiś rysunek) (1)

*cf*(*u,v*) = *c*(*u,v*) - *f*(*u,v*)

1. ścieżka rozszerzająca – ścieżka w sieci rezydualnej łącząca źródło z ujściem, przy czym wszystkie kanały leżące na ścieżce muszą mieć wartości przepustowości rezydualnej nie zerową.
2. przepustowość rezydualna – jest to wartość równa najmniejszej przepustowości rezydualnej kanałów leżących na ścieżce rozszerzającej

*cf*(*p*) = min{*cf*(*u,v*): (*u,v*) obrazek *p*}

Idea algorytmu opiera się na iteracyjnym zwiększaniu przepływu, zaczynając od zerowego przepływu. Podstawowa wersja brzmi następująco:

1. Wyzeruj wszystkie przepływy w sieci
2. Wyznacz sieć rezydualną dla wejściowej sieci przepływów
3. Dopóki w sieci istnieje ścieżka rozszerzająca p, zwiększ przepływ o przepustowość rezydualną znalezionej ścieżki *cf* wzdłuż kanałów zgodnych z kierunkiem ścieżki, a zmniejsz przepływ wzdłuż kanałów przeciwnych

Jak wynika z powyższego opisu algorytm Forda Fulkersona jest bardziej metodą postepowania, niż szczegółowym opisem implementacji. Dlatego też, aby go wykorzystać należy również rozwiązać problem dotyczy sposobu w jaki sieć będzie reprezentowana w pamięci komputera oraz drugi problem jakim jest wyszukiwanie ścieżek rozszerzających w sieci rezydualnej.

Sieć będzie reprezentowana w postaci listy sąsiedztwa. Jest to efektywny pamięciowo sposób reprezentacji grafu, zajmuje pamięć rzędu O(m), gdzie m oznacza liczbę krawędzi grafu.

W porównaniu do macierzy sąsiedztwa sprawdzenie czy dana krawędź istnieje będzie wolniejsze, ale pozwoli za to na operowanie na większych sieciach.

Zostanie stworzona tablica o rozmiarze równym liczbie wierzchołków, zawierająca wskaźniki na listy – kolejne elementy list będą oznaczać kolejnych sąsiadów danego wierzchołka, do którego lista jest przyporządkowana.

Sąsiad będzie opisywany obiektem, który przechowywał będzie informacje o numerze wierzchołka, maksymalnym przepływie i przepływie bieżącym.

Do wyszukiwania ścieżek zostanie zaimplementowane przeszukiwanie wszerz (breadth-first search, BFS). Jest to popularne rozwiązanie pozwalające odnaleźć wszystkie połączone węzły w grafie. Kolejnym plusem jest to, że algorytm ten zawsze będzie wybierał ścieżkę o najmniejszej liczbie krawędzi.

Algorytm zaczyna się od odwiedzenia wierzchołka startowego. Następnie odwiedza się wszystkich jego sąsiadów, a potem wszystkich nieodwiedzonych sąsiadów sąsiadów i tak iteracyjnie. Aby uniknąć zapętleń w przypadku napotkania cyklu do wierzchołka dodaje się parametr mówiący o tym, czy został on już odwiedzony.

To jeszcze nie wszystko będzie tu jeszcze trzeba to bardziej rozwinąć.

1. Projekt testów
2. Założenia programu

Sprawozdanie nr 3

1. Dokumentacja kodu
2. Wyniki testów
3. Wnioski

# Bibliografia

1. **Zaawansowane algorytmy i struktury danych/Wykład 9 - Wazniak.mimuw. [Online] 2009. [Zacytowano: 14 Listopad 2019.] http://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane\_algorytmy\_i\_struktury\_danych/Wyk%C5%82ad\_9**