Adrian Frydmański 209865 Pon 13:15

sdizo – Projekt II

# Wstęp teoretyczny

Tematem tego projektu są 4 algorytmy operujące na grafach. Pierwsze dwa, Prima i Kruskala dotyczą wyznaczania minimalnego drzewa rozpinającego. Mają one złożoność obliczeniową O(ElogV). Kolejne algorytmy wyznaczały najkrótszą ścieżkę w grafie. Były to: algorytm Dijkstry (złożoność O(ElogV))   
i Forda-Bellmana (złożoność O(|V|\*|E|).

# Plan eksperymentu

* Implementacja programu operującego grafach w postaci listowej i macierzowej
* Automatyczne testy dla różnej liczby elementów i różnej gęstości wraz z pomiarem czasu wykonywanych poszczególnych operacji
* Pomiar czasu wykonany według zasugerowanego w instrukcji sposobu, polegający na zliczaniu cykli zegara i dzieleniu wyniku przez częstotliwość taktowania procesora.

# Wyniki

Testy przeprowadzono dla 10, 20, 30, 40 i 50 elementowych grafów, z zagęszczeniem 25, 50, 75 i 99%, przy użyciu generowanych automatycznie, pseudolosowych danych. Test można powtórzyć wybierając w menu głównym [8].

## Wykresy dla reprezentacji

### Wyznaczanie najmniejszego drzewa rozpinającego (MST)

Tabela 1 – algorytm Prima i Kruskala dla reprezentacji listowej przy różnych gęstościach i liczbach elementów  


Wykres 1 - algorytm Prima i Kruskala dla reprezentacji listowej przy różnych gęstościach i liczbach elementów

Tabela 2 – algorytm Prima i Kruskala dla reprezentacji macierzowej przy różnych gęstościach i liczbach elementów  


Wykres 2 - algorytm Prima dla reprezentacji macierzowej przy różnych gęstościach i liczbach elementów

### Wyznaczanie najkrótszej ścieżki

Tabela 3 – algorytm Dijkstry i Forda-Bellmana dla reprezentacji listowej przy różnych gęstościach i liczbach elementów  


Wykres 3 - algorytm Dijkstry dla reprezentacji listowej przy różnych gęstościach i liczbach elementów

Tabela 4 – algorytm Dijkstry i Forda-Bellmana dla reprezentacji macierzowej przy różnych gęstościach i liczbach elementów  


Wykres 4 - algorytm Dijkstry dla reprezentacji macierzowej przy różnych gęstościach i liczbach elementów

## Wykresy dla poszczególnych gęstości

### Wyznaczanie najmniejszego drzewa rozpinającego (MST)

Tabela 5 – algorytmy dla 25% zagęszczenia grafu  


Wykres 5 – algorytmy dla 25% zagęszczenia grafu

Tabela 6 – algorytmy dla 50% zagęszczenia grafu   


Wykres 6 – algorytmy dla 25% zagęszczenia grafu

Tabela 7 – algorytmy dla 75% zagęszczenia grafu   


Wykres 7 – algorytmy dla 25% zagęszczenia grafu

Tabela 8 – algorytmy dla 99% zagęszczenia grafu   


Wykres 8 – algorytmy dla 25% zagęszczenia grafu

### Wyznaczanie najkrótszej ścieżki

Tabela 9 – algorytmy dla 25% zagęszczenia grafu  


Wykres 9 – algorytmy dla 25% zagęszczenia grafu

Tabela 10 – algorytmy dla 50% zagęszczenia grafu   


Wykres 10 – algorytmy dla 50% zagęszczenia grafu

Tabela 11 – algorytmy dla 75% zagęszczenia grafu   


Wykres 11 – algorytmy dla 75% zagęszczenia grafu

Tabela 12 – algorytmy dla 99% zagęszczenia grafu   


Wykres 12 – algorytmy dla 99% zagęszczenia grafu

# Wnioski

Zauważyć możemy lepszą efektywność algorytmu Kruskala, bo na przykład dla zagęszczenia 99% wykazuje podobne czasy, jak Prim dla 25%, przy czym ich wzrost wraz ze wzrostem liczby elementów ma taki sam, logarytmiczny charakter.   
Przy szukaniu najkrótszej ścieżki algorytm Dijkstry radzi sobie lepiej, niż Forda-Bellmana.   
Już przy 25% zagęszczeniu czas operacji dla 50 elementów był podobny do czasu dla 40 elementów. Ze wzrostem zagęszczenia różnica jednak maleje.

W kwestii reprezentacji struktury – listowej, czy macierzowej – znalezienie efektywniejszej nie jest już takie oczywiste, bowiem obie wykazywały podobną efektywność przy każdym z algorytmów – raz jedna, raz druga była odrobinę lepsza. Nie da się jednoznacznie powiedzieć, że jedna z nich jest zdecydowanie najlepsza.