Corona taxi – Specyfikacja implementacyjna, projekt grupowy

Arek Kasprzak
Jakub Komorowski
Marcin Kowalczyk

01.12.2020.

.

Spis treści

1 Informacje ogólne

Program "Corona Taxi", którego celem jest optymalizacja ruchu Karetek Pogotowia, zostanie napisany w języku Java, w wersji 11. Graficzny interfejs programu powstanie przy użyciu frameworku JavaFX. Testy programu zostaną napisane we frameworku TestNG. Kod programu zostanie napisany w zintegrowanym środowisku programistycznym IntelliJ Idea 2020, z wykorzystaniem systemu kontroli wersji Git. Program będzie przeznaczony do użytkowania na systemach Windows oraz Linux.

2 Opis modułów

2.1 Struktura folderów

W projekcie rozróżniamy następujące foldery:

I. data - W folderze tym przechowywane będą dane wejściowe, używane przez program.

II. documentation - Folder, w którym będzie znajdowała się cała dokumentacja projektu tzn. specyfikacja funkcjonalna i specyfikacja implementacyjna.

III. src - Folder właściwy, w którym będzie umieszczony cały kod odpowiedzialny za działanie programu.

IV. test - folder, w którym będą umieszczone wszystkie testy, które zostaną przeprowadzone, aby sprawdzić poprawne funkcjonowanie programu.

2.2 Opis pakietów

I. src.main.java - pakiet, w którym będą umieszczone wszystkie klasy oraz interfejsy odpowiedzialne za działanie programu.

II. src.test.java - pakiet, który będzie zawierał wszystkie klasy testowe, mające sprawdzić poprawne funkcjonowanie programu.

3 Opis klas

Program będzie zawierał następujące klasy:

Main	Główna klasa programu. Metoda main będzie odpowiedzialna za działanie programu, wykorzystując w tym celu inne klasy.
GraphicController	Klasa odpowiedzialna za zapewnienie graficznego interfejsu użytkownika programu. Będzie ona również generować wizualizację mapy szpitali i połączeń w odpowiednim miejscu GUI.
Patient	Klasa reprezentująca pacjenta.
Hospital	Klasa reprezentująca szpital.
Monument	Klasa reprezentująca obiekt rozszerzający granice państwa.
Road	Klasa reprezentująca drogę pomiędzy szpitalami.
Intersection	Klasa reprezentująca skrzyżowanie dróg.
GraphSolver	Klasa odpowiedzialna za realizację zaprojektowanego przez nasz zespół algorytmu.
InputFileReader	Klasa odpowiedzialna za czytanie danych z pliku wejściowego, odpowiednie ich przetwarzanie i reakcję na ewentualne błędy.

4 Opis algorytmu

W programie zostaną zaimplementowane 3 główne algorytmy:

I. Algorytm łańcucha monotonicznego (monotone chain convex hull algorithm).

Zostanie on wykorzystany do stworzenia mapy kraju przy pomocy współrzędnych szpitali oraz obiektów.

```
Wejście: lista P punktów na płaszczyźnie.

Warunek: muszą być przynajmniej 3 punkty.

Posortuj punkty P według współrzędnej x (w przypadku remisu sortuj według współrzędnej y).

Zainicjuj U i L jako puste listy.
Listy będą zawierać odpowiednio wierzchołki górnego i dolnego wielokąta

dla i = 1, 2, ..., n:
    Dopóki L zawiera co najmniej dwa punkty i sekwencja ostatnich dwóch punktów L
    i punktu P[i] nie wykonuje obrotu w kierunku przeciwnym do ruchu wskazówek zegara:
        usuń ostatni punkt z L
    dołącz P[i] do L

dla i = n, n-1, ..., 1:
    Dopóki U zawiera co najmniej dwa punkty i sekwencja ostatnich dwóch punktów U
    i punktu P[i] nie wykonuje obrotu w kierunku przeciwnym do ruchu wskazówek zegara:
        usuń ostatni punkt z U
    dołącz P[i] do U

Usuń ostatni punkt z każdej listy (jest taki sam, jak pierwszy punkt z drugiej listy).
Połącz L i U w celu uzyskania wypukłego wielokąta P.
```

Rysunek 1: Algorytm łańcucha monotonicznego w pseudokodzie.

II. Wyszukiwanie w drzewie czwórkowym (Quadtree).

Zostanie on wykorzystany do znalezienia najbliższego szpitala z wolnymi miejscami, bezpośrednio po pojawieniu się pacjenta.

Algorytm bazuje na drzewie czwórkowym (Quadtree). Jest to drzewo, którego każdy węzeł może mieć do czterech dzieci. Każdy węzeł drzewa czwórkowego jest powiązany z kwadratem.

Niech:

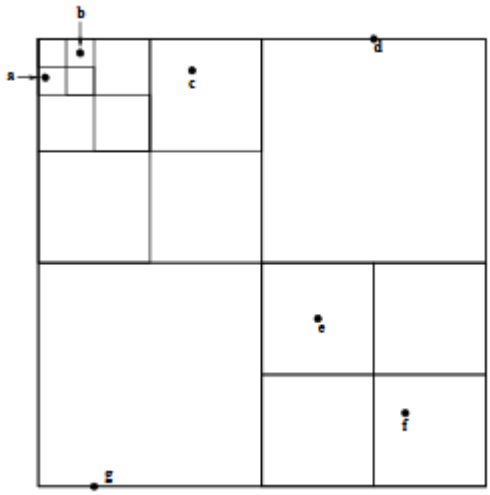
L - liczba poziomów w drzewie czwórkowym.

F_j - zbiór kwadratów na poziomie j drzewa czwórkowego.

E_j - zbiór kwadratów zawierający potencjalne najbliższe punkty na poziomie j drzewa czwórkowego.

Algorytm bada $E_0, E_1, \ldots, E_L$, gdzie $E_j \in F_j$. Zaczyna od $E_0 = F_0$. Aby wyznaczyć Ej, najpierw oblicza $best_{j-1}$ - najbliższy punkt q wśród

reprezentantów kwadratów w $E_0 \cup E_1 \cup \cdots \cup E_{j-1}$. Reprezentantem kwadratów jest ostatni wstawiony do danego kwadratu punkt. Następnie sprawdzone zostaje każde dziecko kwadratu w E_{j-1} i dodane do E_j, pod warunkiem, że odległość q od kwadratu wynosi co najwyżej d(q, $best_{j-1}$).



Rysunek 2: Przykład drzewa czwórkowego.

III. Algorytm Dijkstry.

Zostanie on wykorzystany do znalezienia najbliższego szpitala z wolnymi miejscami w przypadku, gdy pierwszy znaleziony szpital nie posiada wolnych miejsc.

```
Dijkstra(G,w,s):
   dla każdego wierzchołka v w V[G] wykonaj
      d[v] := nieskończoność
      poprzednik[v] := niezdefiniowane
   d[s] := 0
   Q := V
   dopóki Q niepuste wykonaj
      u := Zdejmij_Min(Q)
      dla każdego wierzchołka v - sąsiada u wykonaj
         jeżeli d[v] > d[u] + w(u, v) to
            d[v] := d[u] + w(u, v)
            poprzednik[v] := u
```

Rysunek 3: Algorytm Dijkstry w pseudokodzie.

5 Scenariusz działania programu

Program będzie wykonywał następujące kroki podczas swojego działania:

I. Wczytanie danych wejściowych.

II. Sprawdzenie poprawności otrzymanych danych.

III. Stworzenie obszaru "kraju" zdefiniowanego przez szpitale i obiekty.

IV. Dostarczenie pacjentów do szpitali poprzez:

   a). Sprawdzenie, czy pacjent znajduje się na terenie kraju.

   b). Jeżeli tak, to dostarczenie go do najbliższego szpitala.

   c). Jeśli wybrany w poprzednim kroku szpital nie posiada wolnych łóżek, to znalezienie najbliższego szpitala, który mógłby przyjąć pacjenta i dostarczenie go tam.

V. Wyświetlenie animacji dostarczania pacjentów wraz z informacjami o dokonywanych czynnościach.

6 Testowanie

6.1 Używane narzędzia i opis testów

Testy programu zostaną napisane we frameworku TestNG. Ich nazwy będą zgodne z ustaloną konwencją nazewniczą (punkt 6.2).

Napisane testy będą głównie skupiały się na sprawdzeniu poprawności zaimplementowanego przez nasz zespół algorytmu. Dokładnie zostanie przetestowana również poprawność czytania danych z plików i dodawania pacjentów. Graficzny interfejs użytkownika zostanie przetestowany manualnie przez członków naszego zespołu.
Przygotowane testy skupią się również na możliwych warunkach brzegowych:

- Błędny format pliku wejściowego
- Błędna struktura pliku wejściowego
- Niepoprawne dane w pliku wejściowym
- Pusty plik wejściowy
- Plik wejściowy zawierający dane przekraczające dopuszczalny zakres przyjmowany przez program

6.2 Konwencja nazewnicza

Nazwy testów będą zgodne z następującą konwencją nazewniczą:

nazwaMetody_testowaneZachowanie_spodziewaneZachowanie