Wyszukiwanie geometryczne

Przeszukiwanie obszarów ortogonalnych - drzewa obszarów

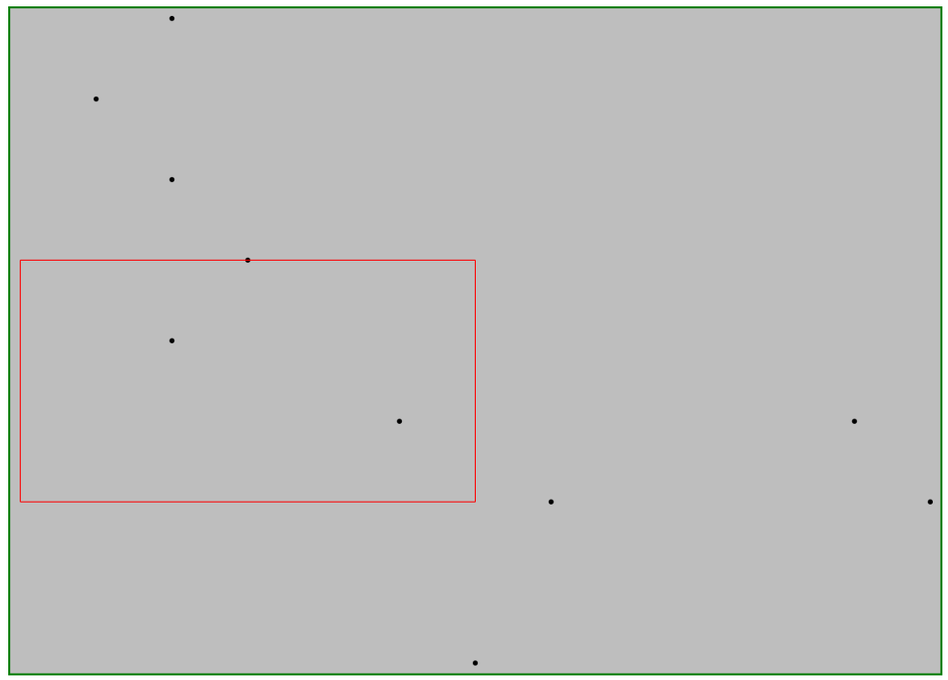
Michał Fudała

Opis problemu

Dane: zbiór punktów **P** na płaszczyźnie.

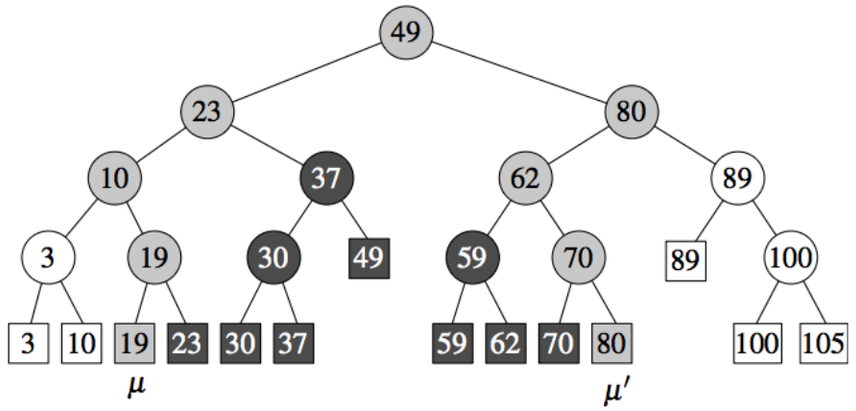
Problem: dla zadanych **x1**, **x2**, **y1**, **y2** znaleźć punkty q ze zbioru **P** takie, że **x1** ≤ **qx** ≤ **x2** ,**y1** ≤ **qy** ≤ **y2**

Opis problemu



Opis algorytmu

1D - budowa drzewa



Opis algorytmu

1D - budowa drzewa

Budujemy zrównoważone drzewo binarne gdzie liśćmi są punkty wejściowe.

Algorytm jest następujący:

Wejście: **P** -**punkty na linii (posortowane)**, **x_beg**, **x_end**

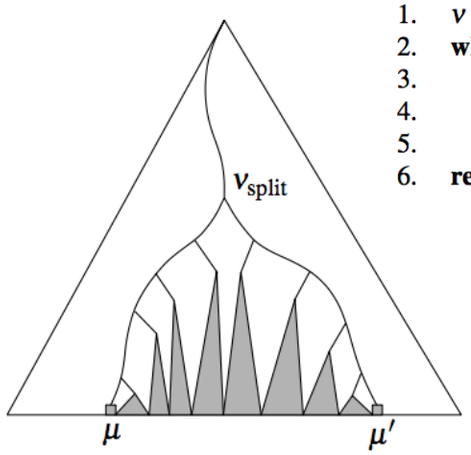
1) stwórz węzeł drzewa **N**, którego wartością jest **mediana** z **P[x_beg:x_end]**

2) wywołaj metodę rekursywnie dla **P**, **x_end**=pozycja od której wszystkie punkty są mniejsze bądz równe medianie i przypisz je do **N->left**

3) wywołaj metodę rekursywnie dla **P**, **x_beg**=pozycja od której wszystkie punkty są większe od mediany i przypisz je do **N->right**

4) return **N**

Opis algorytmu
1D - wyszukiwanie węzła dzielącego

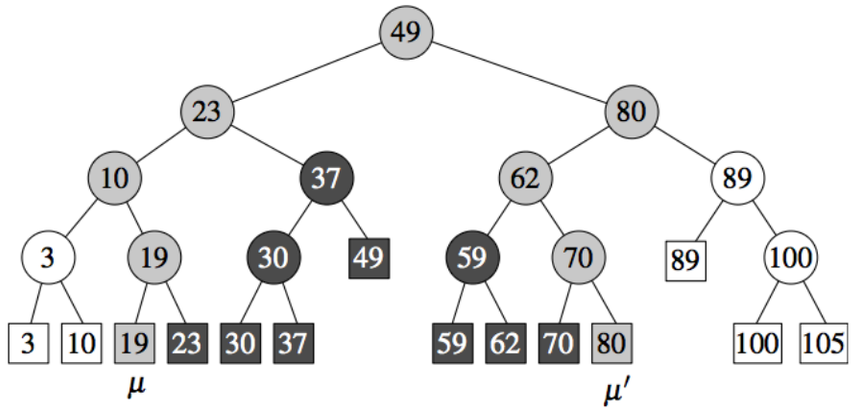


FINDSPLITNODE($\mathcal{T}, x, x'$)
Input. A tree $\mathcal{T}$ and two values x and x' with $x \leqslant x'$.
Output. The node v where the paths to x and x' split, or the leaf where both paths end.

1. $v \leftarrow root(\mathcal{T})$
2. **while** v is not a leaf **and** ($x' \leqslant x_v$ **or** $x > x_v$)
3. **do if** $x' \leqslant x_v$
4. **then** $v \leftarrow lc(v)$
5. **else** $v \leftarrow rc(v)$
6. **return** v

pseudokod z książki **Computational Geometry: Algorithms and Applications** - Mark de Berg

Opis algorytmu
1D - wyszukiwanie w drzewie



Opis algorytmu

1D - wyszukiwanie w drzewie

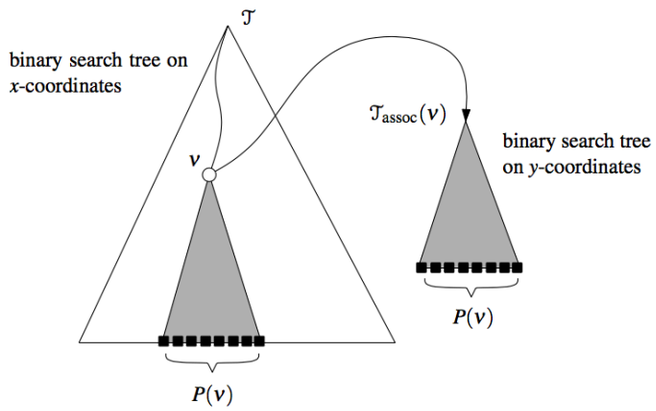
Algorithm 1DRANGEQUERY($\mathcal{T}, [x : x']$)
Input. A binary search tree $\mathcal{T}$ and a range $[x : x']$.
Output. All points stored in $\mathcal{T}$ that lie in the range.

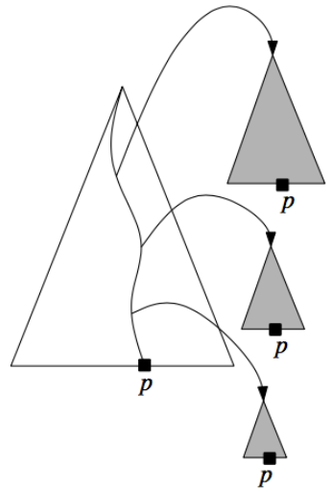
```
1.  v_split ← FINDSPLITNODE(T, x, x')
2.  if v_split is a leaf
3.    then Check if the point stored at v_split must be reported.
4.    else (* Follow the path to x and report the points in subtrees right of the
           path. *)
5.         v ← lc(v_split)
6.         while v is not a leaf
7.           do if x ⩽ x_v
8.                then REPORTSUBTREE(rc(v))
9.                     v ← lc(v)
10.               else v ← rc(v)
11.        Check if the point stored at the leaf v must be reported.
12.        Similarly, follow the path to x', report the points in subtrees left of
           the path, and check if the point stored at the leaf where the path
           ends must be reported.
```

pseudokod z książki **Computational Geometry: Algorithms and Applications** - Mark de Berg

Opis algorytmu

2D - budowa drzewa





Opis algorytmu

2D - budowa drzewa

Algorithm BUILD2DRANGETREE(P)
Input. A set P of points in the plane.
Output. The root of a 2-dimensional range tree.

1. Construct the associated structure: Build a binary search tree $\mathcal{T}_{assoc}$ on the set P_y of y-coordinates of the points in P. Store at the leaves of $\mathcal{T}_{assoc}$ not just the y-coordinate of the points in P_y, but the points themselves.
2. **if** P contains only one point
3. **then** Create a leaf v storing this point, and make $\mathcal{T}_{assoc}$ the associated structure of v.
4. **else** Split P into two subsets; one subset P_{left} contains the points with x-coordinate less than or equal to x_{mid}, the median x-coordinate, and the other subset P_{right} contains the points with x-coordinate larger than x_{mid}.
5. $v_{left} \leftarrow$ BUILD2DRANGETREE(P_{left})
6. $v_{right} \leftarrow$ BUILD2DRANGETREE(P_{right})
7. Create a node v storing x_{mid}, make v_{left} the left child of v, make v_{right} the right child of v, and make $\mathcal{T}_{assoc}$ the associated structure of v.
8. **return** v

pseudokod z książki **Computational Geometry: Algorithms and Applications** - Mark de Berg

Opis algorytmu

2D - wyszukiwanie w drzewie

Algorithm 2DRANGEQUERY($\mathcal{T}, [x:x'] \times [y:y']$)
Input. A 2-dimensional range tree $\mathcal{T}$ and a range $[x:x'] \times [y:y']$.
Output. All points in $\mathcal{T}$ that lie in the range.

1. $v_{\text{split}} \leftarrow$FINDSPLITNODE($\mathcal{T}, x, x'$)
2. **if** v_{split} is a leaf
3. **then** Check if the point stored at v_{split} must be reported.
4. **else** (∗ Follow the path to x and call 1DRANGEQUERY on the subtrees right of the path. ∗)
5. $v \leftarrow lc(v_{\text{split}})$
6. **while** v is not a leaf
7. **do if** $x \leqslant x_v$
8. **then** 1DRANGEQUERY($\mathcal{T}_{\text{assoc}}(rc(v)), [y:y']$)
9. $v \leftarrow lc(v)$
10. **else** $v \leftarrow rc(v)$
11. Check if the point stored at v must be reported.
12. Similarly, follow the path from $rc(v_{\text{split}})$ to x', call 1DRANGEQUERY with the range $[y:y']$ on the associated structures of subtrees left of the path, and check if the point stored at the leaf where the path ends must be reported.

pseudokod z książki **Computational Geometry: Algorithms and Applications** - Mark de Berg

Opis algorytmu
Ogólny zbiór punktów

Problemem opisanego wcześniej algorytmu jest to, że nie działa poprawnie dla dwóch punktów o takich samych wartościach **x** lub **y**.

Opis algorytmu

Ogólny zbiór punktów

Każdy punkt $p = (p_x, p_y)$ możemy reprezentować jako:

$$\hat{p} = ((p_x|p_y),\ (p_y|p_x))$$

Porządek zadany może być w następujący sposób:

$$(a|b) < (\acute{a}|\acute{b}) \Leftrightarrow a < \acute{a} \vee (a = \acute{a} \wedge b < \acute{b})$$

Zbiór wejściowy algorytmu po przekształceniu:

$$\hat{R} = [(x|-\infty), (x'|+\infty)] \times [(y|-\infty), (y'|+\infty)]$$

Złożoność

n - liczba wejściowych punktów

k - liczba punktów na wyjściu

Budowa drzewa (po posortowaniu danych po x oraz po y)

$O(nlogn)$

Wyszukiwanie w drzewie

$O(log^2n+k)$

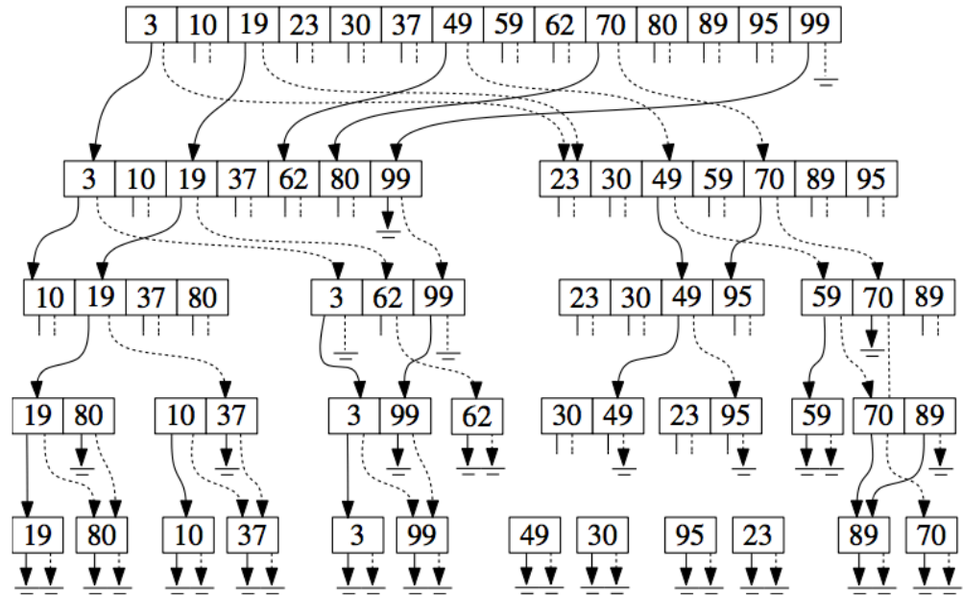
Złożoność - udoskonalenia

Można poprawić złożoność wyszukiwania poprzez ulepszenie algorytmu o Fractional Cascading (kaskadowanie cząstkowe)

Wyszukiwanie w drzewie

O(logn+k)

Fractional Cascading



Bibliografia

- **Computational Geometry: Algorithms and Applications** - Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Over-mars

- **https://www.ii.uni.wroc.pl/~prz/2010zima/zsd/zasoby/DrzewaObszarow.pdf**