AiSD - laboratorium

Projekt zespołowy - specyfikacja implementacyjna

Kacper Baczyński, Michał Kiełczykowski, Marek Knosala,
Edward Sucharda

17 grudnia 2020

1 Wstęp

2 Opis struktury projektu

2.1 Założenia wstępne

//Opis kolejnych postawionych problemów w zadaniu (narysowanie mapy, dodawanie pacjentów, stworzenie i przeszukiwanie grafu itp)

2.2 Wykorzystane technologie

W celu zrealizowania założeń projektu postanowiono wykorzystać jezyk porogramowania Java. Jego dużą zaletą jest fakt możliwośći uruchamiania na większości dostępnych obecnie systemów opracyjnych. Kolejna zaletą jest bardzo dobre wsparcie programowania współbieżnego oraz spora ilość elementów (np. dkonterów danych) zaimplementowana przez autorów języka, co znacząco przyśpiesza wykoanaie projektu oraz zmniejsza podatnosć 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 sporo możliwość seperacji kodu źródłowego napisanego w języku Java realizaującego mechanikę działania GUI oraz samego zadanienia projektu od warstwy wizualnej interfejsu graficznego. Pozwoala to na lepszą możliwosc współpracy w projekcie zespołowym, gdzie dane osoby mogą realizaować swoje cześci projektu minimalujazując kolzje spowodowaną pracą dwóch osób na jednym elemtem projektu.

2.3 Diagram klas

3 Problemy Algorytmiczne

3.1 Znalezienie obrysu państwa

//Problem Convex hull

3.2 Wykrycie 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 w obszarze Państwa

//Even-odd rule Algorithm

3.4 Znalezienie najbliższego szpitala

3.5 Przeszukiwanie grafu w celu znalezienia miejsca w szpitalu

//Zmodyfikowany DFS/BFS

4 Testy oprogramowania

5 Źródła