10.05.2019r.

Magdalena Sałęga 241539

Dane prowadzących:

Wykład: Łukasz Jeleń, śr.17.05

Projekt: Krzysztof Halawa, wt.17.05

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

**Sprawozdanie**

Projekt 2

**Algorytm Dijkstry  
znajdowania najkrótszej ścieżki w grafie**

1. **Wstęp**

Projekt 2 polega na zaimplementowaniu i sprawdzeniu złożoności obliczeniowej algorytmu, w tym przypadku algorytmu Dijkstry. Zaimplementowane zostały dwa sposoby reprezentacji grafu czyli macierz i lista sąsiedztwa. Złożoność tych algorytmów była sprawdzana na 5 wielkościach grafów (10, 50, 100, 500, 1000 wierzchołków) oraz w każdym z tych przypadków cztery gęstości grafu (25%, 50%, 75%, 100%). Każda z podanych opcji została zbadana w 100 powtórzeniach, sprawozdanie zaś zawiera jedynie wartości średnie. Program został napisany obiektowo.  
  
Graf – zbiór wierzchołków, które mogą być połączone krawędziami w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków.  
Graf pełny to taki, w którym każdy wierzchołek ma przynajmniej jedną krawędź. Krawędź może posiadać wagę, czyli liczbę która określa na przykład odległość między wierzchołkami.

1. **Opis badanych algorytmów, określenie ich złożoności w przypadkach średnim i najgorszym**

Złożoność obliczeniowa zależna jest od liczby elementów.

* 1. **Algorytm Dijkstry**

Służy do znajdowania najkrótszej ścieżki z danego wierzchołka w grafie o nieujemnych wagach krawędzi. Algorytm znajduje odległość od danego wierzchołka do wszystkich pozostałych w grafie. W projekcie zaimplementowana również została funkcja znajdująca najkrótsza drogę między dwoma danymi wierzchołkami.  
Złożoność algorytmu zależy od sposobu implementacji programu.

* 1. **Algorytm Dijkstry w reprezentacji grafu jako „macierz sąsiedztwa”**

Graf reprezentowany jest jako macierz o wielkości *N* x *N,* gdzie *N* jest ilością wierzchołków w grafie. Kolejne indeksy odzwierciedlają numery wierzchołków, zaś pola macierzy wskazują na ścieżki łączące te wierzchołki.  
Wymagania pamięciowe w notacji dużego O dla algorytmu Dijkstry w reprezentacji grafu jako „macierz sąsiedztwa” jest równa: O().

* 1. **Algorytm Dijkstry w reprezentacji grafu jako „lista sąsiedztwa”**

Algorytm ten wymaga uprzedniej implementacji struktury danych, którą jest lista jednokierunkowa. Wymagane jest utworzenie tablicy o rozmiarze *N,* której elementami będą listy ścieżek, które wychodzą z danego wierzchołka opisanego indeksem w tablicy. Wymagane jest również zawarcie informacji jaki jest wierzchołek końcowy danej ścieżki.  
Wymagania pamięciowe w notacji dużego O dla algorytmu Dijkstry w reprezentacji grafu jako „lista sąsiedztwa” jest równa: O(*K*), *K* to liczba krawędzi.

1. **Przebieg eksperymentu**
2. Zaimplementowanie listy, klas Wierzcholek, Sciezka i Graf dla obu algorytmów (macierz i lista) oraz potrzebnych funkcji do działania programu.
3. Zaimplementowanie po kolei każdego algorytmu wraz z dodatkowymi algorytmami znajdowania ścieżki między dwoma danymi wierzchołkami.
4. Utworzenie funkcji obsługującej wywoływanie algorytmów z odpowiednimi parametrami:

- ilość wierzchołków w grafie (10, 50, 100, 500, 1000)  
- gęstość grafu (25%, 50%, 75%, 100%)  
- wywołanie wszystkich opcji powtarzając i zapisując wyniki po 100 razy

1. Utworzenie funkcji wczytującej graf z pliku tekstowego.
2. Ręczne sprawdzenie poprawności algorytmów poprzez sprawdzanie grafów o maksymalnie 10 wierzchołkach i różnych wielkościach
3. Wywołanie funkcji w stu powtórzeniach dla każdego opisanego wyżej przypadku
4. Utworzenie zmiennych do zmierzenia czasu potrzebnego na wykonanie się konkretnych przebiegów funkcji
5. Utworzenie funkcji zapisującej do pliku tekstowego wszystkie zmierzone czasy
6. Wczytanie pliku tekstowego do Excel’a, odwrócenie całości i wykonanie odpowiednich tabel oraz wykresów.
7. **Przedstawienie wyników**Porównanie średnich czasów działania dla obu algorytmów w zależności od ilości wierzchołków w grafie i gęstości grafu.  
   1. **Algorytm Dijkstry w reprezentacji grafu jako „macierz sąsiedztwa”**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 25% | 50% | 75% | 100% |
| 10 | 0,000000 | 0,000000 | 0,000000 | 0,000000 |
| 50 | 0,000010 | 0,000196 | 0,000010 | 0,000020 |
| 100 | 0,000050 | 0,000070 | 0,000197 | 0,000237 |
| 500 | 0,005028 | 0,005604 | 0,005458 | 0,006130 |
| 1000 | 0,023529 | 0,023219 | 0,023812 | 0,023981 |

* 1. **Algorytm Dijkstry w reprezentacji grafu jako „lista sąsiedztwa”**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 25% | 50% | 75% | 100% |
| 10 | 0,000010 | 0,000000 | 0,000000 | 0,000000 |
| 50 | 0,000000 | 0,000010 | 0,000323 | 0,000030 |
| 100 | 0,000030 | 0,000196 | 0,000080 | 0,000256 |
| 500 | 0,003238 | 0,005744 | 0,008936 | 0,011724 |
| 1000 | 0,016657 | 0,051654 | 0,079370 | 0,093500 |

1. **Wykresy utworzone na podstawie średnich wartości otrzymanych wyników**
   1. **Ukazanie różnicy w czasie wykonania obu algorytmów szukających najkrótszej ścieżki w zależności gęstości grafu.**
   2. **Ukazanie różnicy w czasie wykonania obu algorytmów szukających najkrótszej ścieżki.**
2. **Wnioski**

Wszystkie wyniki czasów trwania podane zostały w sekundach.

Zmierzone czasy trwania obu przedstawionych algorytmów Dijkstry są zgodne z założeniami teoretycznymi. Algorytm Dijkstry z grafem przedstawionym w postaci listy jest wydajniejszy dla małych wielkości grafów, zaś algorytm w z grafem w wersji jest wydajniejszy dla większych ilości wierzchołków w grafie.

Czas wykonywania programu, zgodnie z przewidywaniami, zwiększał się wraz z coraz to większą ilością wierzchołków oraz większą gęstością grafu.

Rozbieżności w pomiarach oraz czasy równe „0” dla małych wielkości grafów spowodowane są niedokładnością pomiarową.

1. **Bibliografia**

Projekt2.pdf – opis projektu oraz algorytmów

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

<https://pl.wikipedia.org/wiki/Reprezentacja_grafu?fbclid=IwAR27StRx8tA3KxRvhK9UunRUJ8xEGMQf3rmxs598F6CgqIwRaZAu7AvZufY>

<http://www.algorytm.org/algorytmy-grafowe/algorytm-dijkstry.html?fbclid=IwAR0jK1VIhHxuxk3qBf43yPzI9nmDxEREAc7KnsKUbtq7O4Yonndry_fmYd8>

<https://eduinf.waw.pl/inf/alg/001_search/0124.php?fbclid=IwAR3OZoxKfz_4xtqdV_M0jZn70Y0Eip5zSA08jLPKrkklTuzcb-ojebPuG5Y>