

# Wyszukiwanie geometryczne

## Przeszukiwanie obszarów ortogonalnych

### QuadTree i KD-Drzewa

### Dokumentacja techniczna projektu

Stanisław Denkowski  
Maciej Trątnowiecki

Grudzień 2019

## 1 Wprowadzenie

W ramach projektu zaliczeniowego przygotowaliśmy implementację struktur KD-Tree i QuadTree pozwalających na przeszukiwanie statycznego zbioru punktów w dwuwymiarowej przestrzeni euklidesowej. Obie implementacje pozwalają na inicjalizację struktury statycznym zbiorem punktów, oraz wyszukiwanie wszystkich punktów należących do zadanego obszaru ortogonalnego.

## 2 Podstawowe informacje techniczne

Implementacje struktur przygotowano w języku python. Pakiet składa się z modułów implementujących omawiane struktury - o odpowiadających im nazwach "quadtree.py", oraz "kdtree.py". Dodatkowo zaimplementowano moduł pomocniczy - "simple\_geometry.py" wykorzystywany przez powyższe, a także "generator.py" odpowiadający za dostarczanie danych testowych, oraz "test.py" realizujący losowe, integracyjne testy automatyczne.

W celu wizualizacji zasady działania algorytmów, dla celów dydaktycznych, wykorzystano moduł "visualiser.py" przygotowany przez mgr inż. Krzysztofa Podsiadło, a także przygotowano notebook "visualiser.ipynb".

W czasie pracy wykorzystywaliśmy wirtualne środowisko conda, którego konfiguracja została zawarta w pliku "env.yml". Korzystanie z tego środowiska nie jest jednak wymagane do użycia implementacji. Lista pakietów wymagających instalacji została zawarta w pliku "REQUIREMENTS.txt". Rozpiska wszystkich wymaganych modułów została wymieniona w poniższym dokumencie.

Projekt znajduje się na githubie: [https://github.com/maciektr/geometric\\_algorithms\\_project](https://github.com/maciektr/geometric_algorithms_project)

### 2.1 Wymagane pakiety

Dla poprawnego działania oprogramowania wymagane jest uruchomienie modułów w środowisku uruchomieniowym zawierającym poniższe moduły.

- |                   |                   |              |
|-------------------|-------------------|--------------|
| • quadtree        | • test            | • visualiser |
| – simple_geometry | – simple_geometry | – matplotlib |
| – numpy           | – kdtree          | – numpy      |
| – enum            | – quadtree        | – json       |
| • kdtree          | – random          |              |
| – simple_geometry | – generator       |              |
| – numpy           |                   |              |
| • simple_geometry | • generator       |              |
| – numpy           | – random          |              |

## 3 QuadTree

Moduł: quadtree

### 3.1 Struktura modułu

Moduł implementuje klasy:

- Quadtree - enkapsulującą implementację drzewa
- Node - reprezentującą węzeł drzewa
- Child - pomocniczego typu wyliczeniowego

A także funkcje:

- create\_kids(node, points, listoflines)  
**Argumenty:**
  - node - Aktualnie rozpatrywany węzeł quadtree, klasy quadtree.Node.
  - points - Zbiór punktów do podziału, klasy simple-geometry.Point.
  - listoflines - Lista list odcinków, przydatna przy wizualizacji. List.

**Wartość zwracana:** brak.

Funkcja nie jest przewidziana jako część interfejsu publicznego modułu.

Wykorzystywana przy konstrukcji drzewa. Dzieli punkty i tworzy odpowiadające temu podziałowi węzły.

- \_get\_lines(node, sol)  
**Argumenty:**
  - node - Aktualnie rozpatrywany węzeł quadtree, klasy quadtree.Node.
  - sol - Przekazywana przez referencję lista odcinków przydatnych przy wizualizacji.

**Wartość zwracana:** brak.

Funkcja wykorzystywana przy tworzeniu wizualizacji drzewa.

- print\_tree(quad, depth)  
**Argumenty:**
  - quad - Aktualnie rozpatrywany węzeł quadtree, klasy quadtree.Node.
  - depth - Głębokość, na jakiej znajduje się aktualnie przetwarzany węzeł drzewa, liczba naturalna.

**Wartość zwracana:** brak.

Funkcja wypisująca tekstową reprezentację drzewa na standardowe wyjście.

### 3.2 Klasa Quadtree

#### 3.2.1 Implementowane metody

- \_\_init\_\_(self, pkts)  
**Konstruktor klasy.**  
**Argumenty:**
  - pkts - Lista, zawierająca statyczny zbiór punktów z płaszczyzny dwuwymiarowej we współrzędnych euklidesowych, reprezentowanych jako dwuelementowy krotki pierwszej i drugiej współrzędnej.

**Wartość zwracana:** brak.

**Złożoność:**  $O((d+1)n)$ , dla  $d$  - głębokość drzewa i  $n$  - liczba punktów

- find(self, x\_low, x\_high, y\_low, y\_high)  
**Argumenty:**
  - x\_low - Lewy kraniec zadanego przedziału prostokątnego względem osi odciętych, liczba całkowita.
  - x\_high - Prawy kraniec zadanego przedziału prostokątnego względem osi odciętych, liczba całkowita.
  - y\_low - Lewy kraniec zadanego przedziału prostokątnego względem osi rzędnych, liczba całkowita.
  - y\_high - Prawy kraniec zadanego przedziału prostokątnego względem osi rzędnych, liczba całkowita.

**Wartość zwracana:** Lista dwuelementowych krotek liczb całkowitych, zawierająca zbiór punktów przechowywanych w drzewie, należących do zadanego przez argumenty funkcji przedziału.

Dodatkowo zwracana jest lista list odcinków przydatna przy wizualizacji przeszukiwania obszaru.

**Złożoność:**  $O(d!)$ , dla  $d$  - głębokość drzewa i  $l$  - liczba liści obejmujących obszar nierozłączny z zadanym. Funkcja odwołuje się do wewnętrznej funkcji realizującej przeszukiwanie Quadtree, stanowiąc dla niej wygodny dla użytkownika interfejs publiczny.

- `_find_points(self, lowerleft, upperright, solution, tree, listoffines)`

**Argumenty:**

- `lowerleft` - Lewy dolny wierzchołek przeszukiwanego obszaru prostokątnego, dwuelementowa krotka liczb całkowitych.
- `upperright` - Prawy górny wierzchołek przeszukiwanego obszaru prostokątnego, dwuelementowa krotka liczb całkowitych.
- `solution` - Set, początkowo pusty.
- `tree` - Aktualnie rozpatrywany węzeł quadtree, klasy `quadtree.Node`, w przypadku korzenia `None`.
- `listoffines` - Lista list odcinków, przydatna przy wizualizacji. List

**Wartość zwracana:** brak.

Funkcja nie jest przewidziana jako część interfejsu publicznego klasy.

Pomocnicza funkcja realizująca przeszukiwanie obszaru ortogonalnego.

- `get_lines(self)` **Argumenty:** brak.

**Wartość zwracana:** Lista odcinków - list dwuelementowych, każdy element to krotka współrzędnych końca odcinka

Funkcja wykorzystywana przy tworzeniu wizualizacji drzewa.

### 3.2.2 Przechowywane dane

Instancja klasy przechowuje w pamięci korzeń odpowiadającego quadtree.

Dodatkowo pamiętana jest lista list odcinków wykorzystywana przy wizualizacji tworzenia drzewa.

## 3.3 Klasa Node

### 3.3.1 Implementowane metody

- `__init__(self, n, w, s, e, par, typ)`

**Konstruktor klasy.**

**Argumenty:**

- `n` - północny kraniec obszaru obejmowanego przez węzeł - maksymalny  $y$
- `w` - zachodni kraniec obszaru obejmowanego przez węzeł - minimalne  $x$
- `s` - południowy kraniec obszaru obejmowanego przez węzeł - minimalny  $y$
- `e` - wschodni kraniec obszaru obejmowanego przez węzeł - maksymalny  $x$
- `par` - wskaźnik do rodzica obecnego węzła
- `typ` - informacja czy dany wierzchołek jest korzeniem lub którym synem - NE, NW, SE, SW

**Wartość zwracana:** brak.

- `add_kid(self, nr, other)`

**Argumenty:**

- `nr` - numer wskazujący na typ dziecka obecnego węzła, według typu wyliczeniowego `Child`
- `other` - wskaźnik na nowy węzeł, będący konkretnym dzieckiem obecnego węzła

**Wartość zwracana:** brak.

- `get_kid(self, nr)`

**Argumenty:**

- `nr` - Indeks poddrzewa, liczba naturalna.

**Wartość zwracana:** Poddrzewo o zadanym indeksie w węźle, klasy quadtree.Node.  
Funkcja zwraca poddrzewo danego węzła o zadanym indeksie.

- `__str__(self)`

**Argumenty:** brak.

**Wartość zwracana:** Łańcuch znaków.

Funkcja zwraca reprezentację instancji klasy w postaci łańcucha znaków.

### 3.3.2 Przechowywane dane

Instancja klasy przechowuje w pamięci informacje o obszarze obejmowanym przez dany węzeł, dane ułatwiające podział na dzieci, informację o typie węzła (czy korzeń lub które dziecko), informację o liczbie dzieci danego węzła, wskaźniki na rodzica, dzieci i punkt (znajdujący się w danym obszarze, dal liścia) o ile takie istnieją.

## 4 KD-Drzewa

**Moduł:** kdtree

### 4.1 Struktura modułu

Moduł implementuje klasy:

- Kdtree - Enkapsulującą implementację drzewa.
- Node - Reprezentującą węzeł kd-drzewa.

### 4.2 Klasa Kdtree

#### 4.2.1 Implementowane metody

- `__init__(self, points)`

**Konstruktor klasy.**

**Argumenty:**

- points - Lista, zawierająca statyczny zbiór punktów z płaszczyzny dwuwymiarowej we współrzędnych euklidesowych, reprezentowanych jako dwuelementowy krotki pierwszej i drugiej współrzędnej.

**Wartość zwracana:** brak.

**Złożoność:**  $O(n \log n)$

- `_construct(self, points, depth=0)`

**Argumenty:**

- points - Zbiór punktów do podziału, klasy `simple_geometry.Point`.
- depth - Głębokość, na jakiej znajduje się aktualnie tworzony węzeł drzewa, domyślnie 0, liczba naturalna.

**Wartość zwracana:** Skonstruowany korzeń drzewa, klasy `kdtree.Node`.

**Złożoność:**  $O(n \log n)$

Pomocnicza funkcja rekurencyjna, konstruuje kd-drzewo i zwracając jego korzeń.

- `find(self, x_low, x_high, y_low, y_high)`

**Argumenty:**

- x\_low - Lewy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie `-numpy.inf` (reprezentacja nieskończoności), liczba całkowita.
- x\_high - Prawy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie `numpy.inf` (reprezentacja nieskończoności), liczba całkowita.
- y\_low - Lewy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie `-numpy.inf` (reprezentacja nieskończoności), liczba całkowita.
- y\_high - Prawy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie `numpy.inf` (reprezentacja nieskończoności), liczba całkowita.

**Wartość zwracana:** Lista dwuelementowych krotek liczb całkowitych, zawierająca zbiór punktów przechowywanych w drzewie, należących do zadanego przez argumenty funkcji przedziału.

**Złożoność:**  $O(\sqrt{n} + k)$ , gdzie  $k$  jest licznością zbioru wynikowego.

Funkcja odwołuje się do wewnętrznej funkcji realizującej przeszukiwanie kd-drzewa, stanowiąc dla niej wygodny dla użytkownika interfejs publiczny.

#### 4.2.2 Przechowywane dane

- scope - Najmniejszy obszar prostokątny, który zawiera wszystkie punkty przechowywane w drzewie, klasy simple\_geometry.Scope.
- root - Korzeń drzewa, klasy kdtree.Node.

### 4.3 Klasa Node

#### 4.3.1 Implementowane metody

- `__init__(self, point, line, left, right)`

**Konstruktor klasy.**

**Argumenty:**

- point - Dwuelementowa krotka liczb całkowitych, odpowiadająca punktowi z płaszczyzny dwuwymiarowej we współrzędnych euklidesowych.
- line - Współrzędna (liczba całkowita) prostej prostopadłej do osi układu, domyślnie None.
- left - Lewe poddrzewo, klasy kdtree.Node, domyślnie None.
- right - Prawe poddrzewo, klasy kdtree.Node, domyślnie None.

**Wartość zwracana:** brak.

- `report_subtree(self)`

**Argumenty:** Brak.

**Wartość zwracana:** Lista obiektów klasy simple\_geometry.Point

Funkcja zwracająca wszystkie punkty przechowywane w danym poddrzewie.

- `_search(self, scope, actual_scope, depth)` **Argumenty:**

- scope - Obszar, z którego punkty chcemy otrzymać, klasy simple\_geometry.Scope.
- actual\_scope - Obszar w jakim znajdują się punkty z aktualnie rozpatrywanego poddrzewa, klasy simple\_geometry.Scope, domyślnie wywołuje konstruktor tej klasy, bez wskazania żadnych argumentów.
- depth - Głębokość, na jakiej znajduje się aktualnie przetwarzany węzeł drzewa, domyślnie 0, liczba naturalna.

**Wartość zwracana:** Lista obiektów klasy simple\_geometry.Point

Funkcja realizująca rekurencyjnie algorytm przeszukiwania kd-drzewa na poziomie węzła.

- `check_child(self, child, actual_scope, depth, scope)` **Argumenty:**

- child - Aktualnie rozpatrywane poddrzewo, klasy kdtree.Node.
- actual\_scope - Obszar w jakim znajdują się punkty z aktualnie rozpatrywanego poddrzewa, klasy simple\_geometry.Scope.
- depth - Głębokość, na jakiej znajduje się aktualnie przetwarzany węzeł drzewa, domyślnie 0, liczba naturalna.
- scope - Obszar, z którego punkty chcemy otrzymać, klasy simple\_geometry.Scope.

**Wartość zwracana:** Lista obiektów klasy Funkcja nie jest przewidziana jako część interfejsu publicznego klasy.

Jest to funkcja pomocnicza metody `_search`.

### 4.3.2 Przechowywane dane

- point - Jeżeli węzeł jest liściem, to przechowuje odpowiadający mu punkt płaszczyzny, w postaci instancji klasy `simple.geometry.Point`, w przeciwnym przypadku `None`.
- line - Jeśli węzeł nie jest liściem, przechowuje współrzędną (liczba całkowita) prostej prostopadłej do osi układu, podziałowi względem której odpowiada, w przeciwnym wypadku `None`.
- left - Lewe poddrzewo, klasy `kdtree.Node`, lub `None`.
- right - Prawe poddrzewo, klasy `kdtree.Node`, lub `None`.

## 5 Prosta geometria

Moduł: `simple_geometry`

### 5.1 Klasa Point

#### 5.1.1 Implementowane metody

- `__init__(self,s)`  
**Konstruktor klasy.**  
**Argumenty:**
  - s - Dwuelementowa krotka liczb całkowitych, odpowiadająca punktowi z płaszczyzny dwuwymiarowej we współrzędnych euklidesowych.  
**Wartość zwracana:** brak.
- `get_tuple(self)`  
**Argumenty:** brak.  
**Wartość zwracana:** Dwuelementowa krotka liczb całkowitych  
Funkcja zwraca przechowywany punkt w postaci dwuelementowej krotki liczb całkowitych.
- `__str__(self)`  
**Argumenty:** brak.  
**Wartość zwracana:** Łańcuch znaków.  
Funkcja zwraca reprezentację instancji klasy w postaci łańcucha znaków.

#### 5.1.2 Przechowywane dane

- x\_low - Lewy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie `-numpy.inf` (reprezentacja nieskończoności), liczba całkowita.
- x\_high - Prawy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie `numpy.inf` (reprezentacja nieskończoności), liczba całkowita.
- y\_low - Lewy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie `-numpy.inf` (reprezentacja nieskończoności), liczba całkowita.
- y\_high - Prawy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie `numpy.inf` (reprezentacja nieskończoności), liczba całkowita.

### 5.2 Klasa Scope

#### 5.2.1 Implementowane metody

- `__init__(self, xl, xh, yl, yh)`  
**Konstruktor klasy.**  
**Argumenty:**
  - xl - Lewy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie `-numpy.inf` (reprezentacja nieskończoności), liczba całkowita.
  - xh - Prawy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie `numpy.inf` (reprezentacja nieskończoności), liczba całkowita.

- yl - Lewy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie -numpy.inf (reprezentacja nieskończoności), liczba całkowita.
- yh - Prawy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie numpy.inf (reprezentacja nieskończoności), liczba całkowita.

**Wartość zwracana:** brak.

- `__str__(self)`

**Argumenty:** brak.

**Wartość zwracana:** Łańcuch znaków.

Funkcja zwraca reprezentację instancji klasy w postaci łańcucha znaków.

- `from_tuple(self, lowerleft, upperright)`

**Argumenty:**

- lowerleft - Lewy dolny wierzchołek przeszukiwanego obszaru prostokątnego, dwuelementowa krotka liczb całkowitych.
- upperright - Prawy górny wierzchołek przeszukiwanego obszaru prostokątnego, dwuelementowa krotka liczb całkowitych.

**Wartość zwracana:** `simple_geometry.Scope`

Funkcja aktualizuje zawartość przechowywanych pól, zgodnie z zadanym w postaci pary krotek obszarem.

- `in_scope(self, point)`

**Argumenty:**

- point - Punkt na płaszczyźnie, reprezentowany przez instancję klasy `simple_geometry.Point`

**Wartość zwracana:** Wartość logiczna.

Metoda odpowiada na pytanie, czy punkt podany jako argument należy do reprezentowanego obszaru.

- `contains(self, other)`

**Argumenty:**

- other - Obszar na płaszczyźnie, reprezentowany przez instancję klasy `simple_geometry.Scope`

**Wartość zwracana:** Wartość logiczna.

Metoda odpowiada na pytanie, czy obszar podany jako argument zawiera się w całości w reprezentowanym obszarze.

- `intersects(self, other)`

**Argumenty:**

- other - Obszar na płaszczyźnie, reprezentowany przez instancję klasy `simple_geometry.Scope`

**Wartość zwracana:** Wartość logiczna.

Metoda odpowiada na pytanie, czy obszar podany jako argument ma niezerowe przecięcie z reprezentowanym obszarem.

- `common(self, x_low, x_high, y_low, y_high)`

**Argumenty:**

- xl - Lewy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie None (reprezentacja nieskończoności), liczba całkowita.
- xh - Prawy kraniec zadanego przedziału prostokątnego względem osi odciętych, domyślnie None (reprezentacja nieskończoności), liczba całkowita.
- yl - Lewy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie None (reprezentacja nieskończoności), liczba całkowita.
- yh - Prawy kraniec zadanego przedziału prostokątnego względem osi rzędnych, domyślnie None (reprezentacja nieskończoności), liczba całkowita.

**Wartość zwracana:** brak.

Metoda realizująca operację przecięcia reprezentowanego obszaru, z obszarem podanym jako argument funkcji (gdzie nie występuje konieczność wykorzystania wszystkich argumentów, dla przykładu (`x_low = 10`) reprezentuje półpłaszczyznę na prawo od prostej `x=10`).

- `copy(self, other)`

**Argumenty:**

- `other` - Obszar na płaszczyźnie, reprezentowany przez instancję klasy `simple_geometry.Scope`

**Wartość zwracana:** Wartość logiczna.

Metoda kopiuje obszar podany jako argument. Od teraz staje się on nowym obszarem reprezentowanym przez instancję klasy.

### 5.2.2 Przechowywane dane

- `x` - Pierwsza współrzędna przechowywanego punktu, liczba całkowita.
- `y` - Druga współrzędna przechowywanego punktu, liczba całkowita.

## 6 Przykłady użycia

```
import generator
test1 = generator.test_case_1()
scope = (10,100,10,60)

# QuadTree
import quadtree
tree = quadtree.Quadtree(test1)
solution = tree.find(*scope)

# KDTree
import kdtree
kd = kdtree.Kdtree(test1)
solution_kd = kd.find(*scope)
```