Zaawansowane Techniki Programowania Zad 2 Path

Klasa Path

Klasa będąca rozszerzeniem HttpServlet, zawierająca implementacje całego programu

Pola:

- String CONNECTION_STRING - url wykorzystany do połączenia się z bazą danych
- int K_NODE - indeks wierzchołka do którego jest wyszukiwana ścieżka
- ArrayList<Edge> graph - lista zawierająca wszystkie krawędzie pobrane z bazy danych

Metody:

- collectDataFromDB

Opis :

Funkcja nawiązuje połączenie z bazą danych wskazana w connectionString, następnie pobiera wszystkie krawędzie z bazy danych i zapisuje w liście graph.

Dane wejściowe:

- String connectionString - url wykorzystany do nawiązania połączenia z bazą danych

Dane wyjściowe:

- void

- calculateCost

Opis :

Metoda oblicza koszt transportu dla ścieżki zadanej w parametrze wejściowym.

Dane wejściowe:

- ArrayList<Integer> path - lista zawierająca indeksy wierzchołków przez jakie przechodzi ścieżka

Dane wyjściowe:

- float cost - koszt transportu wzdłuż ścieżki

- doGet

Opis :

Metoda obsługuje zapytanie get. Pobiera z parametrów zapytania parametry dB oraz k, odpowiadające za url do bazy danych oraz indeks wierzchołka do którego ma zostać znaleziona ścieżka. Funkcja wywołuje główny algorytm i pobiera wynik końcowy który następnie jest wysyłany jako odpowiedź na zapytanie.

Dane wejściowe:

- HttpServletRequest request - zmienna przechowująca parametry zapytania GET
- HttpServletResponse response - zmienna odpowiedzialna za odpowiedź z servletu

Dane wyjściowe:

- void

- getAllPaths

Opis :

Metoda służy do zainicjowania ścieżki a następnie wywołania funkcji rekurencyjnej wyszukującej wszystkie ścieżki prowadzące do wierzchołka k.

Dane wejściowe:

- int src - indeks wierzchołka początkowego ścieżki (w przypadku tego programu zawsze wynosi 1)
- int des - indeks wierzchołka mającego się znaleźć na końcu ścieżki (jest równa wartości K_NODE)

Dane wyjściowe:

- ArrayList<ArrayList<Integer>> paths - lista wszystkich ścieżek prowadzących do wierzchołka des

- getAllPathsRec

Opis :

Funkcja rekurencyjna. Pobiera sąsiadujące wierzchołki ostatniego wierzchołka z listy currentPath , dodaje je do listy i wywołuje się rekurencyjnie. W momencie gdy ostatni wierzchołek jest równy wierzchołkowi des, dodaje currentPath do listy paths i przerywa rekurencje.

Dane wejściowe:

- ArrayList<Integer> currentPath - lista zawierająca indeksy wierzchołków aktualnie szukanej ścieżki
- int des - indeks wierzchołka mającego się znaleźć na końcu ścieżki (jest równa wartości K_NODE)
- ArrayList<ArrayList<Integer>> paths - lista wszystkich ścieżek prowadzących do wierzchołka des

Dane wyjściowe:

- void

- getEdgeCost

Opis :

Metoda zwraca przepustowość danej krawędzi.

Dane wejściowe:

- int x -indeks wierzchołka z którego wychodzi krawędź
- Int y - indeks wierzchołka do którego prowadzi krawędź

Dane wyjściowe:

- float p - przepustowość danej krawędzi

- getMinimalTransportCost

Opis :

Metoda pobiera wszystkie możliwe ścieżki prowadzące do wierzchołka docelowego a następnie wylicza ich koszt transportu jednocześnie zapamiętując ścieżkę o minimalnym koszcie.

Dane wejściowe:

- brak

Dane wyjściowe:

- float minCost - minimalny koszt transportu wybrany ze wszystkich ścieżek

- getNeighbors

Opis :

Metoda wyszukuje wszystkie krawędzie wychodzące z danego wierzchołka.

Dane wejściowe:

- int x - indeks wierzchołka dla którego mają zostać znalezione krawędzie

Dane wyjściowe:

- ArrayList<Edge> - lista wierzchołków wychodzących z wierzchołka x

Klasa Edge

Klasa zawierająca informacje o krawędzi.

Pola:

- int x - indeks wierzchołka leżącego na początku krawędzi
- int y - indeks wierzchołka leżącego na końcu krawędzi
- float p - przepustowość krawędzi

Metody:

- Edge(int x, iny y, float p)
- getX
- getY
- getP

Opis algorytmu

Algorytm działa w oparciu o rekurencyjne przeszukiwanie w głąb. Znajduje wszystkie możliwe ścieżki do wierzchołka K, następnie oblicza się koszt transportu dla każdej ścieżki. Wśród wszystkich ścieżek zostaje wybrana ścieżka o najmniejszym koszcie transportu.