Praca domowa 10 – plate

Szlachetka Miłosz
Nr albumu. 114014

1.Cel zadania

W zadaniu należało nawiązać połączenie z bazą danych i pobrać z niej niezbędne informacje dotyczące kosztów cięć tafli niejednorodnego materiału. Następnie na podstawie odczytanych danych, należało wyznaczyć minimalny koszt pocięcia całej tafli na pojedyncze elementy.

2.Struktura programu

@Path("/plate")
public class SqPlate - klasa pełniąca funkcję usługi RESTful'owej. Opatrzona adnotacją Path, która jednoznacznie wskazuje adres URL tej usługi. Klasa posiada metodę
@GET
@Path("/{datasource}/{table}")
public String minimumTotalCost(@PathParam("datasource") String datasource,
@PathParam("table") String tableName) - metoda odpowiedzialna za obsługiwanie żądań typu GET (tzw. endpoint). Metoda ta obsługuje żądania kierowane na adres zdefiniowany w adnotacji Path. Parametrami żądania są: datasource - definiuje źródło danych (javax.sql.Datasource) niezbędne do komunikacji z SQL-ową bazą danych, table - nazwa tabeli w bazie danych, która zawiera niezbędne dane. Metoda **minimumTotalCost** nawiązuje połączenie z bazą danych przy użyciu JNDI lookup oraz pobiera z niej koszty cięć tafli materiału. Następnie koszty te umieszczane są w listach (List<Float>) odpowiednio względem współrzędnej x(xCostList) oraz y(yCostList). Następnie przy użyciu metody **calculateMinCost** z klasy MyPlate wyliczany jest minimalny koszt pocięcia całej tafli materiału. Na końcu zwracany jest łańcuch znaków zawierający minimalny koszt.

public class MyPlate - klasa implementująca algorytm wyznaczania najmniejszego kosztu pocięcia tafli materiału na pojedyncze elementy. Zawiera metody:
public static double calculateMinCost(List<Float> xCostList, List<Float> yCostList) - oblicza minimalny koszt pocięcia całej tafli materiału na pojedyncze poprzez wyznaczenie optymalnej kolejności cięć.

private static double sumCosts(List<Float> costList) - dla podanej w argumencie listy zwraca sumę wszystkich elementów w tej liście jednocześnie usuwając z niej już zsumowane elementy. Sumowanie elementów jest kończone, gdy lista jest pusta.

3.Połączenie z bazą danych

Do nawiązania połączenia z bazą danych wykorzystano kontekst aplikacji oraz mechanim JNDI do którego należało podać nazwę, która identyfikuje źródło danych (datasource). Następnie zestawiono połączenie z bazą danych przy użyciu metody getConnection() na obiekce klasy DataSource. Przy użyciu obiektu klasy Statement wykonano zapytanie (SELECT) pobierające z bazy danych koszty cięć tafli materiału.

4. Algorytm

Początkowo z adresu URL żądania pobierana jest wartość definiująca źródło danych (datasource) oraz nazwa tabeli(tableName) w bazie danych. Następnie nawiązywane jest połączenie z SQL-ową bazą danych identyfikowaną przez “datasource”. Następnie z tabeli o nazwie “tableName” w bazie danych odczytywane są koszty pionowych i poziomych cięć tafli materiału. Odczytane koszty umieszczane są w listach (koszty cięć pionowych w jednej liście i poziomych w drugiej).

Wyznaczanie minimalnego kosztu cięcia tafli materiału:

Rozwiązanie tego problemu zostało uzyskane przy wykorzystaniu podejścia zachłannego. Początkowo sumaryczny koszt cięcia ustawiany jest na 0.0. Listy zawierające koszty cięć materiału (pionowo oraz poziomo) są sortowane w kolejności malejącej. Następnie inicjalizowane są zmienne przechowujące ilość cięć tafli wzgędem x-ów oraz y-ków. Następnie z obu list pobierane są największe wartości (są one na początku listy, gdyż uprzednio wykonano sortowanie). Wartości te są porównywane i wybierana jest wartość większa. Wartość ta jest mnożona przez liczbę cięć w innym kierunki niż wskazuje aktualnie wybrana wartość(np. dla wartości cięć względem osi x będzie to liczba cięć względem osi y i na odwrót). Otrzymany wynik jest dodawany do zmiennej przechowującej dotychczasowy, całkowity koszt cięcia. Następnie pobrany koszt cięcia jest usuwana z listy kosztów a licznik cięć dla przeciwnej współrzędnej jest inkrementowany o 1 (licznik przecięć x dla cięcia względem współrzędnej y i na odwrót). Powyższe operacje (zaczynając od pobrania największych elementó z obu list) są powtarzane do mementu gdy któraś z list nie będzie pusta. Gdy jedna z list będzie pusta następuje sumowanie kosztów cięć znajdujących się w niepustej liście oraz przemnożenie ich przez liczbę przecięć względem przeciwnej

współrzędnej (licznik przecięć x dla cięcia względem współrzędnej y i na odwrót). Otrzymana wartość jest dodawana do całkowitego kosztu pocięcia tafli materiału. Na końcu zwracany jest wyznaczony, minimalny koszt cięcia.

5. Złożoność obliczeniowa rozwiązania

N - sumaryczny rozmiar danych

Posortowanie list przechowujcych koszty cięć (kolejność malejąca):	**O(N*logN)**
Iterowanie (w pojedynczej pętli) po obu listach w celu wyznaczenia kolejnego kosztu cięcia	**O(N)**

Sumaryczna złożoność obliczeniowa rozwiązania = O(N*logN)