AiSD - laboratorium

Projekt zespołowy - specyfikacja implementacyjna

Kacper Baczyński, Michał Kiełczykowski, Marek Knosala,
Edward Sucharda

18 grudnia 2020

1 Wstęp

Niniejszy dokument jest ściśle powiązany z dokumentem dotyczącym dokumentacji funkcjonalnej projektu zespołowego z przedmiotu Algorytmy i Struktury Danych w roku akademickim 2020/2021 na Wydziale Elektrycznym Politechniki Warszawskiej, gdyż zawiera opis implementacyjny algorytmu wykorzystanego do rozwiązania problemu przedstawionego jako założenia projektowe. Aby nie powielać informacji ogólnyuch dotyczących projektu zalecane jest zapoznanie, ze wspomnianym dokumentem, gdyż znajduje sie w nim dokładny opis i założenia projektu.

2 Opis struktury projektu

2.1 Założenia wstępne

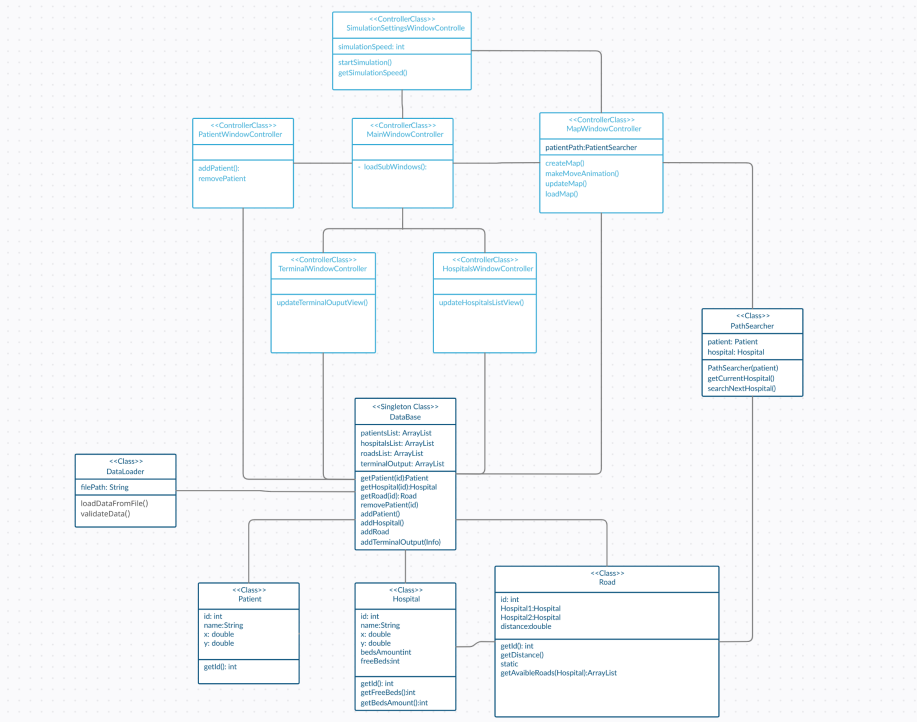
W celu relizacji problemu przedstawionego w celu projektu, rozwiązać należało kwestie sprawdzenia poprawności danych, stworzenie grafu połaczeń pomiędzy szpitalami a następnie kwestie odpowiedniego przydziału pacjenta do najbliższego szpitala oraz ewentualne jego przemieszczenia w przypadku braku wolnego łóżka. Dodatkowo w programie należało rozwiązać kwestie graficznej prezentacji danych wejściowych oraz wyników działania algorytmu w formie czytelnej dla użytkownika.

2.2 Wykorzystane technologie

W celu zrealizowania założeń projektu postanowiono wykorzystać jezyk porogramowania Java. Jego dużą zaletą jest fakt możliwości uruchamiania na większości dostępnych obecnie systemów opracyjnych. Kolejną zaletą jest bardzo dobre wsparcie programowania współbieżnego oraz spora ilość elementów (np. dkonterów danych) zaimplementowanych przez autorów języka, co znacząco przyśpiesza wykonanie projektu oraz zmniejsza podatność na błędy implemntacyjne. Dodatkowo do projektu postanowiono zastosować interfejs graficzny (ang. Graphical user interface - GUI) wykorzystując do tego celu środowisko JavaFX. Zaletą tego środkowiska jest możliwość sporej seperacji kodu żródłowego napisanego w języku Java realizaującego mechanikę działania GUI oraz cel projektu

od warstwy wizualnej interfejsu graficznego. Pozwala to na lepszą możliwość współpracy w projekcie zespołowym, gdzie dane osoby mogą realizować swoje cześci projektu minimalizując kolzje spowodowane pracą dwóch osób nad jednym elemtem projektu.

2.3 Diagram klas



Rysunek 1: Diagram klas.

3 Problemy Algorytmiczne

3.1 Znalezienie mapy państwa

1. Szukaj wierzchołka o najmniejszej współrzędnej x.
2. Dodaj go do nowo utworzonej listy wierchołków.
3. Stwórz początkowo pustą listę współczynników kierunkowych, która będzie gromadzić nachylenie prostej przechodzącej przez wybrane dwa wierchołki grafu.
4. Szukaj kolejnego wierzchołka o najmniejszej możliwej współrzędnej x (może to być również wierchołek, który ma taką samą współrzędną w osi x o ile ma większą wartość w osi y).

5. Po znalezieniu wierzhołka wyznacz współczynnik kierunkowy prostej łączącej go i poprzedni punkt. Jeśli jest on mniejszy od ostatniego elemtnu na liście współczynników to dodaj ten wierzchołek do listy wierzchołków, a współczynnik do listy współczynników. Jeśli lista współczynników jest pusta to warunek jest zawsze spełniony. Jeżeli warunek nie jest sepłniony to usuń z obu list ostatni element i sprawdź analogiczny warunek dla nowych ostatnich elementów obu list. Jeśli warunek jest spełniony to dodaj elementy do listy, jeśli nie to ponownie usuwaj tak długo aż warunek będzie spełniony.

6. Kroki od 3 do 5 powtarzaj tak długo aż zostaną sprawdzone wszystkie wierzchołki - czy nie ma nowego wierzchołka o współrzędnej y większej niż ostatni z listy.

7. Wykonaj kroki 1-6 z tym, że:

   a) zacznij od ostatniego punktu z listy wierzchołków z punktów 1-6,

   b) szukaj kolejnych wierzchołków malejąco wzdłuż osi y takich, żeby ich współrzędne x były coraz większe,

   c) postępuj analogicznie jak w punkcie 5.

8. Wykonaj kroki 1-6 z tym, że:

   a) zacznij od ostatniego punktu z listy wierchołków z punktu 7,

   b) szukaj kolejnych wierzchołków malejąco wzdłuż osi x takich, żeby ich współrzędne y były coraz mniejsze,

   c) postępuj jak w punkcie 5.

9. Wykonaj kroki 1-6 z tym, że:

   a) zacznij od ostatniego punktu z listy wierzchołków z punktu 8,

   b) szukaj kolejnych wierzchołków rosnąco wzdłuż osi y takich, żeby ich współrzędne x były coraz mniejsze,

   c) postępuj analogicznie jak w punkcie 5.

3.2 Znalezienie skrzyżowań

//Naive or Bentley–Ottmann algorithm

W postawionym zadaniu istnieje założenie, że jeżeli drogi przecinają się to w miejscu przecięcia powstaje skrzyżowanie. Powoduje to, że od pewnego szpitala do innego szpitala można dojechać okrężną, lecz szybszą drogą, mimo że nie istnieje ona w pliku wejściowym. Biorąc pod uwagę, że wszystkie obiekty mapy posiadają współrzędne kartezjańskie, punkty przecięć można wyznaczyć w sposób czysto matematyczny. Przedstawiając drogę od szpitala do szpitala jako odcinek, można przedstawić wszystkie drogi jako odcinki i znaleźć ich punkty przecięcia. Jest to metoda naiwna, ponieważ wymaga ona przeanalizowania każdej drogi z każdą inną drogą. O ile dla jednej pary złożoność obliczeniowa to O(1), tak dla n dróg jest to już $O(n^2)$.

Istnieje także bardziej przemyślany algorytm, który poprzez "omiecenie wiązką" przez wszystkie odcinki jest w stanie wykryć ich punkty przecięć. Implementacja algorytmu Bentley–Ottmann wymaga przedstawienia dróg jako posortowanych punktów względem osi X oraz odcinków. Algorytm można przedstawić następująco:

- Pionowa linia "przemiata" wszystkie punkty od lewej do prawej.
- Po natknięciu się na lewy(początkowy) punkt odcinka, odcinek oznaczany jest jako aktywny.
- Następnie sprawdzane są przecięcia z najbliższymi, aktywnymi odcinkami powyżej i poniżej aktualnego odcinka.
- Jeżeli odcinki aktywne się przecinają wtedy jest znajdowany i zapamiętywany punkt przecięcia między tymi odcinkami.
- W momencie gdy pionowa linia napotka końcowy punkt odcinka, wtedy dezaktywuje ona odcinek. Sprawia to, że odcinek nie jest brany dalej do analizy przecięć z innymi odcinkami.

Taki sposób pozwala na ograniczenie zbioru sprawdzanych odcinków do sąsiedztwa potencjalnie przecinających się odcinków. Dzięki temu nie potrzeba sprawdzać każdej pary odcinków ze sobą, co skutkuje złożonością obliczeniową rzędu O((n+k)log n), gdzie n to liczba odcinków, a k liczba przecięć.

Rozwijając ten algorytm, punkty przecięć zostaną skrzyżowaniami z punktu widzenia działania programu, a drogi na przecięciach zostaną zmodyfikowane w zależności od tego w jakiej proporcji punkt podzielił odcinki.

3.3 Określenie czy pacjent znajduje się na terytorium państwa

1. Dla pacjenta o współrzędnych (x_p, y_p) i każdego boku mapy państwa (wielokąta wypukłego) $M = \{W_1 = (x_1, y_1), ..., W_n = (x_n, y_n)\}$, oblicz i sprawdź znak:

$$h = (y_{i+1} - y_i) * (x_p - x_i) - (x_{i+1} - x_i) * (y_p - y_i)$$

2. Jeśli h dla wszystkich boków mapy ma taki sam znak, to pacjent znajduje się na terytorium państwa. Jeśli nie, to pacjent znajduje się poza granicami państwa.

3.4 Transport pacjenta do najbliższego szpitala

1. Oblicz odległość pacjenta do wszystkich szpitali wg wzoru:

$$d_i = \sqrt{(x_{si} - x_p)^2 + (y_{si} - y_p)^2}$$

2. Wybierz szpital, do którego odległość d jest najmniejsza
3. Udaj się do wybranego szpitala i po dojeździe (gdy pozycja pacjenta=pozycji szpitala) sprawdź czy w wybranym szpitalu jest wolne łóżko. Jeśli TAK – KONIEC transportu pacjenta, zostaw pacjenta w szpitalu. Jeśli NIE - oznacz szpital jako odwiedzony i przejdź do punktu 4.

4. Dla szpitala, w którym obecnie znajduje się pacjent wyznacz inny szpital, do którego droga jest najkrótsza:
    4.1. Dla każdego szpitala i wierzchołka bezpośrednio połączonego z obecnym szpitalem (skrzyżowania) wyznacz długości połączeń i zapamiętaj je.
    4.2. Zawsze jeśli sprawdziłeś już jakiś wierzchołek zapamiętaj ten fakt (zapamiętaj, że udało się do niego wyznaczyć jakąkolwiek drogę).
    4.3. Jeśli w danym szpitalu nie było jeszcze karetki, to zakończ przeszukiwanie ścieżek przechodzących przez ten szpital.
    4.4. Jeśli w danym miejscu była już karetka, to przeszukuj dalej graf, zapamiętując sumaryczną drogę i trasę do każdego wierzchołka.
    4.5. Kończ przeszukiwanie danej ścieżki, gdy dotrzesz do wierzchołka, który był już sprawdzony.
    4.6. Po przeszukaniu całego grafu z obecnego szpitala, wybierz nieodwiedzony szpital, do ktorego sumaryczna droga jest najkrótsza. Z tak wybranym szpitalem przejdź do punktu 3.

4 Testy oprogramowania

Do testowania oprogramowania użyte będzie narzędzie JUnit.

Dla plików wejściowych - pliku z mapą oraz pliku z pacjentami przeprowadzone zostaną następujące testy:

1. Pusty plik.
2. Pusta sekcja.
3. Brak tytułu sekcji (znaku "#").
4. Nieprawidłowa liczba znaków strumienia "|".
5. Nieprawidłowy typ danych.
6. Niepoprawne watości danych - niedopuszczalna wartość zerowa, ujemnna lub wartość spoza zakresu danego typu danych.
7. Powtarzające się id lub para id (sekcja "Drogi").
8. Wykorzystanie nieistniejącego id w sekcji "Drogi".
9. Nieistniejący szpital użyty w sekcji "Drogi".

Inne (niedotyczące plików wejściowych) testy programu:

1.

5 Źródła

[1] `http://0x80.pl/articles/point_in_polygon.html`
(data i godzina dostępu: 16.12.2020 21:39)

[2] `https://eduinf.waw.pl/inf/utils/011_2011/0105.php`
(data i godzina dostępu: 17.12.2020 15:21)