Dzielenie i przeszukiwanie płaszczyzny za pomocą KD-Tree oraz Quad-Tree

Wojciech Łoboda, Krzysztof Pęczek

Wstęp teoretyczny

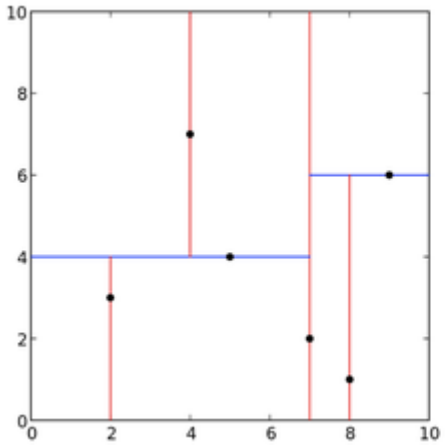
KDTree:

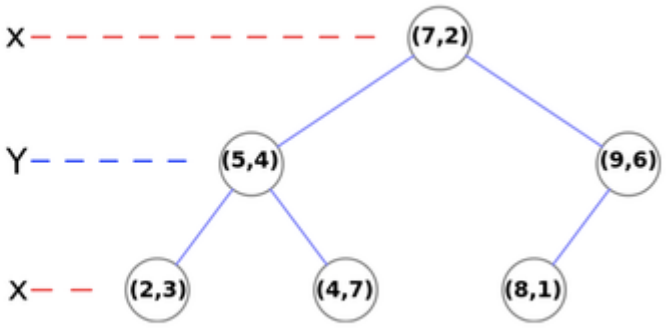
Struktura danych będąca uogólnieniem drzewa przeszukiwań 1-wymiarowego. Jest to drzewo binarne w którym każdy wierzchołek jest punktem k-wymiarowym.

Budowanie drzewa:

Inicjowanie drzewa polega na odpowiednim dzieleniu przestrzeni. Dla wierzchołków nie będących liśćmi drzewa, punkt wyznacza hiperpłaszczyznę podziału przestrzeni. Punkty o wymiarze podziału mniejszym lub równym niż ten wierzchołka, dodawane są do lewego poddrzewa natomiast punkty o większym do prawego.

Istnieją różne warianty wyboru hiperpłaszczyzny podziału. Dobór odpowiedniego ma znaczny wpływ na wydajność drzewa KD. W naszej implementacji zdecydowaliśmy się na wariant w którym do podziałów przestrzeni wykorzystujemy kolejne wymiary tj. Jeżeli ojciec wierzchołka dzielił przestrzeń względem wymiaru “x”, wierzchołek będzie dzielił względem “y” a dziecko znów względem “x” dla przestrzeni 2 wymiarowej. Przy tym wariancie drzewo w większości przypadków powinno pozostać zrównoważone jednak mogą pojawić się “wydłużone podobszary” kiedy wybraliśmy podział względem którego różnice we współrzędnych punktów są małe dla wybranego a odpowiednio duże dla innego wymiaru, które mogą pogorszyć wydajność struktury. Punkt podziału wyznaczający hiperpłaszczyznę dobierany jako jako mediana punktów aktualnie rozpatrywanych względem wymiaru, którym będziemy dzielić przestrzeń.





Przeszukiwanie obszaru:

Przy szukaniu punktów znajdujących się w zadanych obszarze trawersujemy drzewo zaczynając od korzenia w następujący sposób:

- Jeżeli rozpatrywany wierzchołek jest liściem do rozwiązania dodajemy punkt który reprezentuje.
- Jeżeli nie rozpatrujemy liścia sprawdzamy czy regiony które dzielą dzieci wierzchołka należą do szukanego obszaru.
    - Jeżeli region dziecka należy do szukanego obszaru, do rozwiązania dodajmy wszystkie punkty będącymi liśćmi odpowiedniego poddrzewa.
    - Jeżeli region dziecka przecina się z szukanym obszarem przechodzimy do rozpatrywania wierzchołka dziecka.

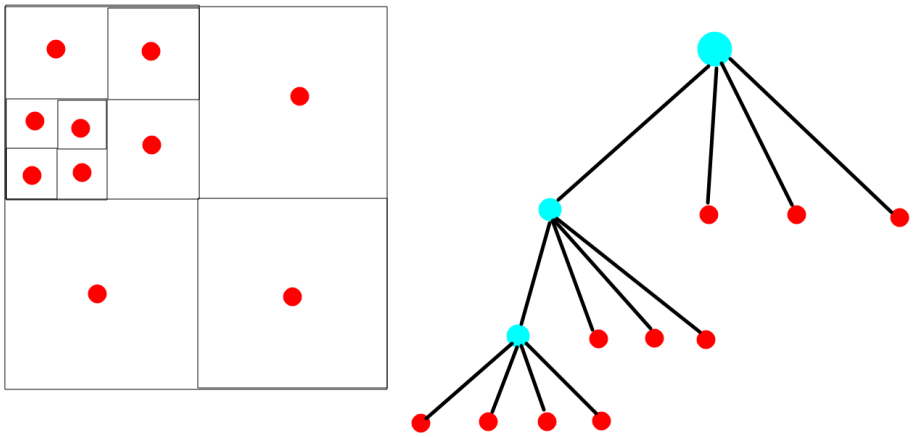
Złożoność obliczeniowa:

Budowanie struktury ma złożoność **O(nlogn)** gdzie n to liczba punktów w strukturze. najkosztowniejszym krokiem jest wstpęne sortowanie punktów, które pozwala nam ustalenie median przy podziałach.

Przeszukiwanie obszaru dla odpowiednio zbalansowanego drzewa ma złożoność czasową **O(**$\sqrt{n}$**)**, gdzie n to ilość punktów w strukturze.

QuadTree:

Struktura QuadTree dzieli określony prostokąt na płaszczyźnie na cztery równe w rozmiarach prostokąty. Z tego wynika, że każde drzewo QuadTree zawiera czwórkę dzieci. Każde drzewo posiada także listę punktów. Lista ta ma ograniczony od góry rozmiar (W naszym przypadku górnym ograniczeniem jest 1). Przykładowa reprezentacja takiego drzewa pokazana jest na rysunku poniżej:



Złożoność czasowa budowania QuadTree

Wkładanie punktu **p** do struktury QuadTree przebiega następująco:

1. Staramy się włożyć element do korzenia drzewa
2. Jeżeli lista punktów nie jest przepełniona (ograniczenie górne jest większe niż obecna ilość punktów w liście) dokładamy **p** do listy korzenia drzewa
3. Jeżeli lista jest przepełniona staramy się włożyć punkt do poddrzewa, którego obszar zawiera **p**. Wracamy się do kroku 1

Jak widać górnym ograniczeniem na czas spędzony, wkładając punkt do struktury jest funkcja klasy **O(h)** gdzie **h** to maksymalna głębokość drzewa. Dlatego włożenie **n** punktów do struktury QuadTree ma złożoność **O(nh)**.

Przeszukiwanie przedziału

Za pomocą tak zdefiniowanej struktury można łatwo przeszukać zadany nam obszar.

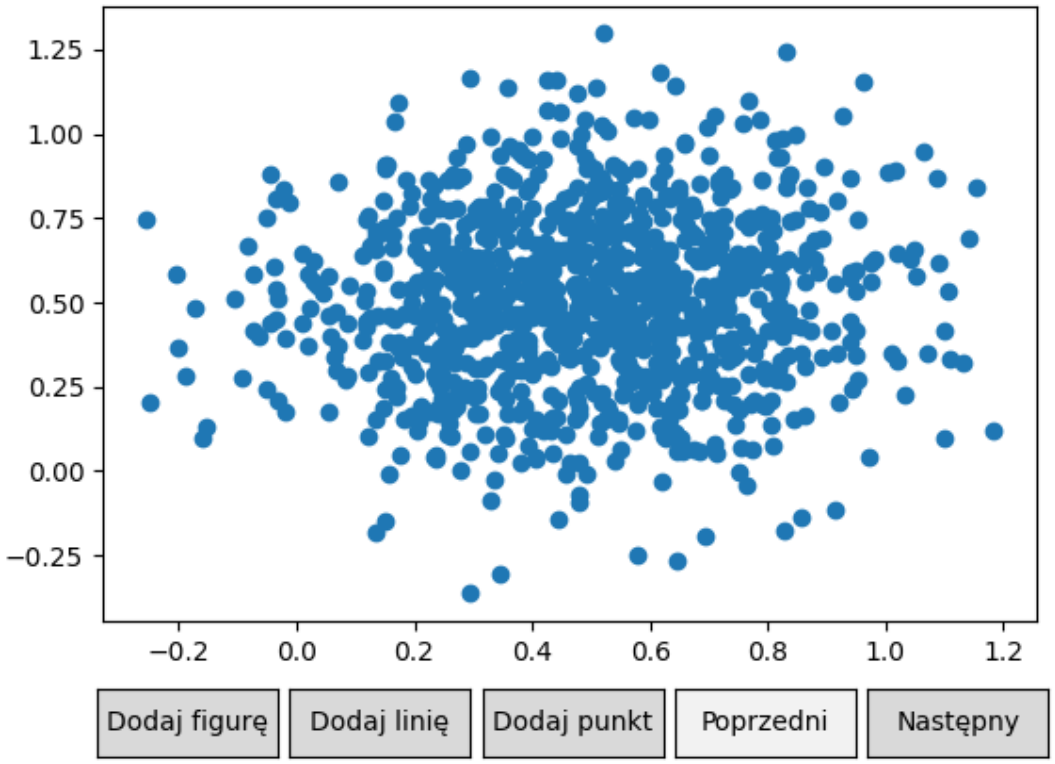
1. Rozpoczynamy od korzenia drzewa.
2. Jeżeli obszar drzewa nie przecina się z obszarem przeszukiwanym to przestajemy przeszukiwać w głąb tą gałąź (“odcinamy ją”)
3. Jeżeli jednak obszar drzewa przecina się z obszarem przeszukiwanym to dodajemy wszystkie punkty tego drzewa które zawierają się w obszarze przeszukiwanym i poszerzamy nasz wynik o wynik tej procedury na wszystkich poddrzewach.

Złożoność obliczeniowa takiego przeszukania to nadal **O(hk)** gdzie **h** - maksymalna głębokość drzewa, **k** - ilość liści zawartych w obszarze przeszukiwanym. W praktyce więc dla wielu zbiorów punktów przeszukiwanie takiego drzewa jest dużo mniej kosztowne. Niestety ponosi się zwiększone koszty zbudowania takiego drzewa oraz koszt obliczeniowy jeżeli struktura jest użyta na niekorzystnym zbiorze punktów.

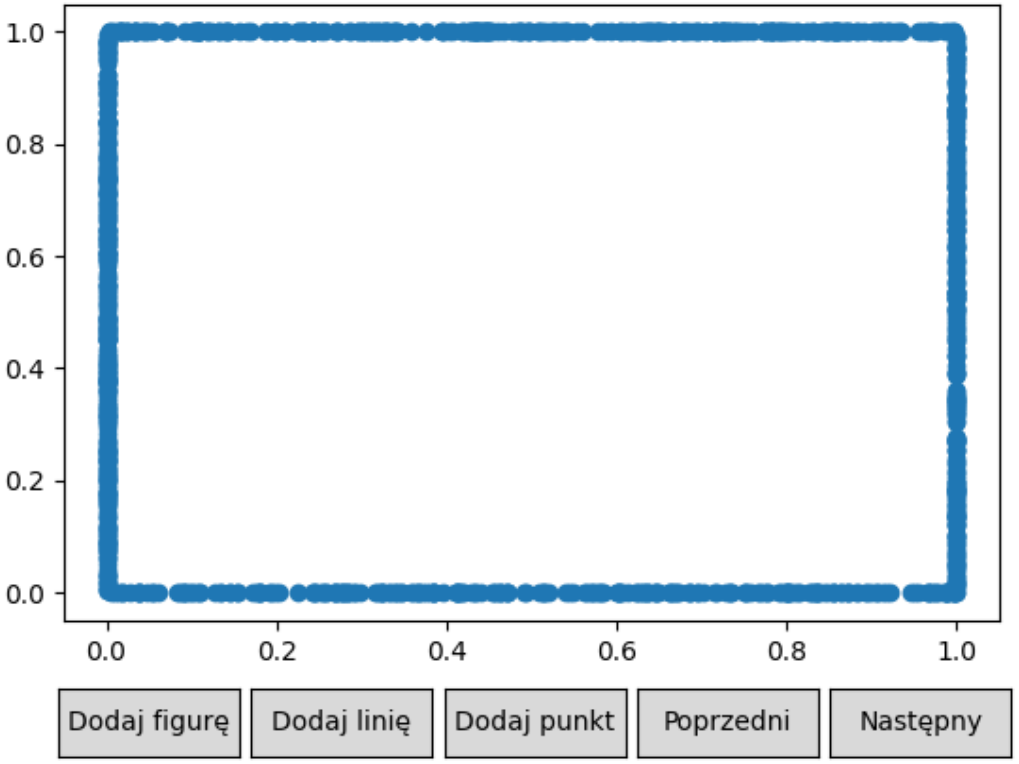
Testy dla różnych zestawów danych

Ilustracja zbiorów

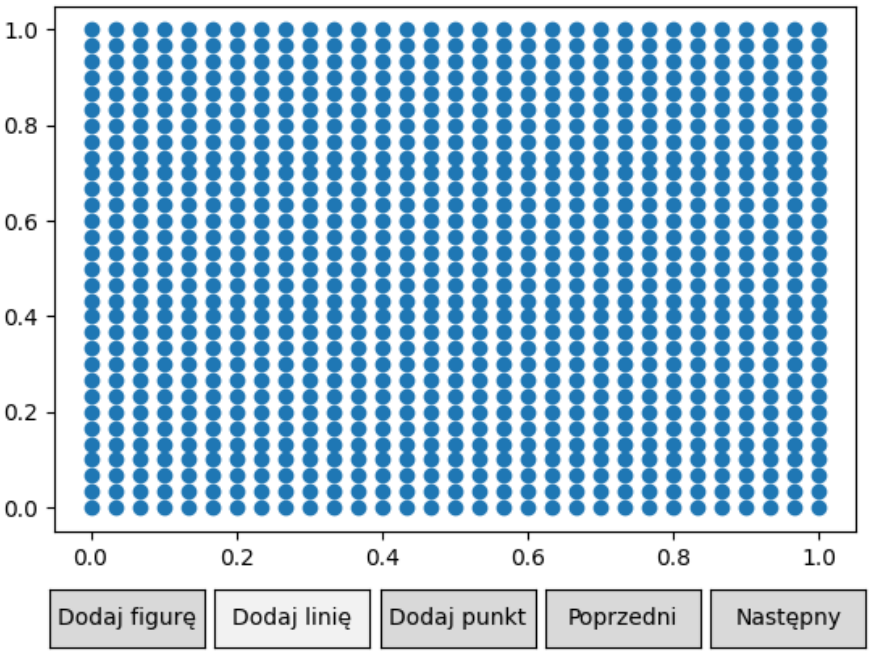
1.Zbiór punktów na rozkładzie normalnym



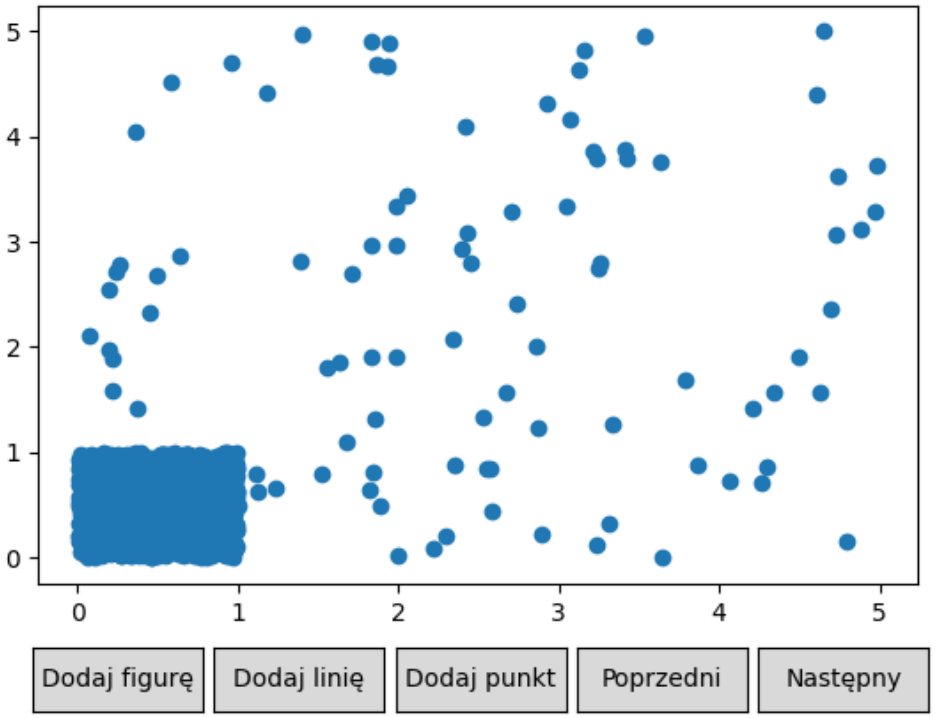
2.Zbiór punktów na obwodzie prostokąta



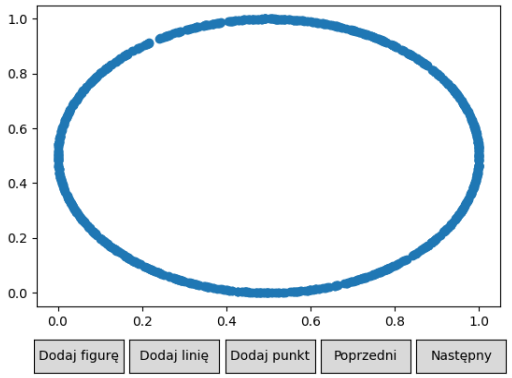
3.Zbiór punktów na siatce

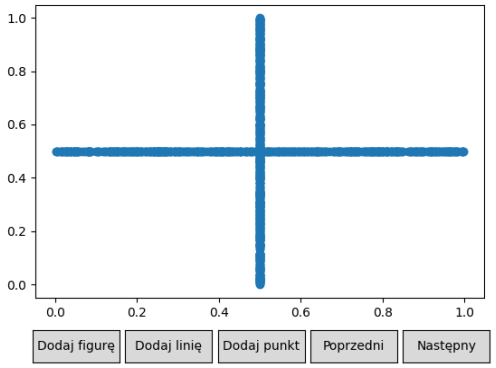


4.Zbiór punktów zgrupowanych z punktami odstającymi



5. 6. Zbiór punktów na okręgu i zbiór punktów na krzyżu

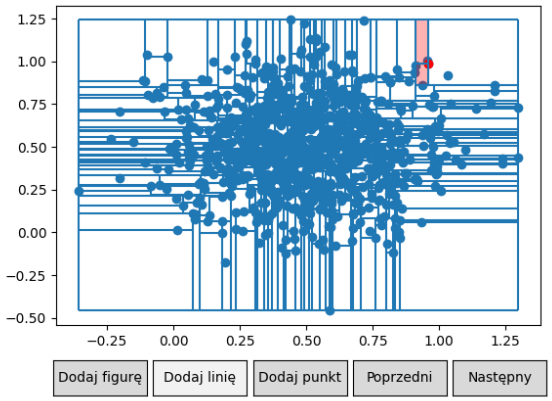


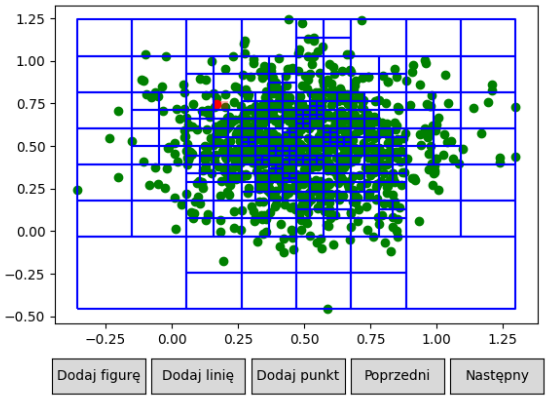


Analiza kolejnych zbiorów

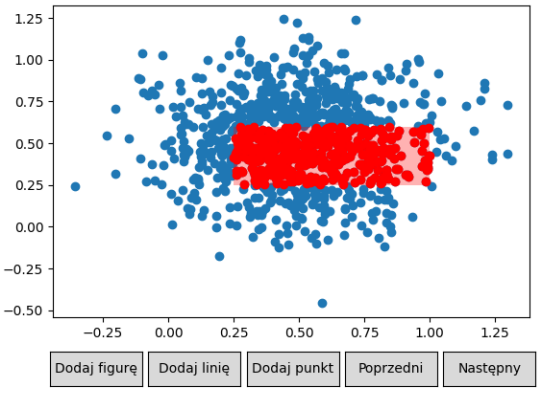
1.Rozkład normalny

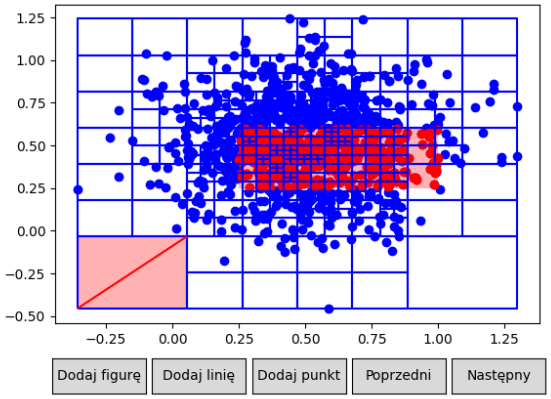
Wizualizacja budowania struktury dla 1000 punktów:





Wizualizacja wyniku zapytania dla x1=0.25 x2=1.0 y1=0.25 y2=0.60



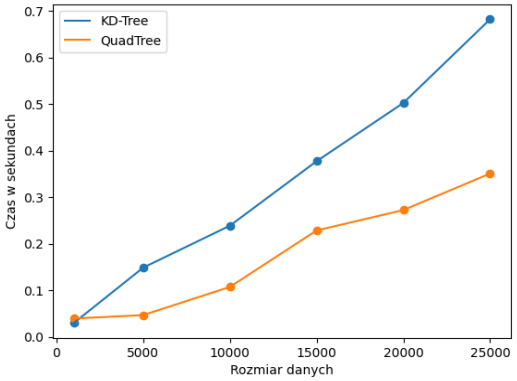


Wizualizacja wyniku zapytania dla x1=0.25 x2=1.0 y1=0.25 y2=0.60

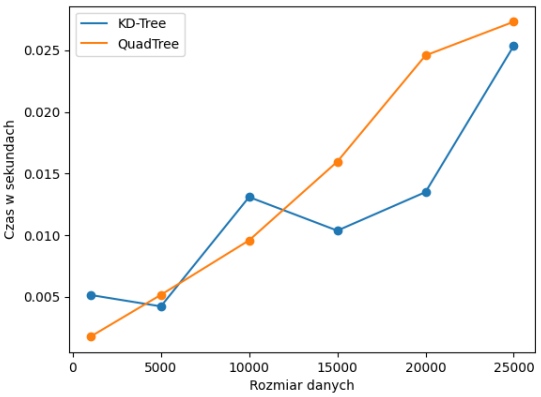
liczba punktów	czas budowania KDTree	czas budowania QuadTree	czas wyszukiwania KDTree	czas wyszukiwania QuadTree
1000	0.0296	0.0394	0.0037	0.0012
5000	0.1485	0.0467	0.0054	0.0067
10000	0.2387	0.1072	0.0301	0.0087
15000	0.3771	0.2284	0.0147	0.0194
20000	0.5022	0.2723	0.0124	0.0214
25000	0.6819	0.3506	0.0121	0.0260

Tabela z porównaniem czasu działania algorytmów. Czas podawany jest w sekundach.

Wykresy:



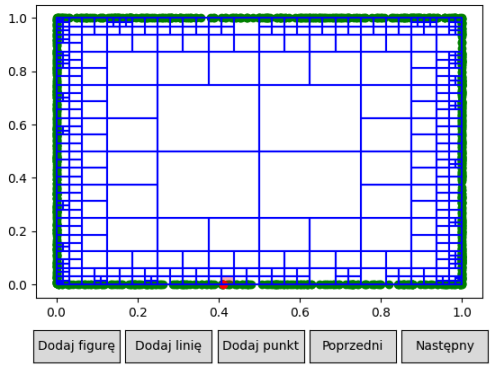
Czas budowania struktur

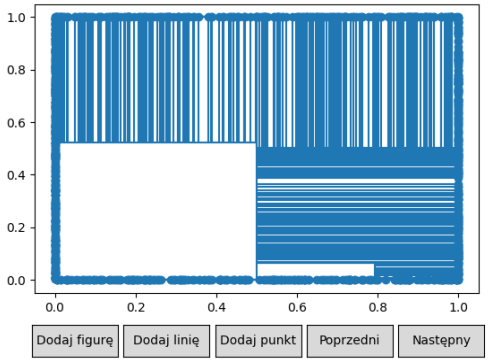


Czas wyszukiwania w strukturach

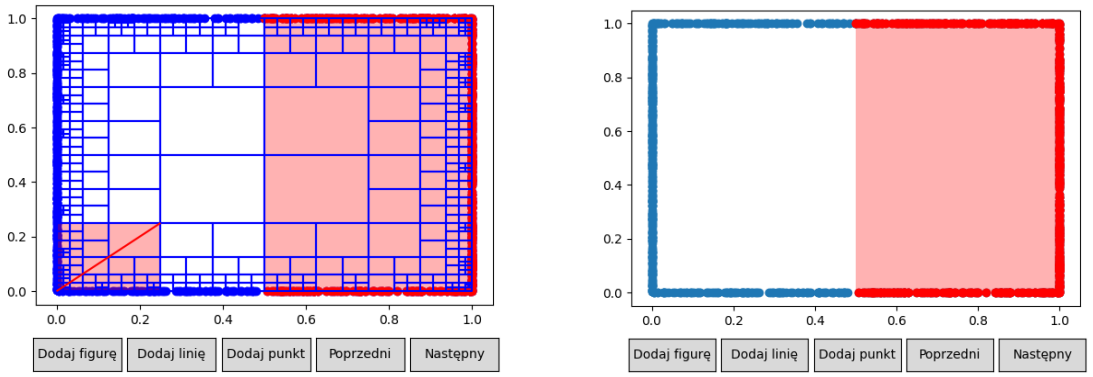
Wyniki zapytania są poprawne, czas budowania struktury wydają się być zbliżone z lekką przewagą struktury QuadTree. Czas wyszukiwania jest dla większości zestawów punktów mniejszy przy strukturze KDTree poza jednym przykładem w którym pojawia się nieoczekiwany wzrost czasu wynikający prawdopodobnie z nieidealnych warunków testowania.

2.Punkty na prostokącie





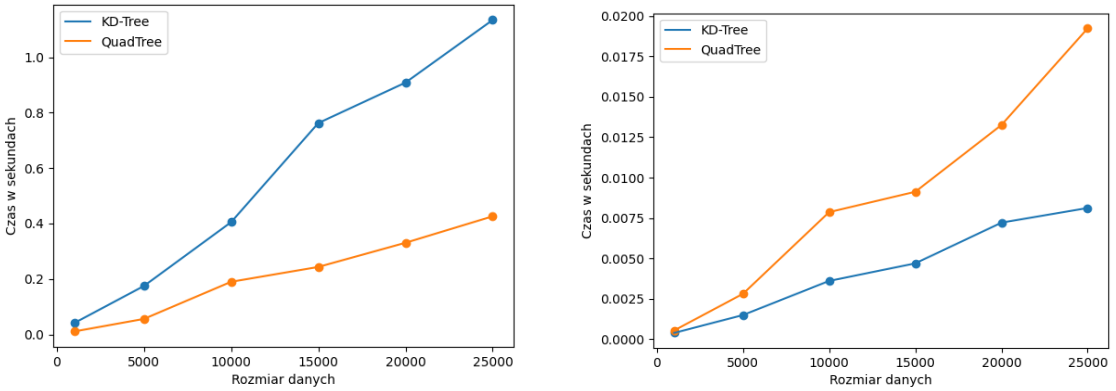
Wizualizacja budowy struktur dla 1000 punktów



Wizualizacja wyniku zapytania dla x1=0.5, y1=0.0, x2=1.0, y2=1.0

liczba punktów	czas budowania KDTree	czas budowania QuadTree	czas wyszukiwania KDTree	czas wyszukiwania QuadTree
1000	0.0407	0.0104	0.0004	0.0005
5000	0.1746	0.0557	0.0015	0.0028
10000	0.4049	0.1897	0.0036	0.0079
15000	0.7625	0.2433	0.0047	0.0091
20000	0.9086	0.3305	0.0072	0.0133
25000	1.1346	0.4261	0.0081	0.0192

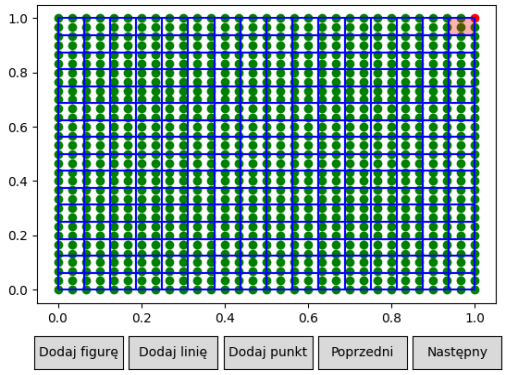
Tabela z porównaniem czasu działania algorytmów. Czas podawany jest w sekundach.

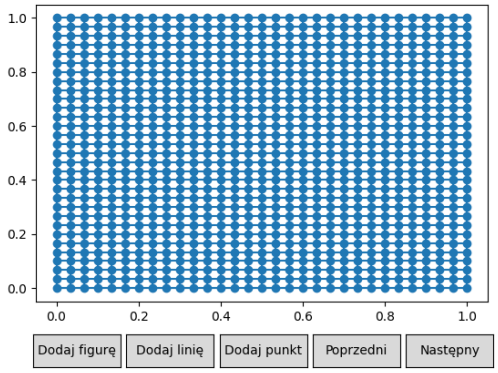


Wyniki porównania czasów budowy struktury danych (lewy wykres) oraz przeszukiwania struktury danych (prawy wykres). Dla różnej ilości punktów

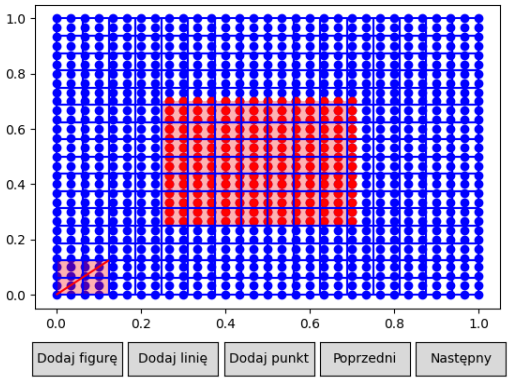
Dla tego zestawu danych również widoczna jest różnica w czasach wykonywania algorytmu. Wygenerowanie struktury zajmuje dłużej przy KDTree a dla największych zbiorów punktów różnica jest znaczna. Zaskakującą jest różnica w czasach przeszukiwania struktury QuadTree.

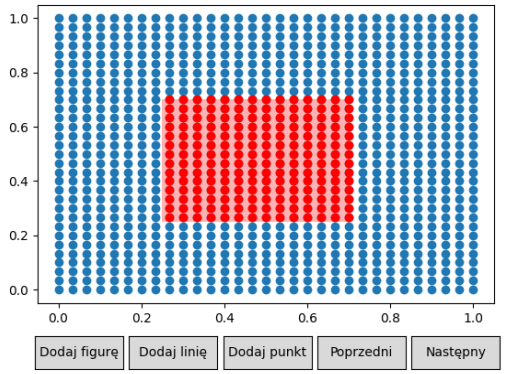
3.Punkty na siatce





Wizualizacja budowy struktur dla 1000 punktów

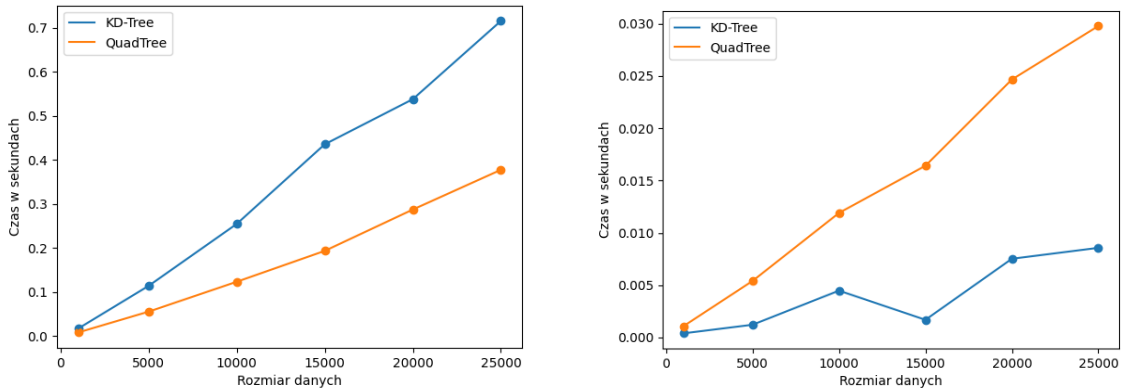




Wizualizacja wyniku zapytania dla x1=0.25, y1=0.25, x2=0.7, y2=0.7

liczba punktów	czas budowania KDTree	czas budowania QuadTree	czas wyszukiwania KDTree	czas wyszukiwania QuadTree
1000	0.0170	0.0082	0.0004	0.0011
5000	0.1141	0.0555	0.0012	0.0054
10000	0.2547	0.1233	0.0045	0.0119
15000	0.4356	0.1935	0.0017	0.0164
20000	0.5381	0.2874	0.0075	0.0247
25000	0.7157	0.3776	0.0086	0.0298

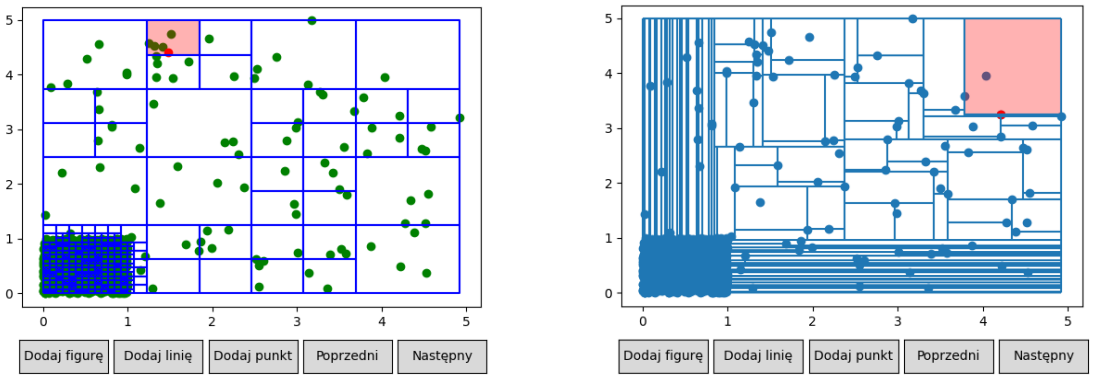
Tabela z porównaniem czasu działania algorytmów. Czas podawany jest w sekundach.



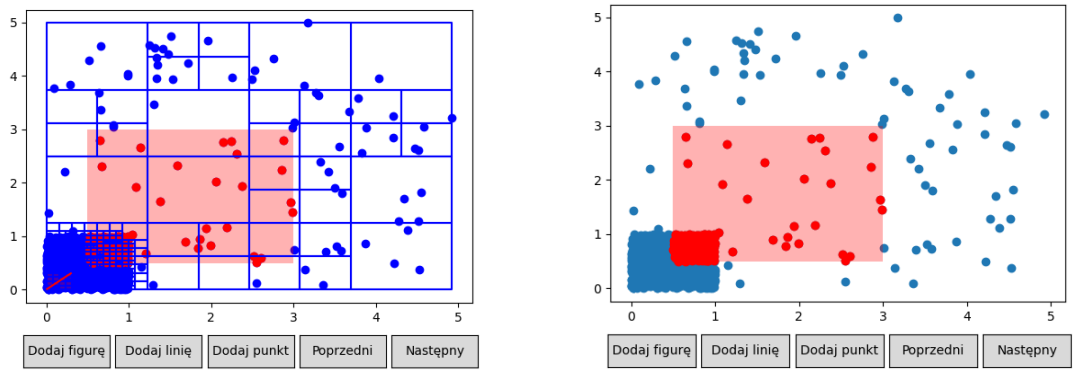
Wyniki porównania czasów budowy struktury danych (lewy wykres) oraz przeszukiwania struktury danych (prawy wykres). Dla różnej ilości punktów

Dla punktów jednakowo rozmieszczonych na siatce czas inicjowania jest podobny dla obu struktur z niewielką przewagą QuadTree jednak wyszukiwanie jest znacznie szybsze dla KDTree wynika to prawdopodobnie z tego że równomierny rozkład punktów na siatce gwarantuje że ta struktura będzie zbalansowana przy wykorzystanej implementacji.

4.Zbiór punktów zgrupowanych z punktami odstającymi



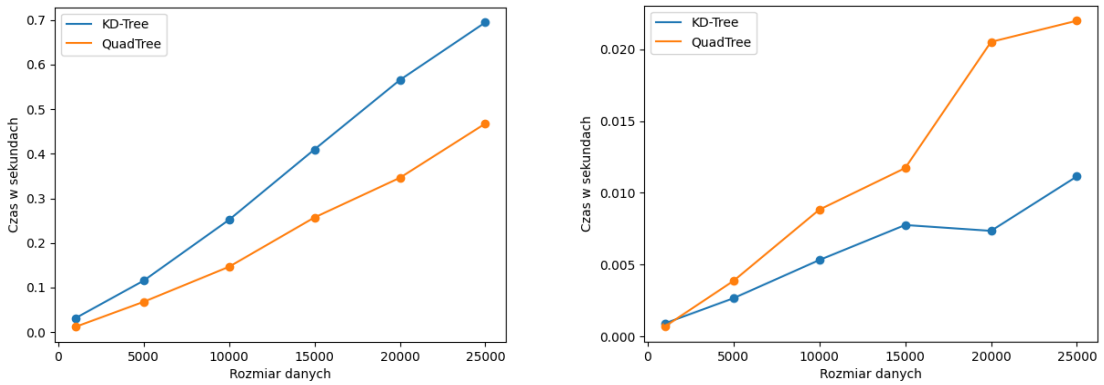
Wizualizacja budowy struktur dla 1000 punktów



Wizualizacja wyniku zapytania dla x1=0.5, y1=0.5, x2=3.0, y2=3.0

liczba punktów	czas budowania KDTree	czas budowania QuadTree	czas wyszukiwania KDTree	czas wyszukiwania QuadTree
1000	0.0311	0.0117	0.0009	0.0007
5000	0.1154	0.0680	0.0027	0.0039
10000	0.2519	0.1466	0.0053	0.0088
15000	0.4097	0.2572	0.0078	0.0117
20000	0.5655	0.3462	0.0074	0.0205
25000	0.6946	0.4677	0.0111	0.0220

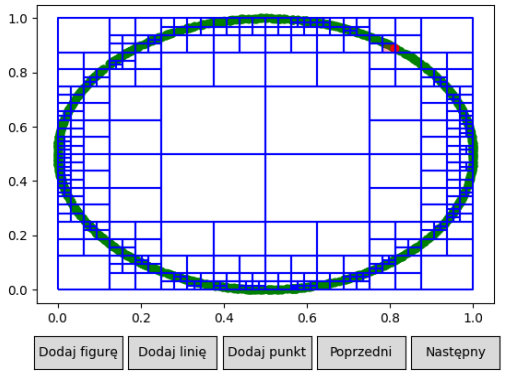
Tabela z porównaniem czasu działania algorytmów. Czas podawany jest w sekundach.

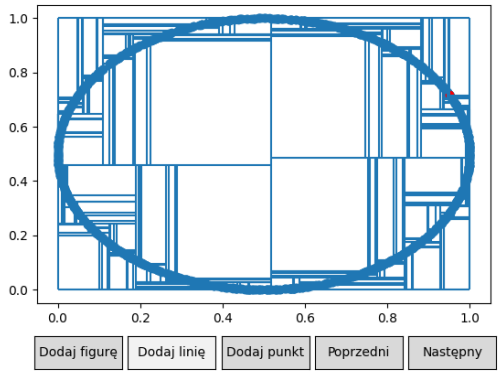


Wyniki porównania czasów budowy struktury danych (lewy wykres) oraz przeszukiwania struktury danych (prawy wykres). Dla różnej ilości punktów

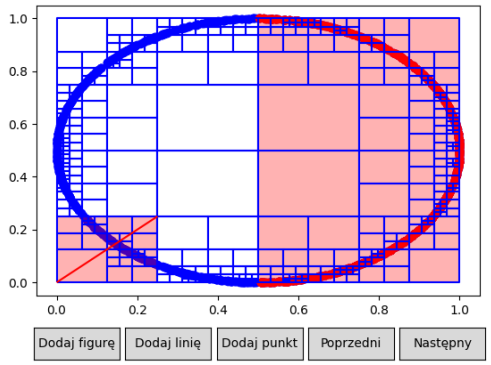
Podobnie jak w pozostałych zestawach czas budowania jest minimalnie krótszy przy QuadTree oraz wyszukiwanie jest szybsze przy użyciu struktury KDTree. Jak widać na wykresie powyżej wyszukiwanie w obszarze za pomocą QuadTree jest znacznie wolniejsze niż za pomocą KD-Tree. Wynika to prawdopodobnie z faktu, że drzewo zbudowane na takim zbiorze danych nie jest zbalansowane i przeszukiwanie na tak niezbalansowanym drzewie przypomina bardziej przeszukiwanie liniowe.

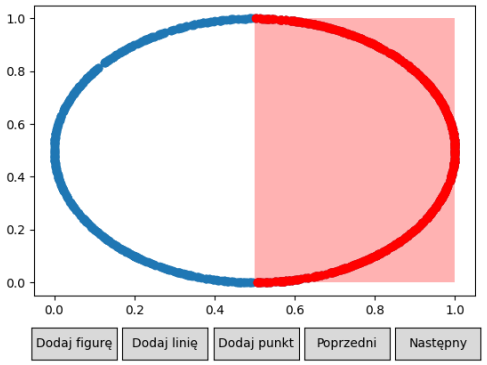
5.Zbiór punktów na okręgu





Wizualizacja budowy struktur dla 1000 punktów

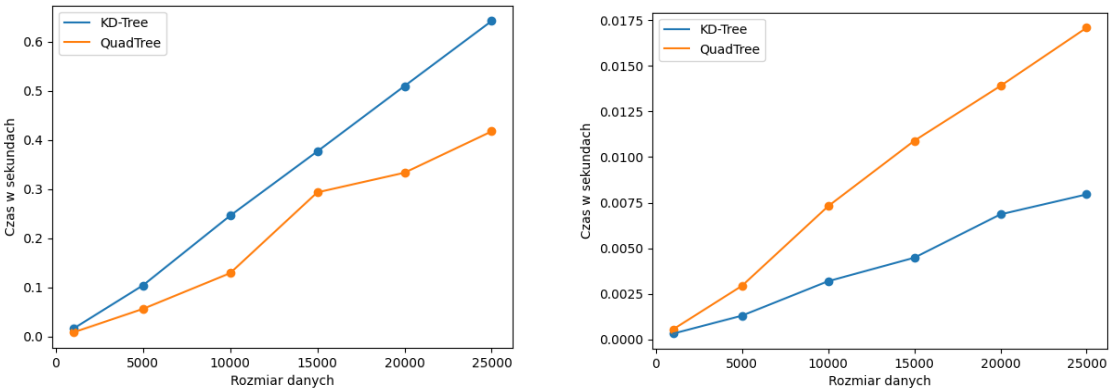




Wizualizacja wyniku zapytania dla x1=0.5, y1=0.0, x2=1.0, y2=1.0

liczba punktów	czas budowania KDTree	czas budowania QuadTree	czas wyszukiwania KDTree	czas wyszukiwania QuadTree
1000	0.0156	0.0081	0.0003	0.0005
5000	0.1040	0.0561	0.0013	0.0029
10000	0.2458	0.1288	0.0032	0.0073
15000	0.3768	0.2934	0.0045	0.0109
20000	0.5098	0.3333	0.0069	0.0139
25000	0.6423	0.4175	0.0080	0.0171

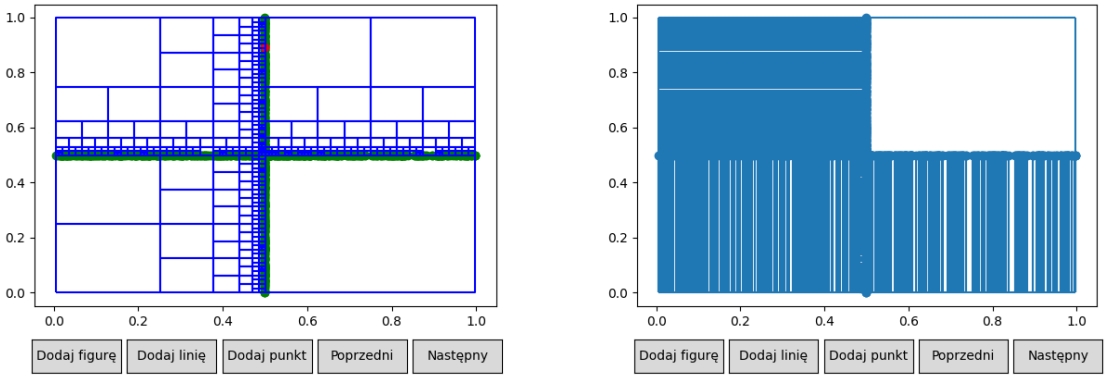
Tabela z porównaniem czasu działania algorytmów. Czas podawany jest w sekundach.



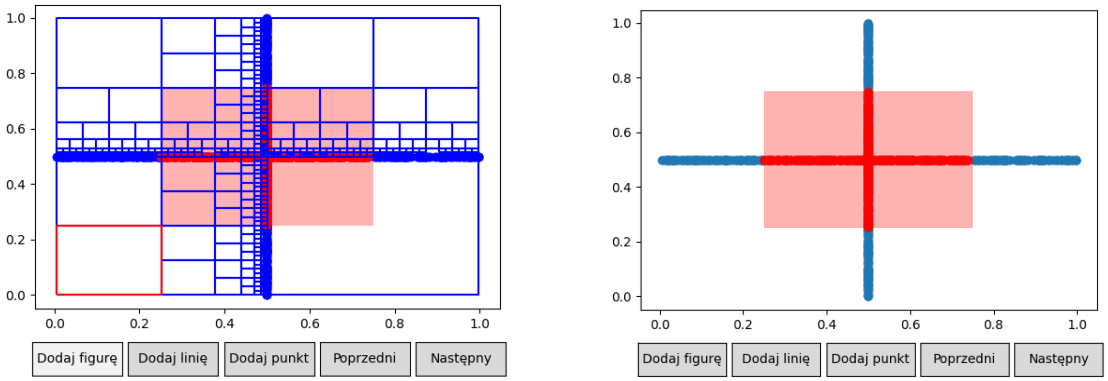
Wyniki porównania czasów budowy struktury danych (lewy wykres) oraz przeszukiwania struktury danych (prawy wykres). Dla różnej ilości punktów

Podobnie jak wcześniej czas budowania jest porównywalny z minimalną przewagą QuadTree oraz czas wyszukiwania jest szybszy przy wykorzystaniu KDtree.

6.Zbiór punktów na krzyżu



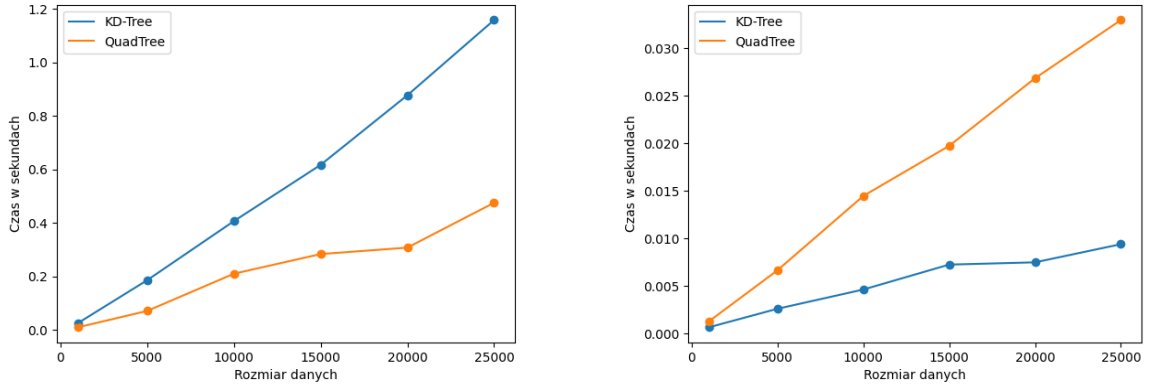
Wizualizacja budowy struktur dla 1000 punktów



Wizualizacja wyniku zapytania dla x1=0.25, y1=0.25, x2=0.75, y2=0.75

liczba punktów	czas budowania KDTree	czas budowania QuadTree	czas wyszukiwania KDTree	czas wyszukiwania QuadTree
1000	0.0252	0.0099	0.0004	0.0011
5000	0.1860	0.0712	0.0016	0.0051
10000	0.4070	0.2104	0.0031	0.0114
15000	0.6174	0.2839	0.0431	0.0203
20000	0.8777	0.3080	0.0538	0.0235
25000	1.1591	0.4757	0.0095	0.0317

Tabela z porównaniem czasu działania algorytmów. Czas podawany jest w sekundach.



Wyniki porównania czasów budowy struktury danych (lewy wykres) oraz przeszukiwania struktury danych (prawy wykres). Dla różnej ilości punktów

Zgodnie z poprzednimi obserwacjacjami inicjowanie struktury jest szybsze dla QuadTree. Przy wyszukiwaniu również jak wcześniej szybsze okazuje się być KDTree.

Wnioski

Z przeprowadzonych testów i zgodnych z oczekiwaniami wyników, wnioskujemy że obie struktury zostały zaimplementowane poprawnie. W każdym z zestawów testowych czas budowania struktur okazał się być szybszy dla struktury QuadTree. W większości zestawów różnica jest nieznaczna, jednak dla punktów na prostokącie QuadTree jest znacznie szybsze jest to zgodne z teoretycznymi założeniami. Złożoność budowania algorytmu QuadTree jest zależna od głębokości h budowanego drzewa i wynosi O(nh) co dla dużego n i równomiernie rozłożonych punktów na bokach prostokąta będzie mniejsze niż O(nlogn).

Z analizy wykresów poszczególnych zbiorów danych przy wyszukiwaniu wynika, że zaimplementowane struktury osiągnęły złożoność lepszą niż liniowa. Pozwala nam to przypuszczać, że programy osiągnęły złożoność oczekiwaną. Szybszą strukturą okazało się być KDTree. Wyszukiwanie w QuadTree trwało znacznie dłużej przy zestawie z dużą ilością punktów gęsto skupionych z niewielką ilością punktów leżących poza skupiskiem. W tym przypadku drzewo jest nie zbalansowane i posiada dużą głębokość. Tego typu zbiór punktów stanowi przypadek w którym struktura QuadTree jest znacznie gorsza.