Algorytmy geometryczne

Sprawozdanie z ćwiczenia 4.

Krzysztof Kwiecień
gr. Wt_14.40_B

Dane techniczne urządzenia na którym wykonano ćwiczenie:

Komputer z systemem Windows 10 x64
Procesor: Intel Core I5 2500k @4.5Ghz
Pamięć RAM: 8GB
Środowisko: Jupyter notebook

Ćwiczenie zrealizowano w języku Python 3, z wykorzystaniem bibliotek *numpy* (umożliwiających wiele operacji numerycznych), *random* (do generacji losowych punktów) oraz *matplotlib* (do rysowania wykresów).
Jako tolerancje dla zera dla wyznacznika przyjęto 10^{-12}.

Opis realizacji ćwiczenia:

Ćwiczenie polegało na implementacji algorytmu wykrywającego przecięcia między odcinkami na płaszczyźnie 2D.

1. Tworzenie zbiorów danych (odcinków)

Aplikacja pozwala na tworzenie wprowadzanie odcinków przez użytkownika oraz generowanie losowych odcinków. Wprowadzony / wygenerowany zestaw danych zapisywany jest do pliku w formacie json jako lista linii. Następnie plik ten może być wczytywany, a jego zawartość wykorzystywana w algorytmie.

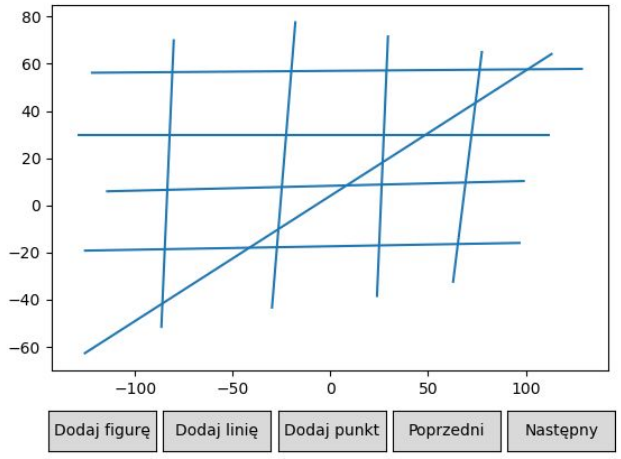
Wprowadzanie ręczne odcinków odbywa się za pomocą dostarczonego narzędzia graficznego. Za pomocą myszki należy wprowadzić kolejne odcinki i wykonać kod służący konwersji i zapisowi do pliku.

Generowanie odcinków polega na losowaniu dla każdej linii (których ilość jest zadana parametrem) pary odcinków o współrzędnych mieszczących się w zadanym przedziale.

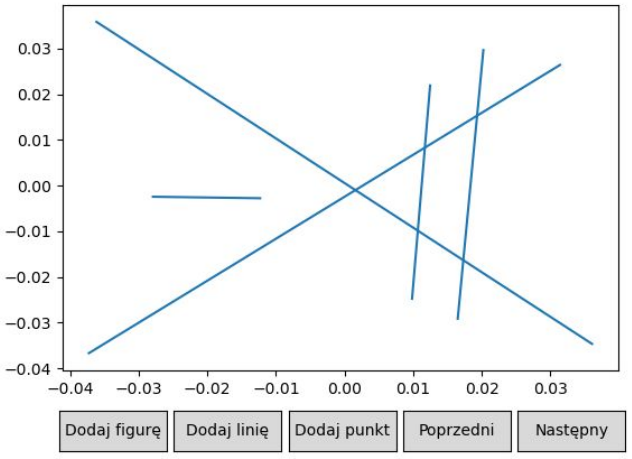
W celu przetestowania algorytmów zostały utworzone następujące zestawy odcinków:

- Przecięta siatka (“siatka.json”)
- Odpływ (“odplyw.json”)
- Przecinak (“przecinak.json”)
- Losowe 25 linii (“losowe.json”)

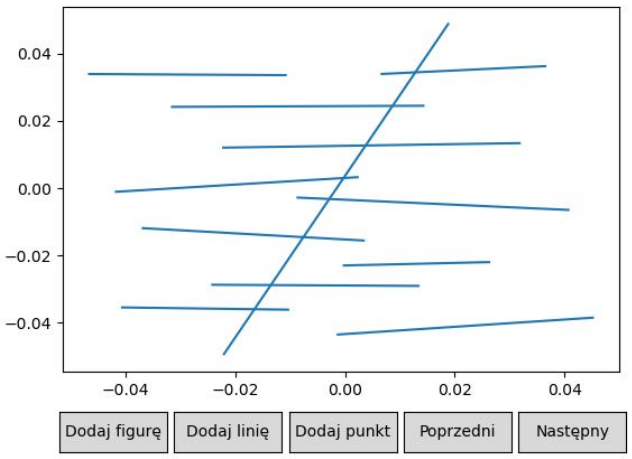
Rysunek 1.1 Zestaw odcinków ‘Przecięta siatka’



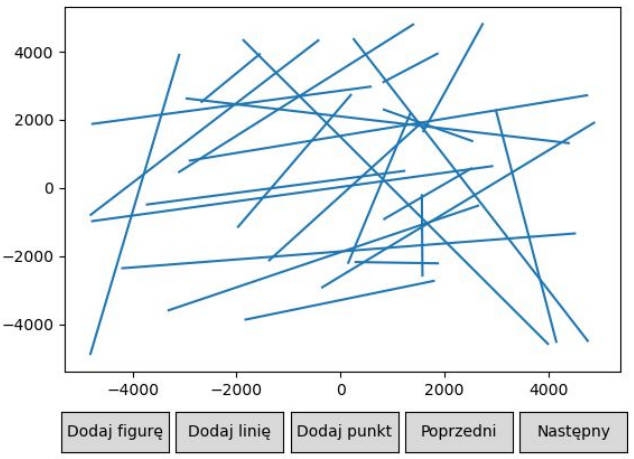
Rysunek 1.2 Zestaw odcinków ‘Odpływ’



Rysunek 1.3 Zestaw odcinków ‘Przecinak’



Rysunek 1.4 Zestaw odcinków ‘Losowe 25 linii’



2. Struktury danych

Dla struktur zdarzeń i struktur stanów wykorzystano zbiór uporządkowany (SortedSet z biblioteki sortedcontainers) zarówno w przypadku algorytmu wykrywającego jedno jak i wszystkie przecięcia. Jest to struktura oparta na zrównoważonym drzewie binarnym, a zatem operacje wstawiania i usuwania elementów mają złożoność logarytmiczną. Struktura ta zachowuje porządek wg. zadanego klucza - dla punktów jest to współrzędna x, natomiast dla linii jest to jej wartość y w danym położeniu miotły.

W strukturze zdarzeń znajdują się obiekty reprezentujące punkty. Dla każdego punktu poz jego współrzędnymi przechowywane są również indeksy linii na których ten punkt się znajduje. Indeksy te są takie same w przypadku punktów początkowych i końcowych linii, natomiast różne jeśli punkt ten jest punktem przecięcia.

W strukturze stanu znajdują się obiekty reprezentujące linie. Każda linia zawiera dwa punkty które ją definiują (początkowy i końcowy) oraz dodatkowo współczynniki m i b opisujące prostą zawierającą tą linie opisaną równaniem y = mx + b. Jest to potrzebne do porządkowania linii wg. Ich położenia pionowego.

W przypadku tej implementacji nie była wymagana zmian struktur między algorytmem wykrywającym jedno przecięcie a algorytmem wykrywającym wszystkie przecięcia. W przypadku tego pierwszego można to było uprościć z powodu braku konieczności zamieniania miejscami linii po wykryciu przecięcia, a zatem dzięki łatwiejszemu utrzymywaaniu porządku w strukturze.

3. Wykrywanie przecięć i zapobieganie duplikatów

Przecinanie się odcinków wykrywane z wykorzystaniem wyznaczników oraz parametrycznej reprezentacji linii. Sprawdzamy, czy przecięcie prostych, w których zawierają się nasze odcinki zawiera się w nich (czy nie leży poza nimi), a następnie wyliczamy dokładny punkt przecięcia. Znajdując nowy punkt przecięcia sprawdzamy czy nie jest on duplikatem - wykorzystujemy do tego fakt że każda para linii może przecinać się maksymalnie jeden raz. Zatem dla nowego punktu przecięcia sprawdzamy dla wszystkich wcześniej odkrytych punktów przecięć czy nie są one punktami leżącymi na tych samych dwóch liniach - jeśli są, to znaczy że jest to duplikat i nie dodajemy go ponownie do struktury zdarzeń ani zbioru wynikowego.
Jeśli nie jest to duplikat, dodajemy go do struktury zdarzeń i zbioru wynikowego.

4. Obsługa zdarzeń

Wyróżniamy 3 rodzaje zdarzeń (punktów) - punkt jest początkiem odcinka, punkt jest końcem odcinka, punkt jest punktem przecięcia dwóch odcinków. W przypadku punktu początkowego aktualizujemy we wszystkich liniach współrzędną x miotły (aby przy wstawianiu nowej linii porównania były wykonane zgodnie z aktualnym stanem). Następnie wstawiamy linię powiązaną z tym punktem do struktury stanu. Po wstawieniu ma ona (maksymalnie) dwóch sąsiadów (dolny i górny) z którymi należy sprawdzić czy się nie przecina.

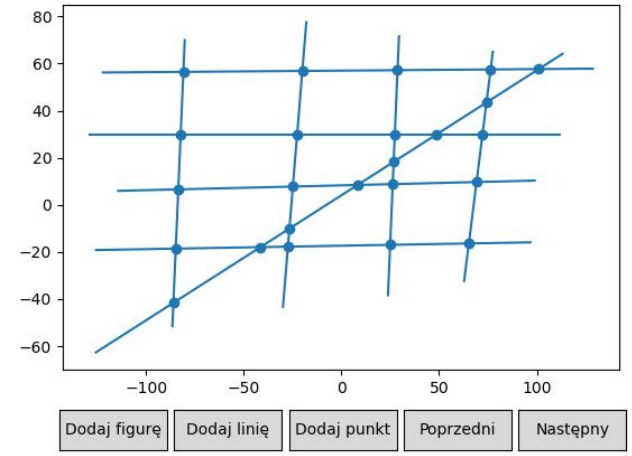
Po dojściu do punktu końcowego odcinka pobieramy jego górnego i dolnego sąsiada (o ile istnieją) i sprawdzamy czy się nie przecinają, po czym usuwamy ten odcinek ze struktury stanu.

W przypadku punktu przecięcia usuwamy ze struktury stanu oba odcinki na których leży ten punkt, następnie aktualizujemy dla wszystkich linii współrzędną x miotły z lekkim przesunięciem w prawo (aby miotła znajdowała się za przecięciem) po czym dodajemy je ponownie. Dzięki zaktualizowania położenia miotły odcinki zostaną zamienione miejscami. Następnie sprawdzamy czy odcinki te nie przecinają się z nowymi sąsiadami (nad i pod nimi).

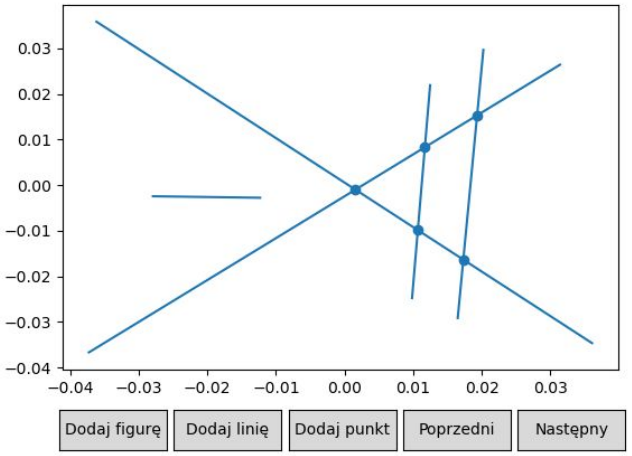
5. Wynik algorytmu

Algorytm na wejściu przyjmuje tablicę linii i zwraca zbiór punktów przecięć w postaci listy krotek, zbiór punktów będących obiektami (a więc zawierający informacje które linie się przecinają) oraz listę scen potrzebnych do wizualizacji.

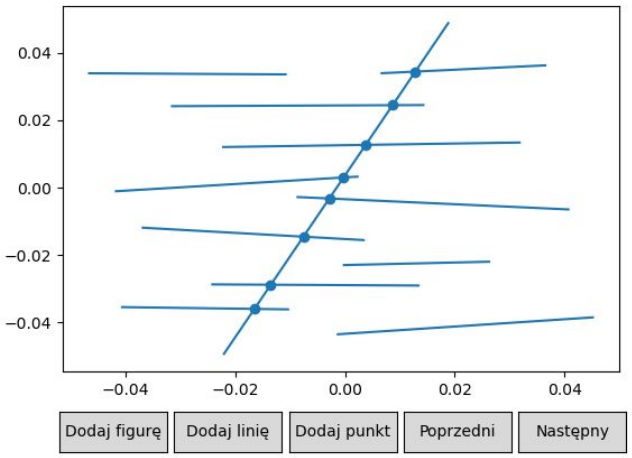
Rysunek 2.1 Wynik algorytmu dla zestawu odcinków ‘Przecięta siatka’



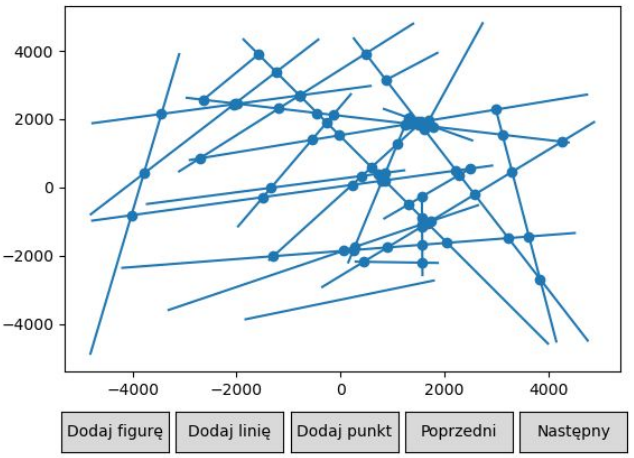
Rysunek 2.2 Wynik algorytmu dla zestawu odcinków ‘Odpływ’



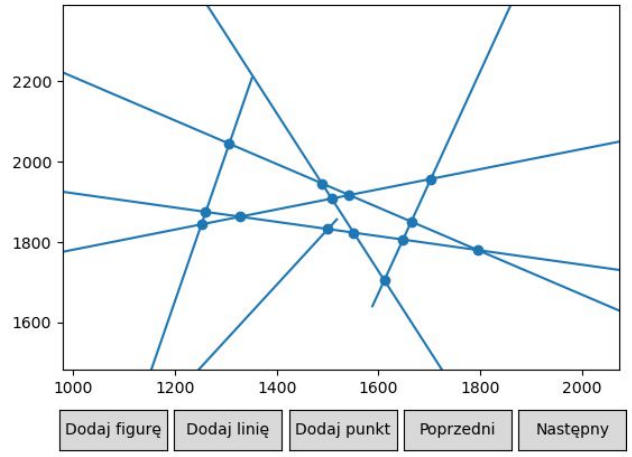
Rysunek 2.3 Wynik algorytmu dla zestawu odcinków ‘Przecinak’



Rysunek 2.4 Wynik algorytmu dla zestawu odcinków ‘Losowe 25 linii’



Rysunek 2.4.1 Wynik algorytmu dla zestawu odcinków ‘Losowe 25 linii’ - przybliżenie gęstego zbioru punktów przecięcia w otoczeniu [1600,1900]



Wnioski i spostrzeżenia

Algorytm zadziałał poprawnie dla testowanych zbiorów odcinków oraz dla dużych zbiorów odcinków generowanych losowo. Dzięki zastosowanym strukturom danych złożoność jest optymalna dla tego typu algorytmu. Wykrycie tylko jednego przecięcia jest dosyć proste, natomiast to właśnie obsługa zdarzeń będących punktami przecięcia jest trudnością w algorytmie głównym. Kolejną trudnością było usuwanie duplikatów - nie wystarczało sprawdzenie czy nowo wykryty punkt nie znajduje się już w zbiorze wynikowym, ponieważ w teorii te same punkty różniły się na współrzędnych na kilku miejscach po przecinku. Aby to rozwiązać można było zaokrąglać współrzędne punktów, lecz lepszym wyborem było skorzystanie z faktu, że para linii może przecinać się w maksymalnie jednym punkcie. Dzięki takiemu warunkowi dla dowolnej konfiguracji odcinków duplikaty nie mają prawa wystąpić.