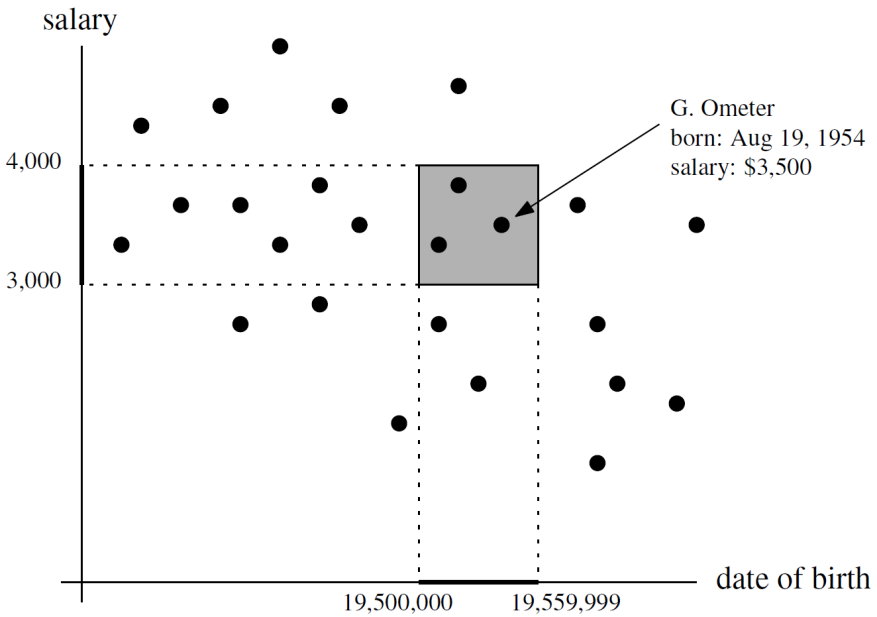
Geometria obliczeniowa

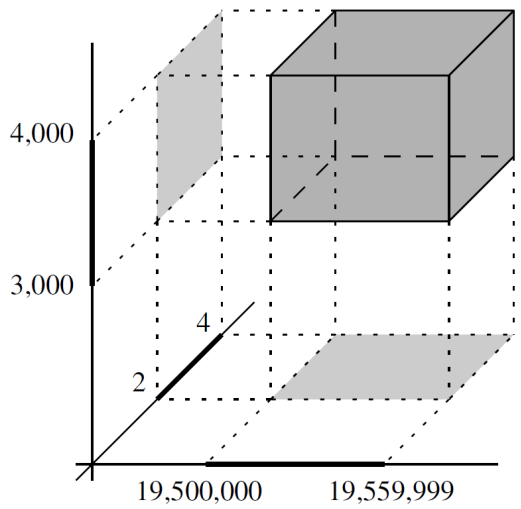
Krzysztof Gdawiec

UNIWERSYTET ŚLĄSKI
INSTYTUT INFORMATYKI

Przeszukiwanie obszarów ortogonalnych

Wiele typów zapytań w bazach danych możemy interpretować geometrycznie. W tym celu przekształcamy rekordy w bazie danych w punkty w wielowymiarowej przestrzeni i zmieniamy zapytania o rekordy na zapytania o odpowiedni zbiór punktów.





Jeśli jesteśmy zainteresowani odpowiedzią na zapytanie dotyczące d pól rekordów przekształcamy rekordy w punkty w d-wymiarowej przestrzeni.

Zapytanie o podanie wszystkich rekordów, których pola leżą między podanymi wartościami zamienia się wówczas na pytanie dotyczące wszystkich punktów wewnątrz d-wymiarowego prostopadłościanu o bokach równoległych do osi.

Takie zapytanie nazywamy w geometrii obliczeniowej zapytaniem o obszar prostokątny (ang. rectangular range query) lub zapytaniem o obszar ortogonalny (ang. orthogonal range query).

Przeszukiwanie obszarów jednowymiarowych

Niech $P = \{p_1, p_2, \ldots, p_n\}$ będzie danym zbiorem punktów na prostej rzeczywistej oraz $[x, x']$ będzie przedziałem definiującym nasze zapytanie.

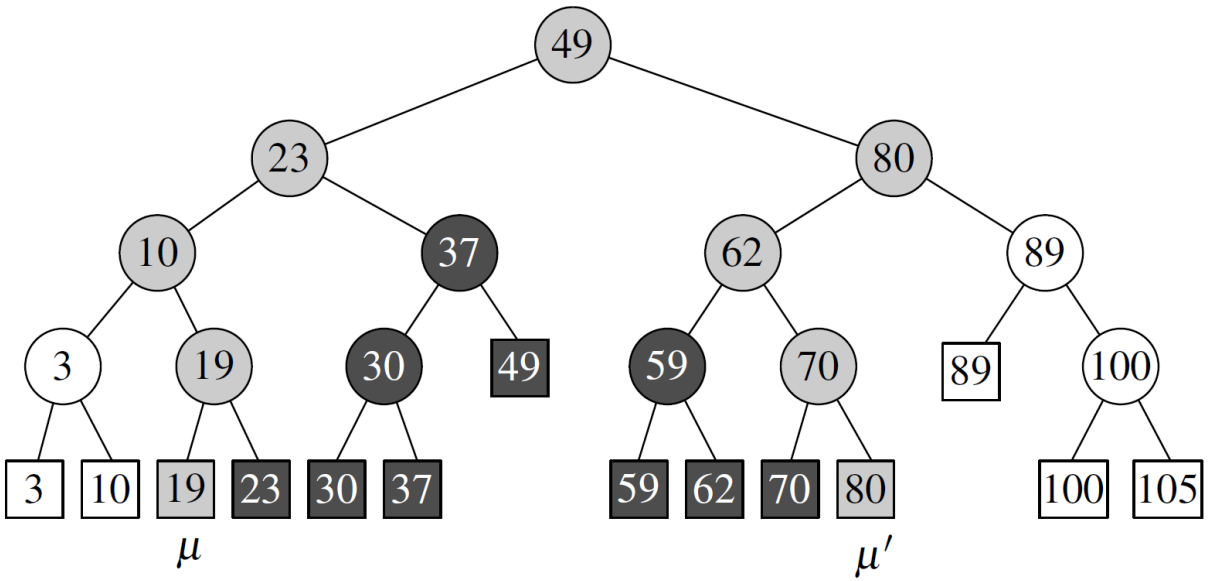
Nasz problem możemy efektywnie rozwiązać stosując zrównoważone drzewo BST $\mathcal{T}$. Liście $\mathcal{T}$ pamiętają punkty P, a węzły wewnętrzne przechowują wartości dzielące, które sterują przeszukiwaniem.

Oznaczmy wartość dzielącą pamiętaną w węźle v przez x_v. Zatem lewe poddrzewo węzła v zawiera wszystkie punkty mniejsze lub równe x_v, a prawe poddrzewo wszystkie punkty większe od x_v.

W celu znalezienia punktów z przedziału $[x, x']$ najpierw poszukujemy x i x' w $\mathcal{T}$. Niech μ i μ' będą liśćmi, w których kończy się przeszukiwanie.

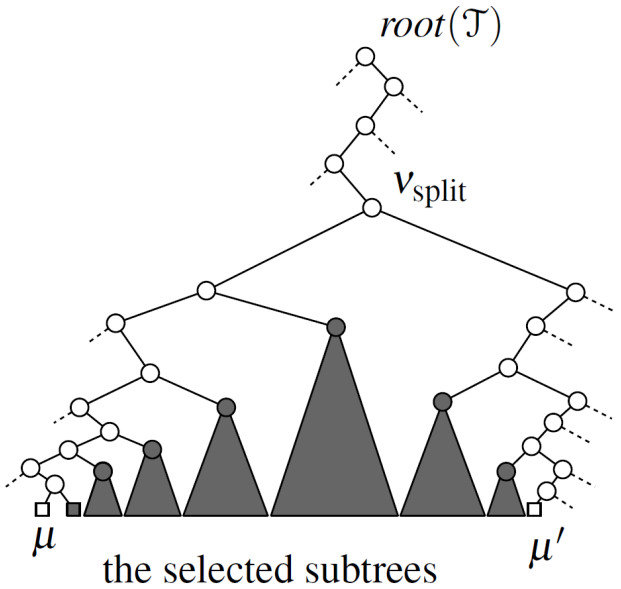
Punkty z przedziału $[x, x']$ są punktami pamiętanymi w liściach między μ i μ' oraz być może punktem pamiętanym w μ i μ'.

Przykład: Zapytanie o przedział $[18, 77]$.



Jak znaleźć liście pomiędzy μ i μ'?

Liście te są liśćmi pewnych poddrzew między ścieżkami przeszukiwań do μ i μ' (ciemno szary kolor na rysunku z poprzedniego slajdu).
Poddrzewa, które wybieramy są zakorzenione w węzłach między dwoma ścieżkami przeszukiwań, których rodzice są na ścieżce przeszukiwań. Aby znaleźć te węzły poszukujemy węzła v_{split}, w którym ścieżki do x i x' rozchodzą się.



Niech $lc(v)$ i $rc(v)$ oznaczają odpowiednio lewe i prawe dziecko węzła v.

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

Mając v_{split} idziemy dalej ścieżką przeszukiwań x. W każdym węźle, z którego ścieżka skręca w lewo wyliczamy wszystkie liście z jego prawego poddrzewa.

Podobnie idziemy ścieżką x' i wyliczamy liście w lewym poddrzewie węzłów, w których ścieżka skręca w prawo. Na końcu sprawdzamy punkty pamiętane w liściach, w których ścieżki kończą się.

W algorytmie użyjemy funkcji ReportSubtree, która przechodzi poddrzewo zakorzenione w danym węźle i wylicza punkty pamiętane w jego liściach.

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.

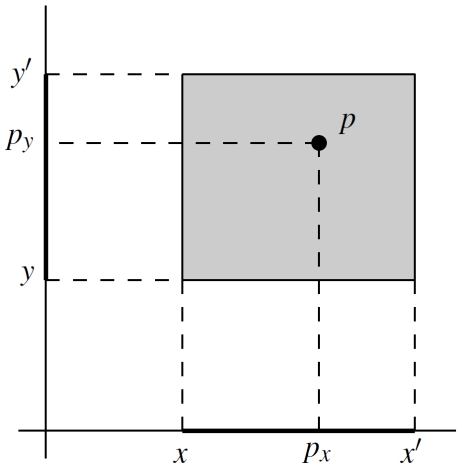
```
1.  v_split ← FINDSPLITNODE(T, x, x')
2.  if v_split is a leaf
3.     then Check if the point stored at v_split must be reported.
4.     else (* Follow the path to x and report the points in subtrees right of the path. *)
5.          v ← lc(v_split)
6.          while v is not a leaf
7.             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.
```

Niech P będzie zbiorem n punktów w jednowymiarowej przestrzeni. Zbiór P można zapamiętać w zrównoważonym drzewie wyszukiwań binarnych, które korzysta z pamięci $\mathcal{O}(n)$ i ma czas konstrukcji $\mathcal{O}(n \log n)$ tak że punkty w obszarze zapytania można podać w czasie $\mathcal{O}(k + \log n)$, gdzie k jest liczbą podawanych punktów.

Kd-drzewa

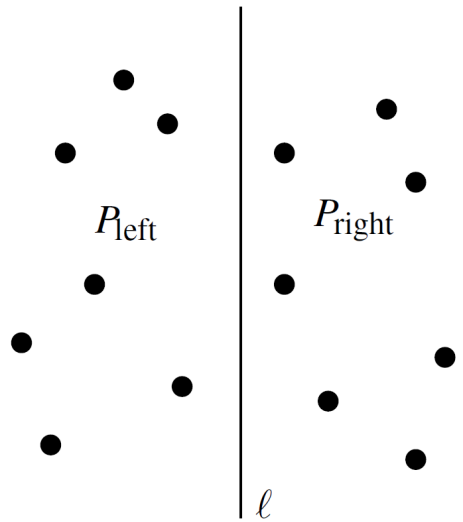
Niech P będzie zbiorem n punktów na płaszczyźnie oraz załóżmy, że żadne dwa punkty z P nie mają takiej samej współrzędnej x i żadne dwa punkty nie mają takiej samej współrzędnej y. Później pokażemy jak pozbyć się tego ograniczenia.

Tym razem obszar zapytania ma następującą postać $[x, x'] \times [y, y']$. Punkt $p = (p_x, p_y)$ leży wewnątrz tego prostokąta $\iff$ $p_x \in [x, x']$ i $p_y \in [y, y']$.



Uogólnimy teraz przypadek jednowymiarowy na dwuwymiarowy. Ponownie użyjemy zrównoważonego drzewa BST.

W korzeniu dzielimy P prostą pionową ℓ na dwa podzbiory o w przybliżeniu równych rozmiarach. Prosta dzieląca pamiętana jest w korzeniu. W lewym poddrzewie pamiętamy zbiór P_{left} punktów po lewej stronie lub na prostej dzielącej, zaś w prawym poddrzewie zbiór P_{right} punktów na prawo od prostej dzielącej.

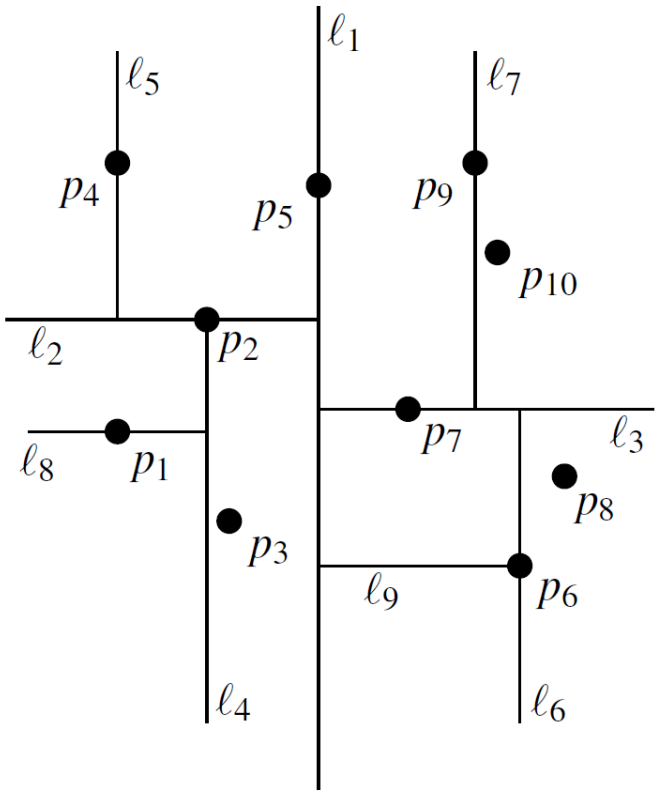


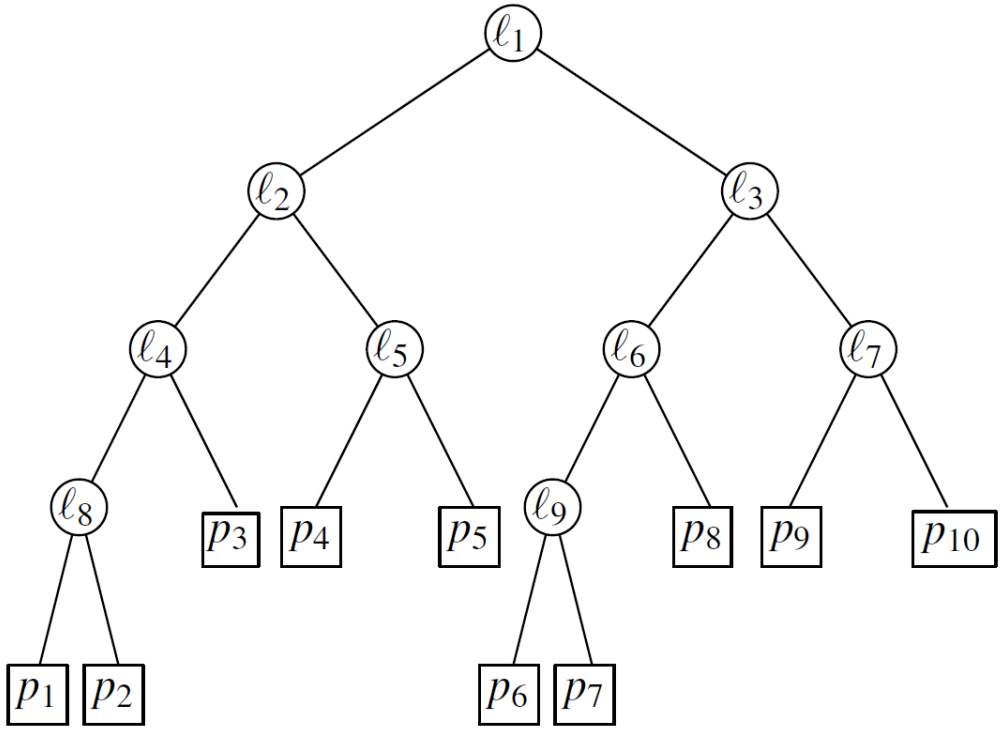
W lewym dziecku korzenia dzielimy P_{left} prostą poziomą na dwa podzbiory. Punkty poniżej lub na prostej są pamiętane w lewym poddrzewie tego dziecka, a punkty powyżej niej pamiętane są w prawym poddrzewie. Podobnie dzielimy P_{right} prostą poziomą na dwa podzbiory, które pamiętane są w lewym i prawym poddrzewie.

We wnukach korzenia znów dzielimy punkty prostą pionową.

Ogólnie dzielimy punkty prostą pionową w węzłach, których głębokość jest parzysta, a prostą poziomą w węzłach o głębokości nieparzystej.

Drzewo skonstruowane w ten sposób nazywamy dwuwymiarowym kd-drzewem.


ℓ_1
ℓ_5
ℓ_7
p_4
p_5
p_9
p_{10}
ℓ_2
p_2
ℓ_8
p_1
p_7
ℓ_3
p_8
p_3
ℓ_9
p_6
ℓ_4
ℓ_6


ℓ_1
ℓ_2
ℓ_3
ℓ_4
ℓ_5
ℓ_6
ℓ_7
ℓ_8
ℓ_9
p_1
p_2
p_3
p_4
p_5
p_6
p_7
p_8
p_9
p_{10}

Algorithm BUILDKDTREE($P, depth$)
Input. A set of points P and the current depth $depth$.
Output. The root of a kd-tree storing P.

1. **if** P contains only one point
2. **then return** a leaf storing this point
3. **else if** $depth$ is even
4. **then** Split P into two subsets with a vertical line ℓ through the median x-coordinate of the points in P. Let P_1 be the set of points to the left of ℓ or on ℓ, and let P_2 be the set of points to the right of ℓ.
5. **else** Split P into two subsets with a horizontal line ℓ through the median y-coordinate of the points in P. Let P_1 be the set of points below ℓ or on ℓ, and let P_2 be the set of points above ℓ.
6. $v_{\text{left}} \leftarrow$ BUILDKDTREE($P_1, depth + 1$)
7. $v_{\text{right}} \leftarrow$ BUILDKDTREE($P_2, depth + 1$)
8. Create a node v storing ℓ, make v_{left} the left child of v, and make v_{right} the right child of v.
9. **return** v

Algorytm uruchamiamy z drugim parametrem (głębokość rekurencji) ustawionym na 0.

Do wyznaczenia mediany możemy wykorzystać jeden z algorytmów, które działają w czasie liniowym. Są one skomplikowane, więc zamiast tego możemy posortować tablicę punktów po współrzędnej x i y. Zatem zbiór P jest przekazywany do procedury w postaci dwóch posortowanych list.

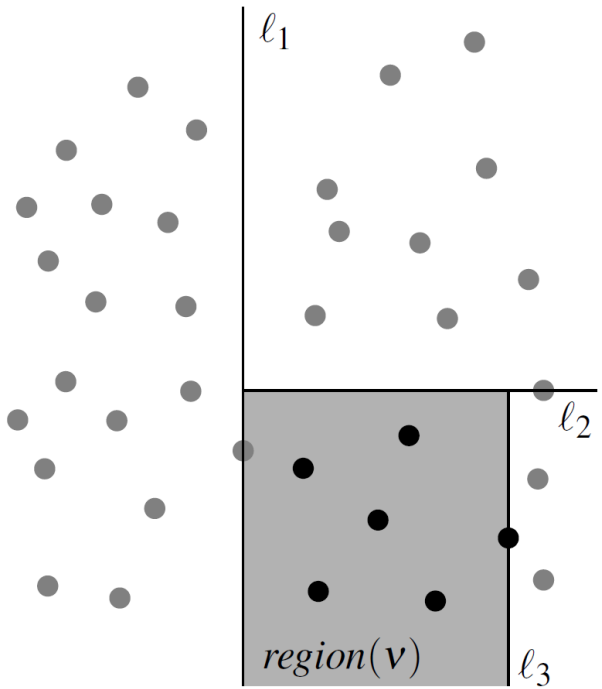
Algorytm uruchamiamy z drugim parametrem (głębokość rekurencji) ustawionym na 0.

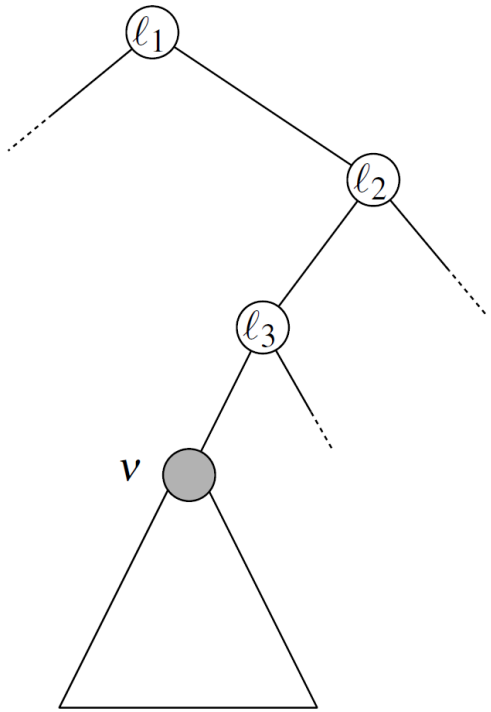
Do wyznaczenia mediany możemy wykorzystać jeden z algorytmów, które działają w czasie liniowym. Są one skomplikowane, więc zamiast tego możemy posortować tablicę punktów po współrzędnej x i y. Zatem zbiór P jest przekazywany do procedury w postaci dwóch posortowanych list.

Kd-drzewo dla zbioru n punktów korzysta z pamięci $\mathcal{O}(n)$ i można je zbudować w czasie $\mathcal{O}(n \log n)$.

Jak wyglądają zapytania w kd-drzewie?

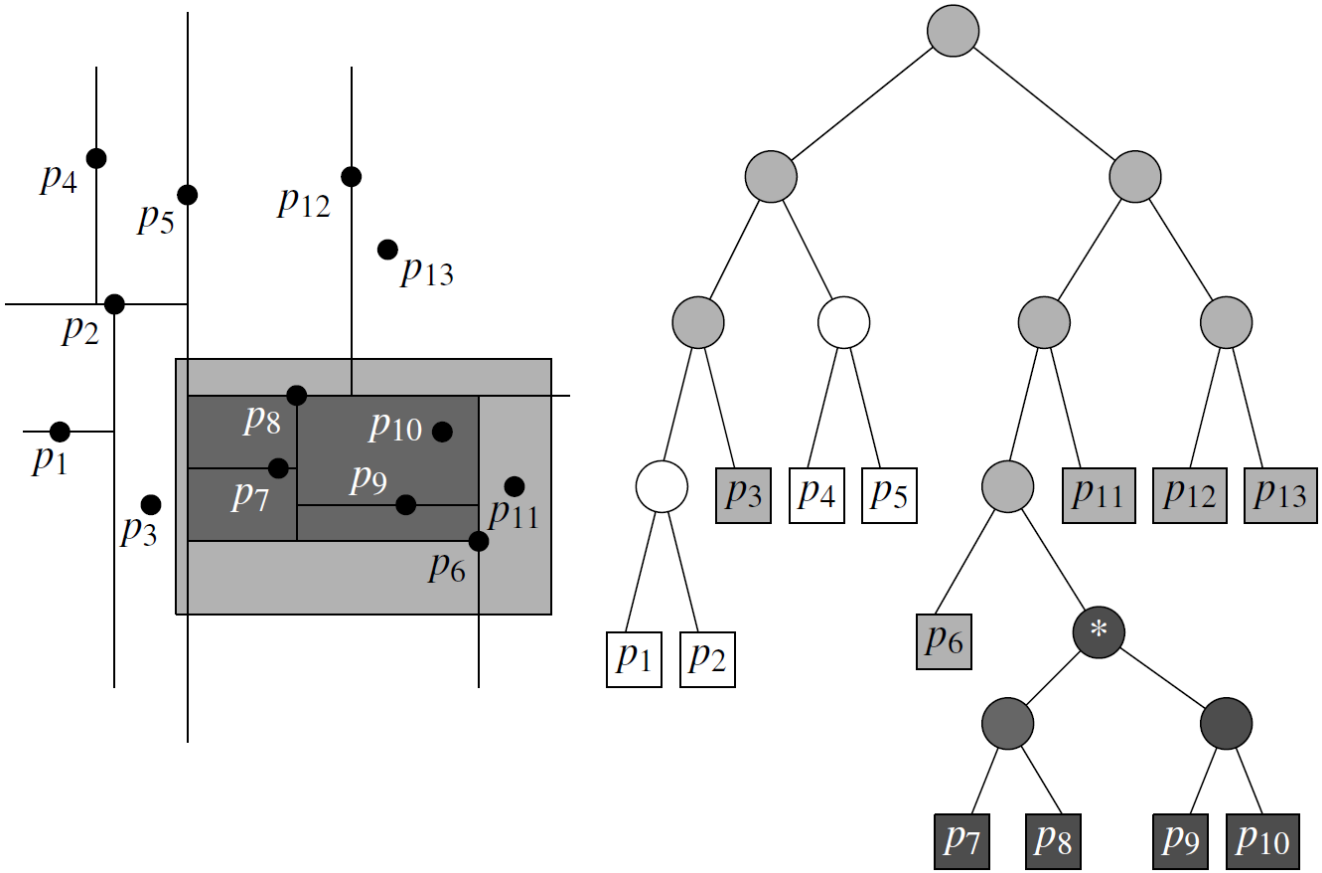
Obszar odpowiadający węzłowi v w kd-drzewie jest prostokątem, który może być nieograniczony z jednej lub więcej stron. Jest on ograniczony przez proste dzielące zapamiętane w przodkach v. Niech $region(v)$ oznacza obszar odpowiadający węzłowi v.





Punkt jest pamiętany w poddrzewie zakorzenionym w węźle $v \iff$ gdy leży w $region(v)$. Zatem przeszukujemy drzewo zakorzenione w $v \iff$ prostokąt zapytania przecina $region(v)$.

Nasz algorytm zapytań wygląda w skrócie następująco. Przechodzimy kd-drzewo, ale odwiedzamy tylko węzły, których obszary są przecinane przez prostokąt zapytania. Gdy obszar jest całkowicie zawarty w prostokącie zapytania, możemy podać wszystkie punkty pamiętane w poddrzewie. Gdy dotrzemy do liścia musimy sprawdzić czy punkt pamiętany w liściu jest zawarty w obszarze zapytania.



Szare węzły są odwiedzane, gdy pytamy o szary prostokąt.

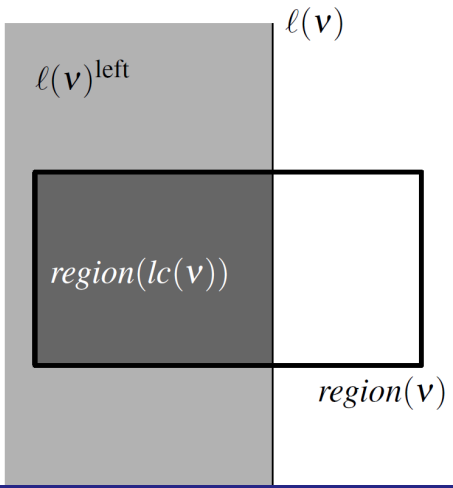
Algorithm SEARCHKDTREE(v, R)
Input. The root of (a subtree of) a kd-tree, and a range R.
Output. All points at leaves below v that lie in the range.

```
1.  if v is a leaf
2.      then Report the point stored at v if it lies in R.
3.      else if region(lc(v)) is fully contained in R
4.              then REPORTSUBTREE(lc(v))
5.              else if region(lc(v)) intersects R
6.                      then SEARCHKDTREE(lc(v), R)
7.           if region(rc(v)) is fully contained in R
8.              then REPORTSUBTREE(rc(v))
9.              else if region(rc(v)) intersects R
10.                     then SEARCHKDTREE(rc(v), R)
```

Główny test w algorytmie sprawdza czy obszar zapytania R przecina obszar odpowiadający pewnemu węzłowi v.

W fazie przetwarzania wstępnego możemy obliczyć $region(v)$ dla każdego węzła i zapamiętać go. Możemy także w trakcie wywołania rekurencyjnego utrzymywać aktualny obszar używając prostych pamiętanych w węzłach wewnętrznych.
Np. obszar $region(lc(v))$ węzła o parzystej głębokości możemy obliczyć następująco:

$$region(lc(v)) = region(v) \cap \ell(v)^{left}.$$



Nasz algorytm zapytań nigdzie nie zakłada, że obszar zapytania jest prostokątem. Zadziała on również dla każdego innego obszaru zapytania.

Zapytanie o prostokąt o bokach równoległych do osi można wykonać w kd-drzewie przechowującym n punktów w czasie $\mathcal{O}(\sqrt{n} + k)$, gdzie k jest liczbą podawanych punktów.

Nasz algorytm zapytań nigdzie nie zakłada, że obszar zapytania jest prostokątem. Zadziała on również dla każdego innego obszaru zapytania.

Zapytanie o prostokąt o bokach równoległych do osi można wykonać w kd-drzewie przechowującym n punktów w czasie $\mathcal{O}(\sqrt{n} + k)$, gdzie k jest liczbą podawanych punktów.

Kd-drzewa możemy również stosować dla punktów trój- lub wielowymiarowej przestrzeni.

W algorytmie konstrukcji w korzeniu dzielmy zbiór punktów hiperpłaszczyzną względem pierwszej współrzędnej. W dzieciach korzenia dzielimy względem drugiej współrzędnej, w węzłach o głębokości dwa względem trzeciej itd. aż do głębokości $d - 1$. Na głębokości d zaczynamy wszystko od nowa.

Tak skonstruowane drzewo używa $\mathcal{O}(n)$ pamięci, a czas konstrukcji wynosi $\mathcal{O}(n \log n)$.

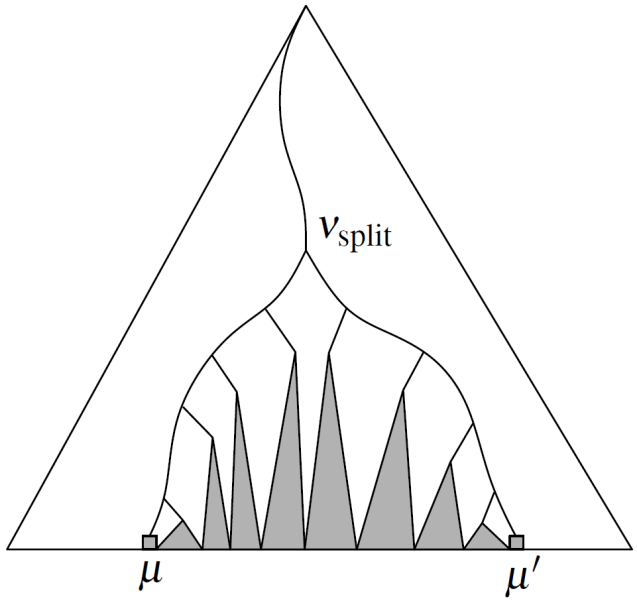
Węzły w d-wymiarowym kd-drzewie odpowiadają obszarom. Algorytm zapytań odwiedza te węzły, których obszary są właściwie przecinane przez obszar zapytania i obchodzi poddrzewa zakorzenione w węzłach, których obszary są całkowicie zawarte w obszarze zapytania.

Czas zapytania dla d-wymiarowego kd-drzewa dla n punktów wynosi $\mathcal{O}(n^{1-1/d} + k)$, gdzie k jest liczbą podawanych punktów.

Dwuwymiarowe zapytanie o obszar jest złożone z dwóch jednowymiarowych zapytań. Jednego o współrzędną x, a drugiego o współrzędną y. Pomysł ten był wykorzystany w kd-drzewach do dzielenia na przemian względem współrzędnych x i y. Teraz wykorzystamy to spostrzeżenie nieco inaczej.

Niech ponownie P będzie zbiorem n punktów na płaszczyźnie oraz $[x, x'] \times [y, y']$ będzie obszarem zapytania.

Najpierw koncentrujemy się na znalezieniu punktów, których współrzędna x leży w $[x, x']$. Jest to zapytanie o obszar jednowymiarowy, które omawialiśmy wcześniej i było realizowane za pomocą drzewa BST.



Podzbiór punktów pamiętany w liściach poddrzewa zakorzenionego w węźle v nazywamy podzbiorem kanonicznym (ang. canonical subset) v i oznaczamy $P(v)$.

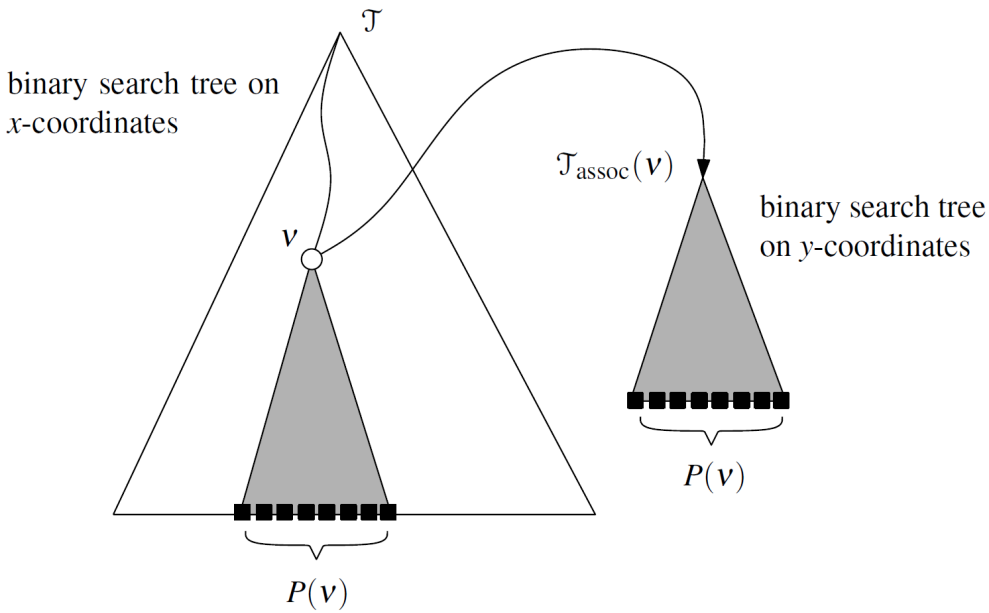
Podzbiór kanoniczny korzenia drzewa jest całym zbiorem P, a liścia jest po prostu punktem pamiętanym w tym liściu.

Zatem podzbiór punktów, których współrzędna x leży w obszarze zapytania można wyrazić jako sumę $\mathcal{O}(\log n)$ rozłącznych podzbiorów kanonicznych. Są to zbiory $P(v)$ węzłów v, które są korzeniami wybranych poddrzew.

Nie interesują nas wszystkie punkty w tych podzbiorach tylko te, których współrzędna y leży w przedziale $[y, y']$. Jest to kolejne zapytanie jednowymiarowe, które również możemy rozwiązać pod warunkiem, że dostępne jest drzewo BST względem współrzędnej y punktów w $P(v)$.

Zatem definiujemy następującą strukturę danych:

- Główne drzewo jest zrównoważonym drzewem BST $\mathcal{T}$ zbudowanym względem współrzędnej x punktów z P.
- Dla każdego węzła wewnętrznego lub liścia v w $\mathcal{T}$ podzbiór kanoniczny $P(v)$ jest pamiętany w zrównoważonym drzewie BST $\mathcal{T}_{assoc}(v)$ względem współrzędnej y punktów. Węzeł v pamięta wskaźnik do drzewa $\mathcal{T}_{assoc}(v)$, które nazywamy strukturą stowarzyszoną (ang. associated structure) z v.



Strukturę danych zdefiniowaną na poprzednim slajdzie nazywamy drzewem obszarów (ang. range tree).

Struktury danych, w których węzły mają wskaźniki do struktur stowarzyszonych nazywamy wielopoziomowymi strukturami danych (ang. multi-level data structures). Główne drzewo $\mathcal{T}$ nazywa się wtedy drzewem pierwszego poziomu, a struktury stowarzyszone są drzewami drugiego poziomu.

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 $\mathcal{T}_{\text{assoc}}$ on the set P_y of y-coordinates of the points in P. Store at the leaves of $\mathcal{T}_{\text{assoc}}$ 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 $\mathcal{T}_{\text{assoc}}$ 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. $v_{\text{left}} \leftarrow$ BUILD2DRANGETREE(P_{left})
6. $v_{\text{right}} \leftarrow$ 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 $\mathcal{T}_{\text{assoc}}$ the associated structure of v.
8. **return** v

Drzewo obszarów dla zbioru n punktów na płaszczyźnie wymaga $\mathcal{O}(n \log n)$ pamięci i czasu $\mathcal{O}(n \log n)$.

Algorytm zapytań najpierw wybiera $\mathcal{O}(\log n)$ kanonicznych podzbiorów zawierających łącznie punkty, których współrzędna x leży w $[x, x']$. Robimy to za pomocą głównego drzewa naszego drzewa obszarów. Następnie z tych podzbiorów podajemy punkty, których współrzędna y leży w $[y, y']$. Do tego celu stosujemy struktury stowarzyszone, które pamiętają wybrany kanoniczny podzbiór.

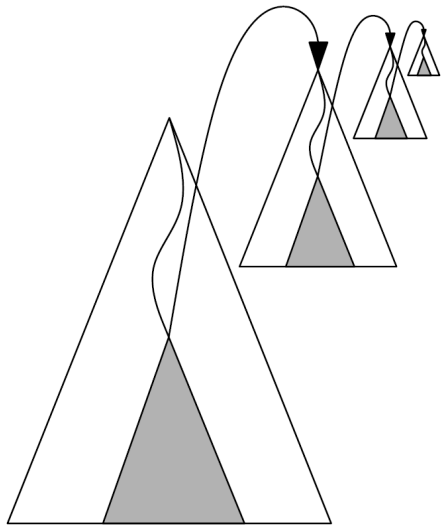
Algorithm 2DRANGEQUERY($\mathcal{T}, [x : x'] \times [y : y']$)
Input. A 2-dimensional range tree $\mathcal{T}$ and a range $[x : x'] \times [y : y']$.
Output. All points in $\mathcal{T}$ that lie in the range.

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.

Zapytanie o prostokąt o bokach równoległych do osi wymaga w drzewie obszarów przechowującym n punktów czasu $\mathcal{O}(\log^2 n + k)$, gdzie k jest liczbą podawanych punktów.

Zapytanie o prostokąt o bokach równoległych do osi wymaga w drzewie obszarów przechowującym n punktów czasu $\mathcal{O}(\log^2 n + k)$, gdzie k jest liczbą podawanych punktów.

Podobnie jak kd-drzewa również drzewa obszarów możemy uogólnić z przypadku przestrzeni dwuwymiarowej na przestrzeń d-wymiarową.



Niech P będzie zbiorem n punktów w d-wymiarowej przestrzeni, gdzie $d \geqslant 2$. Drzewo obszarów dla P używa $\mathcal{O}(n \log^{d-1} n)$ pamięci i można je zbudować w czasie $\mathcal{O}(n \log^{d-1} n)$. Punkty z P leżące w prostokątnym obszarze zapytania można podać w czasie $\mathcal{O}(\log^d n + k)$, gdzie k jest liczbą podawanych punktów.

Ogólny zbiór punktów

Do tej pory zakładaliśmy, że żadne dwa punkty nie mogą mieć równych współrzędnych x ani y. Teraz zobaczymy jak zlikwidować to założenie.

Nigdy w rozważaniach nie zakładaliśmy, że współrzędne są liczbami rzeczywistymi. Potrzebowaliśmy tylko, aby pochodziły z liniowo uporządkowanego zbioru, tak abyśmy mogli porównywać dowolne dwie współrzędne i obliczać medianę.

W związku z tym zastosujemy pewną „sztuczkę”.

Zastępujemy współrzędne, które są liczbami rzeczywistymi przez elementy tak zwanej przestrzeni liczb złożonych (ang. composite-number space). Elementy tej przestrzeni są parami liczb rzeczywistych.

Liczbę złożoną z dwóch liczb rzeczywistych a i b oznaczamy $(a|b)$.

Definiujemy porządek liniowy w przestrzeni liczb złożonych stosując porządek leksykograficzny, tzn.

$$(a|b) < (a'|b') \iff a < a' \vee (a = a' \wedge b < b')$$

dla dowolnych liczb złożonych $(a|b)$, $(a'|b')$.

Załóżmy, że mamy zbiór P dowolnych n punktów na płaszczyźnie. Zastępujemy każdy punkt $p = (p_x, p_y)$ nowym punktem

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

W ten sposób otrzymujemy nowy zbiór punktów $\hat{P}$, w którym pierwsze współrzędne dowolnych dwóch punktów są różne, podobnie drugie współrzędne.

Mając zbiór $\hat{P}$ możemy zbudować kd-drzewo czy drzewo obszarów.

Przypuśćmy teraz, że zadajemy zapytanie o obszar $R = [x, x'] \times [y, y']$. Ponieważ nasze drzewo jest zbudowane dla $\hat{P}$, więc obszar też musimy przekształcić w przestrzeń liczb złożonych. Nowy obszar $\hat{R}$ będzie postaci:

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

Lemat. Niech p będzie punktem, a R prostokątnym obszarem. Wtedy

$$p \in R \iff \hat{p} \in \hat{R}.$$

Zauważmy na koniec, że nie musimy pamiętać przekształconych punktów. Możemy pamiętać początkowe punkty pod warunkiem, że porównujemy współrzędne w przestrzeni liczb złożonych.