**PROJEKTOWANIE ALGORYTMÓW i METOD SZTUCZNEJ INTELIGENCJI**

Dawid Sielużycki 241508

1. **Cele zadania**

Zadanie polegało na zaimplementowaniu grafu w postaci listy i macierzy sąsiedztwa. Dodatkowo trzeba było wyposażyć kod w algorytmy: Dijkstry i Bellmana-Forda. Oba algorytmy służą do wyliczania najkrótszej drogi w grafie między wierzchołkiem startowym a pozostałymi wierzchołkami. Badania były przeprowadzone dla następujących gęstości grafów: 25,50,75 i 100%. Wierzchołki zostały dobrane dla obu algorytmów inne co wyjaśnię poniżej.

1. **Algorytmy**
   1. Dijkstra

Algorytm Dijkstry znajduje w grafie wszystkie najkrótsze ścieżki pomiędzy wybranym wierzchołkiem a wszystkimi pozostałymi, przy okazji wyliczając również koszt przejścia każdej z tych ścieżek.

Złożoność obliczeniowa algorytmu Dijkstry zależy od **V** wierzchołków **E** krawędzi grafu oraz sposobu implementacji kolejki priorytetowej. Wykorzystując „naiwną” implementację poprzez zwykłą tablicę, otrzymujemy algorytm o złożoności **.**

W implementacji kolejki poprzez kopiec, złożoność wynosi **O(E \* log V).**

Po zastąpieniu zwykłego kopca kopcem Fibonacciego złożoność zmienia się **O(E+V \* log V).**

Pierwszy wariant jest optymalny dla grafów gęstych, drugi jest szybszy dla grafów rzadkich, trzeci jest bardzo rzadko używany ze względu na duży stopień skomplikowania i niewielki w porównaniu z nim zysk czasowy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Macierz** | **Czas [s]** | | | |
| **Ilość wierzch.** | **25%** | **50%** | **75%** | **100%** |
| **100** | 0,00095 | 0,00145 | 0,001 | 0,0008 |
| **500** | 0,0249 | 0,026 | 0,0221 | 0,0243 |
| **1000** | 0,1105 | 0,08875 | 0,08855 | 0,09765 |
| **2000** | 0,4098 | 0,3754 | 0,3563 | 0,3608 |
| **3000** | 0,9032 | 0,70645 | 0,83615 | 0,857 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lista** | **Czas [s]** | | | |
| **Ilość wierzch.** | **25%** | **50%** | **75%** | **100%** |
| **100** | 0,00015 | 0,00155 | 0,0003 | 0,0002 |
| **500** | 0,00375 | 0,0055 | 0,0087 | 0,00445 |
| **1000** | 0,01525 | 0,0249 | 0,03445 | 0,0157 |
| **2000** | 0,05685 | 0,10435 | 0,15755 | 0,06285 |
| **3000** | 0,1333 | 0,24905 | 0,38 | 0,14115 |

Porównanie Macierzy i Listy

* 1. Bellman- Ford

W odróżnieniu od algorytmu Dijkstry, algorytm Bellmana-Forda działa poprawnie także dla grafów z wagami ujemnymi (nie może jednak wystąpić cykl o łącznej ujemnej wadze osiągalny ze źródła). Za tę ogólność płaci się jednak wyższą złożonością czasową. Działa on w czasie **O\*(|V|\*|E|), gdzie E=(gęstość[%]\*V\*(V-1))/200**. Jest to ilość krawędzi.

Algorytm może być wykorzystywany także do sprawdzania, czy w grafie występują ujemne cykle osiągalne ze źródła.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Macierz** | **Czas [s]** | | | |
| **Ilość wierzch.** | **25%** | **50%** | **75%** | **100%** |
| **10** | 0,00005 | 0,0001 | 0,00005 | 0,00005 |
| **50** | 0,0041 | 0,0044 | 0,00415 | 0,0038 |
| **100** | 0,02975 | 0,0254 | 0,02395 | 0,02455 |
| **500** | 2,85415 | 3,0585 | 2,93165 | 2,78005 |
| **1000** | 22,6157 | 24,2745 | 23,5552 | 22,8643 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lista** | **Czas [s]** | | | |
| **Ilość wierzch.** | **25%** | **50%** | **75%** | **100%** |
| **10** | 0,0001 | 0,0001 | 0,0001 | 0,0001 |
| **50** | 0,0001 | 0,0003 | 0,0003 | 0,0005 |
| **100** | 0,00125 | 0,0026 | 0,0046 | 0,0045 |
| **500** | 0,378 | 2,09525 | 3,77785 | 1,63685 |
| **1000** | 10,3715 | 22,0299 | 34,0686 | 13,3169 |

Porównanie Macierzy i Listy

Specyfikacja komputera wykorzystanego do przeprowadzenia testów:   
- Intel core i7-7750U 2,90GHz,  
 - 8 GB RAM,  
 - 64 bitowy system operacyjny Windows 10 Educational Edition.  
 W trakcie przeprowadzenia testów jedynym programem uruchomionym było Visual Studio.

1. **Wnioski**

W algorytmie Dijkstry graf skierowany na macierzy trwa dłużej, choć zużywa mniej pamięci niż graf na liście. Graf utworzony na liście sąsiedztwa powoduje przecieki pamięci, których pochodzenia nie jestem w stanie znaleźć. Mimo usuwania list w tablicy, samej tablicy oraz tablicy pomocniczej zawierającej flagi wciąż zużycie pamięci przez program w przypadku listy rośnie, nie jest to jednak przeciek na tyle duży aby w dobranej ilości wierzchołków spowodować problemy podczas testowania.   
W algorytmie Bellmana- Forda użyłem mniejszej ilości wierzchołków z powodu dłuższych czasów jego działania (czas oczekiwania na stworzenie 100 instancji przy ilości 3k wierzchołków byłby zbyt długi). Tylko w przypadku gęstości 75% lista trwa o 10s dłużej niż macierz. Może być to spowodowane niezwykłym szczęściem w losowym tworzeniu struktury. Pomiary czasu były wykonane na podstawie średniej arytmetycznej 100 instancji każdej ze struktur.  
Ogólnie można stwierdzić, że graf na listach jest szybszy niż na macierzach. Podsumowując można stwierdzić, że algorytm Bellmana- Forda wypada dość kiepsko. Jego największą zaletą jest możliwość rozpatrzenia ujemnych wag połączeń jednakże chcąc zastanowić się nad praktycznym wykorzystaniem algorytmu, którym może być np szukanie najkrótszej drogi na mapie (podobnie jak w Google Maps), gdzie ilość skrzyżowań (wierzchołków grafu) będzie znacznie większa niż testowane 1000, to jest on po prostu zbyt wolny. Jego złożoność obliczeniowa wprost sugeruje jak horrendalne będą to czasy dla chociażby 10 000 wierzchołków, a przecież czasem potrzebujemy ich nawet więcej.  
Czas działania algorytmu Dijkstry w obu przypadkach są wyjątkowo szybkie i pomimo wątpliwości ich poprawności narysowane wykresy złożoności obliczeniowych są zbliżone do oczekiwanych ( O(E \* log V) – Liczba krawędzi E drastycznie wzrasta w zależności od ilości wierzchołków natomiast logV wraz ze wzrostem ilości wierzchołków wygładza się.)

Źródła:

<https://pl.wikipedia.org/wiki/Algorytm_Dijkstry>

<https://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda>

<https://eduinf.waw.pl/inf/alg/001_search/0124.php>

<https://eduinf.waw.pl/inf/alg/001_search/0138.php>

<http://davinci.mimuw.edu.pl/matematyka/gestosc-grafu>

<https://pl.wikipedia.org/wiki/Graf_(matematyka)>

<https://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda>

<http://home.agh.edu.pl/~horzyk/lectures/wdi/WDI-Grafy.pdf>