



Geometria Obliczeniowa

Porównanie KD-tree i Quad Tree

KD-Tree

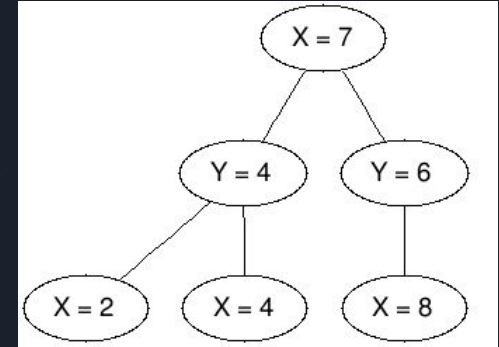
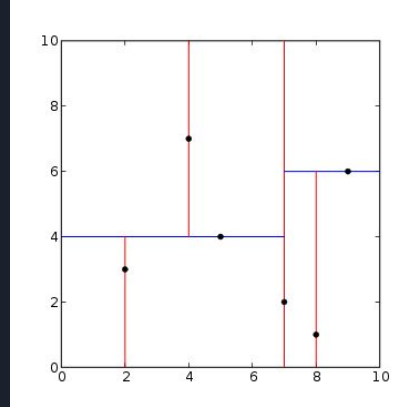
n - liczba punktów

Struktura:

- Zbalansowane drzewo binarne (po stworzeniu)
- Każdy węzeł dzieli punkty w zadanej podprzestrzeni na 2
- Głębokość: $\log_2 n$

Właściwości:

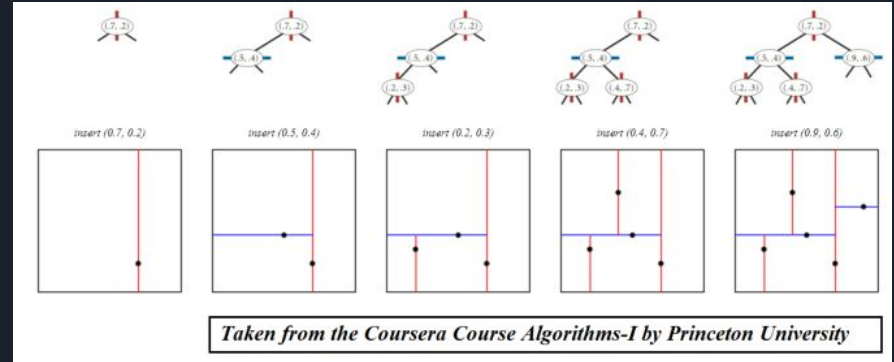
- Szybkie poszukiwanie najbliższego sąsiada danego punktu
- Wydajne szukanie punktów na zadanym prostokątnym obszarze
- Balansowanie drzewa po dodaniu nowego elementu zmienia strukturę całego drzewa
- Możliwość rozszerzenia do większej liczby wymiarów



KD-Tree - tworzenie

```
twórz_drzewo_KD(punkty, orientacja):  
    punkty = sort(punkty, key=orientacja)  
    return węzeł = Węzeł_KD(  
        punkty.mediana,  
        punkty < mediana,  
        punkty > mediana,  
        przeciwna(orientacja)  
    )
```

```
Węzeł_KD(mediana, mniejsze, większe, orientacja):  
    this.dane = punkt  
    this.mniejsze = twórz_drzewo_KD(mniejsze, orientacja)  
    this.większe = twórz_drzewo_KD(większe, orientacja)  
    this.orientacja = orientacja
```



KD-Tree - wyszukiwanie

znajdź_w_drzewie_KD(przedział, węzeł):

jeśli przedział leży na granicy hiperpłaszczyzny węzła:

result += znajdź_w_drzewie_KD(przedział, węzeł.lewe_dziecko)

result += znajdź_w_drzewie_KD(przedział, węzeł.prawe_dziecko)

w przeciwnym przypadku:

jeśli przedział leży po lewej stronie hiperpłaszczyzny węzła

result += znajdź_w_drzewie_KD(przedział, węzeł.lewe_dziecko)

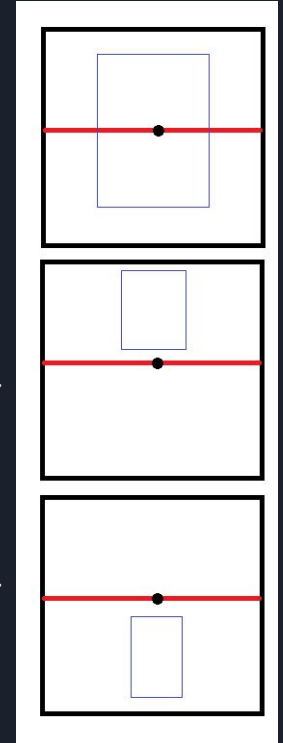
w przeciwnym przypadku:

result += znajdź_w_drzewie_KD(przedział, węzeł.prawe_dziecko)

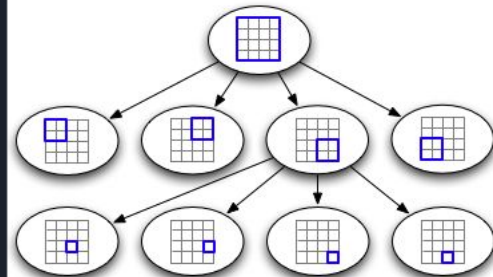
jeśli węzeł.dane jest w przedział:

result += węzeł

return result

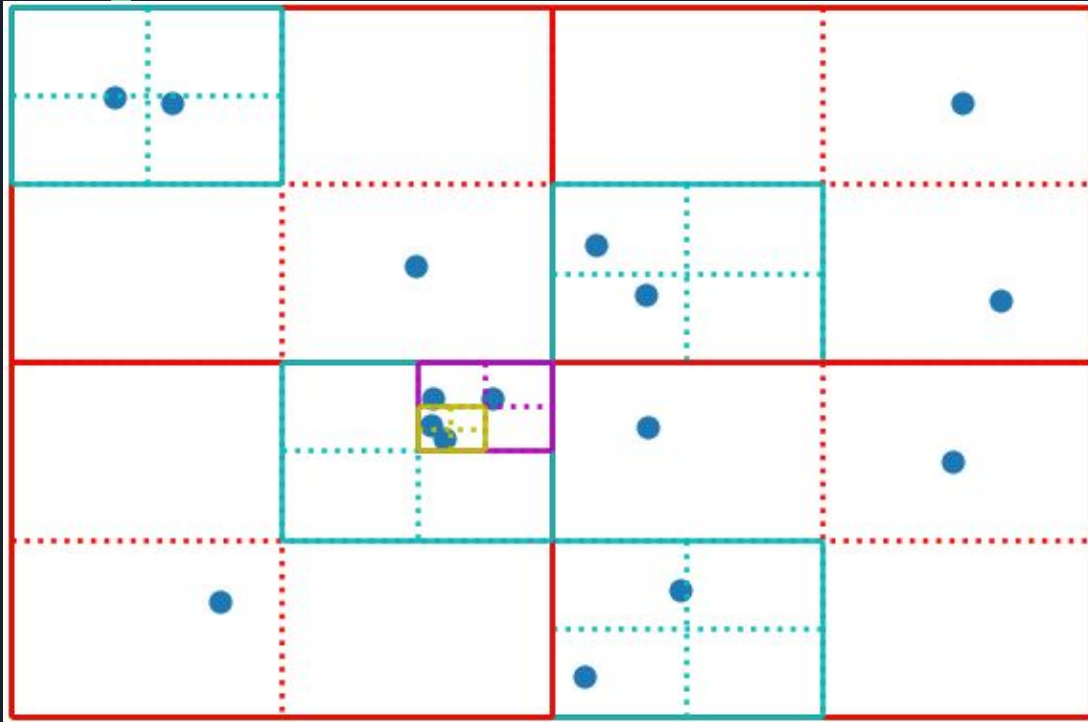


Quad-Tree

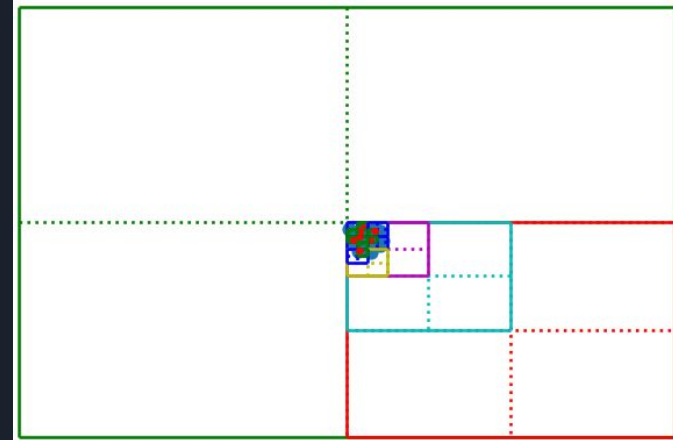


- Drzewo dzieli przestrzeń, dodanie wielu punktów o identycznych współrzędnych spowoduje błąd.
- Każdy węzeł quad-drzewa dzieli przestrzeń 4 równe części, posiada swoją pozycję, która jest środkiem podziału.
- Liście quad-drzewa zawierają listę punktów, mają ograniczony maksymalny rozmiar.
- Drzewo tworzone jest przez dodawanie kolejnych punktów i umieszczanie ich w odpowiednich liściach.
- W podobny sposób można konstruować podział przestrzeni w większej liczbie wymiarów. Dla 3 wymiarów jest to octree, w którym każdy wierzchołek dzieli przestrzeń na 8 sześcianów

Quad-Tree - podział przestrzeni



Każdy liść zawiera 1 punkt.
Dodając punkty w jednym miejscu można spowodować, że drzewo nie będzie zbalansowane.





Quad-Tree - inne zastosowania

Quad drzewa można również wykorzystać do przetwarzania obrazów i kompresji



Quad-Tree - tworzenie

class QT_Leaf - liść drzewa

class QT_Node - wierzchołek drzewa

QT_Node::dodaj_punkt(punkt):

znajdź odpowiedniego potomka dla punktu

jeśli ten potomek jest liściem i nie może przyjąć dodatkowego punktu:

pobierz wszystkie punkty z potomka

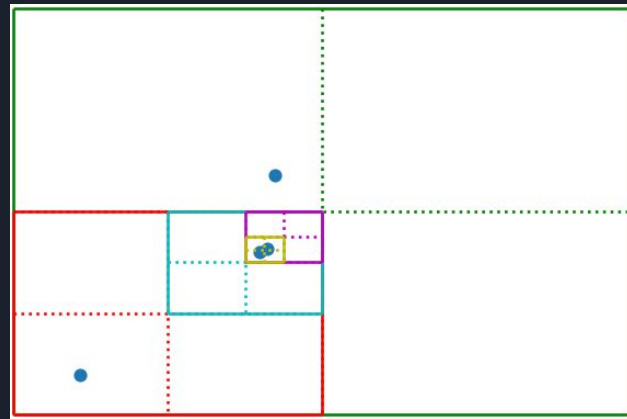
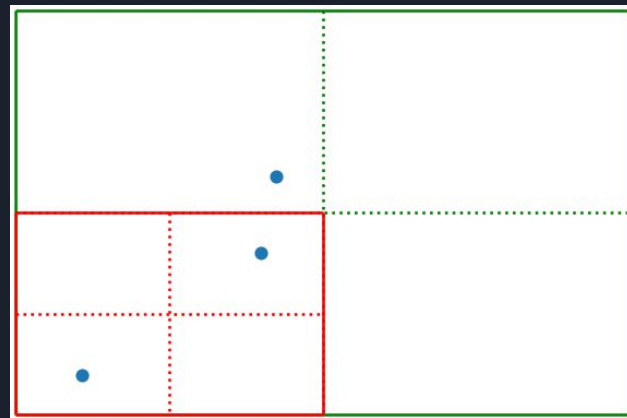
zamień potomka na obiekt QT_Node z odpowiednim środkiem

dodaj do potomka wszystkie punkty pobrane wcześniej

dodaj punkt do potomka

w przeciwnym przypadku:

dodaj punkt do potomka



Quad-Tree - wyszukiwanie

def QT_Node::znajdź_punkty_w_prostokącie(prostokąt):

jeśli punkt centralny zawiera się w prostokącie:
zwróć znalezione punkty z każdego potomka

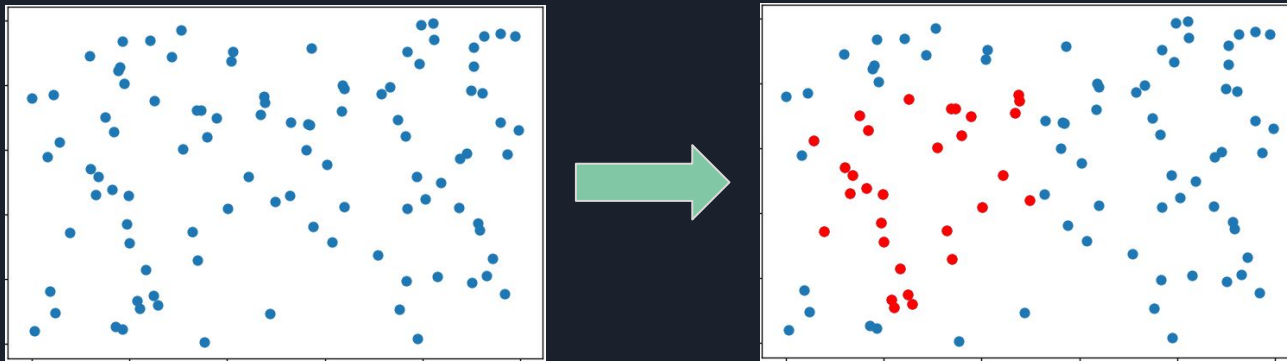
w przeciwnym przypadku:

dla każdego potomka:

kwadrat potomka - kwadrat, który należy do potomka, jego punkt centralny \pm zasięg

jeśli przecięcie kwadratu potomka z prostokątem jest nie puste:

zwróć znalezione punkty z tego potomka






Porównanie

Porównanie czasu wykonania algorytmów dla problemu znajdowania punktów w danym obszarze. Liczba punktów - 100000 z przedziału $[0,0],[1000,1000]$, wyszukiwanie punktów z przedziału $[200,100],[400,400]$. Czasy mierzone są w milisekundach.

	Czas tworzenia struktur [s]	Czas znajdowania punktów w przedziale	Czas znajdowania 1 punktu	Czas znajdowania wszystkich punktów
KD Tree	2.04	14.4	0.027	231.9
Quad Tree	3.45	23.1	0.21	314.9
Sprawdzanie listy	N/D	40.4	24.2	74.0



Szymon Czarny, Maciej Mucha