1 Wstęp

Celem projektu była implementacja algorytmu rozwiązującego problem lokalizacji punktu w przestrzeni dwuwymiarowej metodą trapezową. W sekcji 2 znajduje się opis teoretyczny działania algorytmu, wizualizacja wyników na zbiorach testowych oraz analiza efektywności działania algorytmu. Sekcja 3 zawiera dokumentację załączonego w projekcie programu.

2 Sprawozdanie

2.1 Wstęp teoretyczny

2.1.1 Opis problemu

Problem lokalizacji punktu w przestrzeni dwuwymiarowej można sformułować następująco:

Niech $\mathcal{S}$ oznacza podział przestrzeni dwuwymiarowej zawierający n krawędzi. Zapytanie o lokalizację punktu w $\mathcal{S}$ oznacza znalezienie takiej ściany f zawartej w $\mathcal{S}$, w której znajduje się zadany punkt q. Celem algorytmu jest stworzenie takiej reprezentacji $\mathcal{S}$, która pozwala odpowiadać na zapytanie lokalizacji punktu w jak najmniejszym czasie. Stworzona przez nas reprezentacja powinna także zajmować jak najmniej miejsca w pamięci.

2.1.2 Podział trapezowy

Metoda trapezowa, nazywana także podziałem trapezowym, polega na stworzeniu mapy trapezowej $\mathcal{T}(S)$ dla danego zbioru S zawierającego n odcinków, który reprezentuje podział przestrzeni. Oprócz mapy $\mathcal{T}(S)$, algorytm tworzy strukturę $\mathcal{D}$ pozwalającą odpowiadać na zapytania lokalizacji punktu. Mapa trapezowa $\mathcal{T}(S)$ budowana jest w sposób przyrostowy poprzez dodawanie kolejnych odcinków z S i tworzenie pionowych przedłużeń wychodzących z ich wierzchołków i kończących się po napotkaniu innego odcinka. Kolejność dodawania odcinków z S wpływa na wielkość i czas odpowiedzi na zapytania struktury $\mathcal{D}$, jednak przy użyciu metody randomizacji, jesteśmy w stanie założyć [1, s. 133-136], że złożoność pamięciowa struktury $\mathcal{D}$ jest rzędu $O(n)$, a czas odpowiedzi na zapytanie rzędu $O(logn)$. Ostatecznie, randomizowany algorytm przyrostowy budujący mapę $\mathcal{T}(S)$ i strukturę $\mathcal{D}$ ma oczekiwaną złożoność obliczeniową $O(nlogn)$.

Podział trapezowy przyjmuje następujące założenia dla zbioru S:

1. Odcinki nie przecinają się.
2. Wierzchołki odcinków mają parami różne współrzędne x.

Zbiór S spełniający powyższe założenia jest w *położeniu ogólnym*.

Jesteśmy w stanie pozbyć się założenia 2 poprzez dokonanie *przekształcenia ścinającego* na odcinkach z S o dostatecznie mały kąt φ i stworzenie mapy trapezowej dla nowo powstałego zbioru φS, który już jest w *położeniu ogólnym*. Okazuje się jednak [1, s. 137-139], że przekształcenie to może pozostać jedynie w domyśle i przy sprawdzaniu czy dany punkt leży na lewo od drugiego wystaczy porównywać je leksykograficznie.

2.1.3 Struktura mapy trapezowej

Mapa trapezowa $\mathcal{T}(S)$ reprezentowana jest jako graf trapezów. Każdy stworzony trapez ma dwa boki pionowe (uznajemy także boki o długości zerowej). Uznajemy, że dwa trapezy ze sobą sąsiadują gdy posiadają wspólną pionową ścianę. Dla zbioru S w *położeniu ogólnym*, każdy trapez ma maksymalnie czterech sąsiadów, do których przechowujemy wskaźniki. Każdy trapez zawiera także wskaźnik do odpowiadającego mu węzła w strukturze przeszukiwań $\mathcal{D}$.

2.1.4 Struktura przeszukiwań

Struktura przeszukiwań $\mathcal{D}$ reprezentowana jest przez *acykliczny graf skierowany* przypominający strukturą drzewo binarne. Struktura ta zawiera trzy typy węzłów:

1. Węzeł trapezu - zawiera wskaźnik do trapezu w mapie $\mathcal{T}(S)$
2. Węzeł wierzchołka - zawiera wskaźnik do jednego z wierzchołków odcinków z S oraz wskaźniki do dwóch węzłów-dzieci (lewego i prawego).
3. Węzeł odcinka - zawiera jeden z odcinków z S oraz wskaźniki do dwóch węzłów-dzieci (lewego i prawego).

Zapytanie w strukturze przechodzi następująco:

Zaczynamy w korzeniu grafu, w każdym węźle wierzchołka testujemy czy zadany punkt leży na lewo czy na prawo od trzymanego wierzchołka; jeżeli leży na prawo, to idziemy do lewego dziecka, a w przeciwnym przypadku idziemy do prawego. Analogicznie postępujemy w węzłach odcinka, tym razem jednak testując czy zadany punkt leży nad odcinkiem (idziemy w lewo), czy pod (idziemy w prawo). Jeżeli zadany punkt jest równy jednemu z wierzchołków, bądź leży na jednym z odcinków, to zwracamy odpowiednią informację.

2.1.5 Opis algorytmu

Algorytm w i-tym kroku buduje mapę $\mathcal{T}(S_i)$ oraz strukturę $\mathcal{D}_i$ dla listy punktów $S_i = (s_1, s_2, ..., s_i)$. Wynikiem jest mapa $\mathcal{T}(S_n)$ oraz struktura $\mathcal{D}_n$, które zawierają wszystkie odcinki z S. Poszczególne kroki algorytmu przebiegają następująco:

1. Tworzymy prostokąt (trapez), w którym zawiera się każdy odcinek w S i za jego pomocą inicjalizujemy strukturę $\mathcal{T}(S_0) := \mathcal{T}(\emptyset)$.
2. Tworzymy listę $S_n = (s_1, s_2, ..., s_n)$ bedącą losową permutacją odcinków ze zbioru S.
3. Dla każdego $i = 1, 2, ..., n$ powtarzamy:
   (a) Znajdujemy w obecnej mapie $\mathcal{T}(S_{i-1})$ trapezy $\Delta_1, \Delta_2, ..., \Delta_k$ przecięte przez odcinek s_i (przecięcie tylko na wierzchołkach nie jest liczone).
   (b) Usuwamy z $\mathcal{T}(S_{i-1})$ trapezy $\Delta_1, \Delta_2, ..., \Delta_k$ zamieniając je na trapezy powstałe po dodaniu s_i. Otrzymujemy w ten sposób mapę $\mathcal{T}(S_i)$
   (c) Usuwamy z $\mathcal{D}_{i-1}$ węzły odpowiadające trapezom $\Delta_1, \Delta_2, ..., \Delta_k$ i dodajemy węzły dla nowych trapezów odpowiednio je łącząc za pomocą węzłów wierzchołków i odcinków.
4. Zwracamy $\mathcal{T}(S_n)$ jako wynik.

Krok 3.(a) jesteśmy w stanie wykonać wykorzystując połączenia sąsiedzkie trapezów w mapie. Najpierw znajdujemy trapez Δ_1, wykonując zapytanie w strukturze $\mathcal{D}_{i-1}$ dla punktu będącego lewym wierzchołkiem odcinka s_i. Następnie podążamy wzdłuż odcinka s_i przechodząc po i dodając do listy przeciętych trapezów odpowiednich sąsiadów w mapie. Kończymy po napotkaniu prawego wierzchołka odcinka s_i. Dokładna implementacja wszystkich kroków algorytmu znajduje się w załączonym programie opisanym w sekcji 3. Opis każdego poszczególnego kroku algorytmu można znaleźć także w: [1, s. 129-133]

2.2 Specyfikacja środowiska użytego do obliczeń

2.3 Testy i wizualizacja

2.4 Analiza efektywności algorytmu

3 Dokumentacja

3.1 Część użytkownika

3.2 Część techniczna

4 Podsumowanie

Literatura

[1] M. van de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, (1997) *Computational Geometry*, Springer.