Praca domowa 6 – graph

Przemysław Kleszcz

Zadanie

Cel pracy domowej: Wyznaczyć ilość spójnych składowych grafu, o krawędziach podanych z zewnętrznego pliku.

Organizacja strukturalna

Projekt zawiera pięć plików: **AppClient.java, Graph.java, Search.java, IgraphRemote.java, ISearchRemote.java**

AppClient.java:
public class AppClient
public static void main(String[] args)
private static void prepareConnections()
private static BufferedReader getBufferedReader(String file)
private static void prepareGraph(String file)

Graph.java:
public class Graph
public Map<Integer, ArrayList<Integer>> obtainGraph()
public void appendEdge(Integer first, Integer second)
public boolean appendVertex(Integer vertex)
public void reInitialize()

Search.java:
public class Search
public int getNumOfConsComp(Map<Integer, ArrayList<Integer>> graph)
private List<Integer> getClosestMembers(Integer node, Map<Integer, ArrayList<Integer>> graph)
private void dfs(int v, Map<Integer, Boolean> checked, Map<Integer, ArrayList<Integer>> graph)

ISearchRemote oraz **IGraphRemote** to interfejsy modelujące odpowiedzialność klas Search i Graph.

main(String[] args) - Przyjmuje jako parametr tablicę argumentów linii komend. Punkt wejścia programu.

prepareConnections() - Inicjalizacja zmiennych będących referencjami do komponentów Graph i Search.

getBufferedReader(String file) - Przyjmuje jako parametr ścieżkę do pliku z danymi krawędzi grafu. Zwraca obiekt BufferedReader, pozwalający na odczyt linii pliku.

prepareGraph(String file) - Przyjmuje jako parametr ścieżkę do pliku z danymi krawędzi grafu. Inicjalizuję strukturę grafu oraz wypełnia ją wierzchołkami i krawędziami.

obtainGraph() - Zwraca strukturę grafu.

appendEdge(Integer first, Integer second) - Przyjmuję jako parametry wierzchołki grafu stanowiące pojedynczą krawędź. Dodaję krawędź do struktury grafu.

appendVertex(Integer vertex) - Przyjmuję jako parametr wierzchołek grafu. Dodaje wierzchołek do struktury. Zwraca *false* jeśli wierzchołek już istnieje w strukturze, w przeciwnym razie *true.*

reInitialize() - Tworzy nową, pustą strukturę grafu.

getNumOfConsComp(Map<Integer, ArrayList<Integer>> graph) - Przyjmuje jako parametr strukturę grafu. Zwraca liczbę spójnych składowych.

getClosestMembers(Integer node, Map<Integer, ArrayList<Integer>> graph) - Przyjmuje jako parametry wierzchołek oraz strukturę grafu. Zwraca listę sąsiadów wierzchołka.

dfs(int v, Map<Integer, Boolean> checked, Map<Integer, ArrayList<Integer>> graph) - Przyjmuje jako parametry wierzchołek grafu oraz strukturę grafu. Przeszukuje graf w głąb

Opis mechanizmu

Program jest zrealizowany w postaci aplikacji klienckiej + EJB. Na początku pobierana jest ścieżka do pliku z danymi krawędzi grafu z parametrów linii komend. Tworzona jest struktura nowego grafu a przeczytane wierzchołki i krawędzie dodane do struktury. Graf jest przekazywany do komponentu Search, w którym odbywa się zliczanie ilości spójnych składowych grafu. Zwrócona wartość jest przekazywana na wyjście i jest to wynik działania programu.

Do rozwiązania problemu znalezienia spójnych składowych został wykorzystany algorytm DFS. Przechodzenie grafu w głąb polega na odłożeniu wierzchołka na stosie/liście i oznaczeniu jako sprawdzony. W następnej kolejności przechodzi się do następnego wierzchołka i procedura jest powtórzona (rekurencja). Jeśli program dojdzie do wierzchołka, który nie ma krawędzi incydentnych z nieodwiedzonymi wierzchołkami, należy go wyciągnąć ze stosu/listy i pobrać kolejny wierzchołek.