Tomasz Kąkol, Kamila Kalecińska
Informatyka Stosowana
275893
279732

Systemy równoległe i rozproszone

PROJEKT 2 - SPRAWOZDANIE

1. Wprowadzenie

Celem projektu była rozproszona realizacja algorytmu Dijkstry do odnajdywania drzew najkrótszych ścieżek przy użyciu technologii Java RMI (Remote Method Invocation).

Podziałem zadań zarządza wielowątkowy program nadrzędny, natomiast część obliczeniową będzie realizować wiele procesów z serwerów RMI.

2. Algorytm Dijkstry

Algorytm Dijkstry polega na znalezieniu w grafie ścieżek o najmniejszym koszcie (najkrótszej długości krawędzi). Wejściem jest graf, podany najczęściej przy pomocy tzw. Macierzy sąsiedztwa. Kolejne kroki algorytmu można opisać następująco:

1. Wczytanie macierzy sąsiedztwa
2. Utworzenie tablicy najkrótszych ścieżek i wypełnienie jej wartościami: ∞
3. Wierzchołek początkowy –przypisanie wartości 0
4. Dla każdego z wierzchołków sąsiadujących zapisanie w Tablicy dystansu do danego sąsiedniego wierzchołka oraz sumy dystansów do bieżącego wierzchołka i sumy dystansów do kolejnego sąsiedniego wierzchołka
5. Powtórzenie procedury dla kolejnego wierzchołka – wierzchołka o najmniejszym dystansie

Realizacja algorytmu w postaci rozproszonej polega na podzieleniu tablicy i wykonaniu każdego z jej fragmentów przez poszczególne procesy: każdy z procesów oblicza n/p kolejnych wartości ścieżek. Wyniki są następnie przesunięte do procesu głównego.

3. Realizacja

mikefile – plik służący do zbudowania i uruchomienia programu

Client – jest to zbudowana aplikacja kliencka (zarządca). Zawiera takie klasy jak ‘Client’, ‘Dijkstra’ oraz ‘Graph’. Każda z nich pełni samodzielną funkcjonalność. Pierwsza jest punktem wyjścia do programu klienta, druga zawiera implementację omówionego w pkt. 2 algorytmu Dijkstra. Ostatnia klasa pełni rolę wczytania wartości macierzy sąsiedztwa z pliku z golderu testcase.

Server – jest to aplikacja serwera, wykonująca obliczenia programu. Klasa służy do odbierania zadań od klienta i zwracania wyników

IServer – jest to interfejs dla klasy Serwera (implementowany przez serwer). Interfejs jest wykorzystywany w aplikacji klienckiej

testcases – folder zawierający przykładowe przypadki tesotwe

4. Kompilacja i uruchomienie

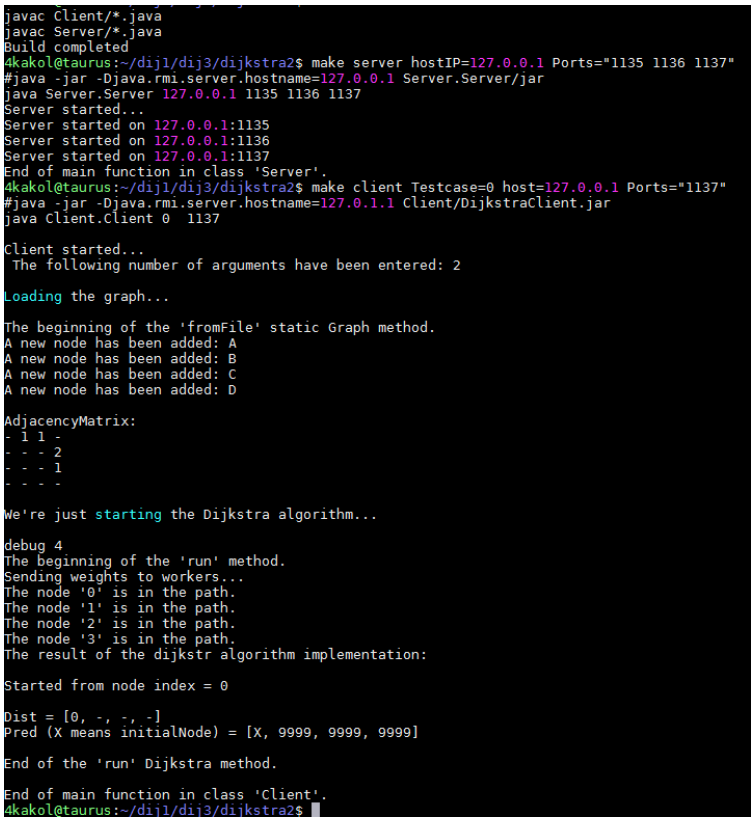
Aby zbudować projekt, należy użyć instrukcję make.

Następnie, by uruchomić procesy serwera, należy zastosować polecenie:

make server hostIP=127.0.0.1 Ports="1132 1133 1134"

By uruchomić procesy klienta, należy zastosować polecenie:

make client Testcase=0 host=127.0.0.1 Ports="1134"

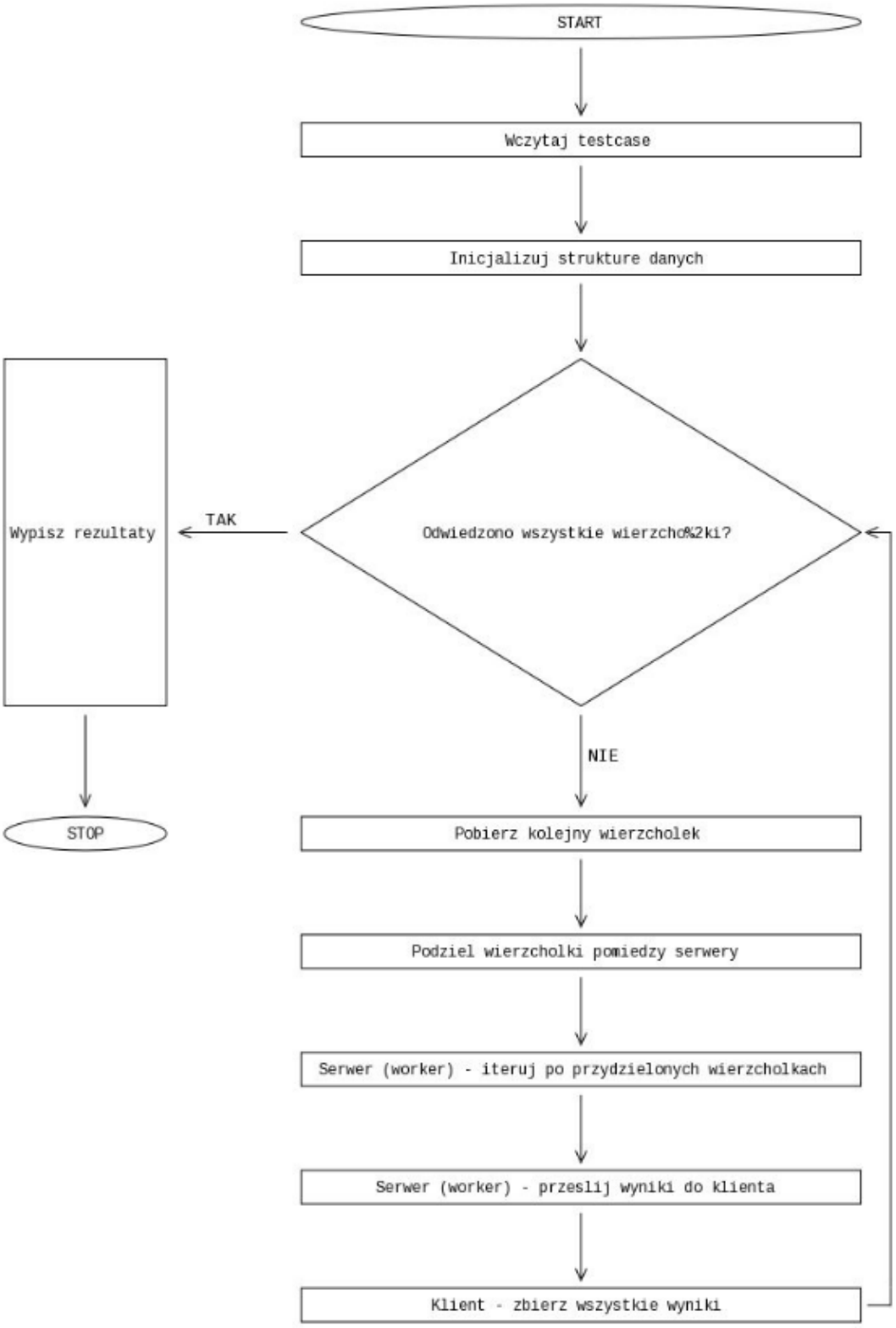


Rys.1 Przykładowy widok okna konsoli wykonanego programu.

- Testcase – nazwa pliku dla danego przypadku testowego
- hostIP – jest do wartość adresu IP, pod którym będą dostępne serwery
- Ports – są to numery portó○, na jakich zostaną uruchomione serwery

5. Algorytm działania aplikacji:

Poniżej zaprezentowano schemat wykonawczy programu:



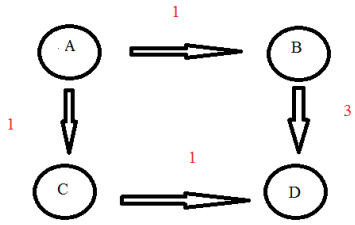
Rys.2. Schemat blokowy aplikacji

6. Podstawowy przypadek testowy:

Poniżej przedstawiono rozwiązanie dla przykładowego przypadku testowego.

Macierz sąsiedztwa dla 4 węzłów oraz jej interpretacja w formie grafu:

```
-,  1,  1,  -
-,  -,  -,  3
-,  -,  -,  1
-,  -,  -,  -
```



Rezultat:

```
Dist (- means no path) = [ 0,  1,  1,  2]

Pred (X means initialNode) = [ X, 0, 0, 2 ]
```

– gdzie 0 oznacza wierchołek ‘A’, natomiast 2 oznacza wierzchołek ‘C’

Analogicznie, zgodnie z algorytmem Dijkstra, rozwiązywane są kolejne przypadki testowe.