Geometria obliczeniowa
Wykład 4

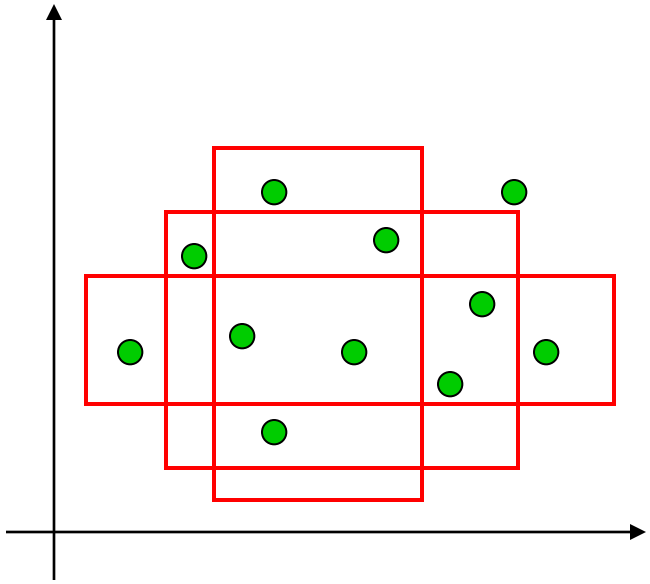
Przeszukiwanie obszarów ortogonalnych

1. Liczby złożone
2. Kd drzewa
3. Drzewa obszarów
4. Kaskadowanie
5. Warstwowe drzewo obszarów
6. Drzewa przeszukiwań priorytetowych
7. Drzewa przedziałów

Przeszukiwanie obszarów ortogonalnych

Problem:

Dany jest zbiór n punktów na płaszczyźnie. Pragniemy znaleźć podzbiór tych punktów, które zawierają się w danym prostokącie $[x_1,x_2]\times[y_1,y_2]$. Zadanie to jest stawiane wielokrotnie dla różnych prostokątów, ale tego samego zbioru punktów.



Szybkie wyszukiwanie interesującego nas zbioru może być utrudnione, gdy wiele punktów ma takie same niektóre współrzędne (założenie o rozróżnialności współrzędnych jest nierealistyczne ze względów praktycznych). W takiej sytuacji możemy zastosować liczby złożone.

Liczby złożone.

Zastępujemy współrzędne, które są liczbami rzeczywistymi, przez elementy tak zwanej ***przestrzeni liczb złożonych***. Elementy tej przestrzeni są parami liczb rzeczywistych. Liczbę złożoną dwóch liczb rzeczywistych a i b oznaczamy przez (a|b). Definiujemy porządek liniowy w przestrzeni liczb złożonych stosując porządek leksykograficzny. Zatem, dla dwóch liczb złożonych $(a_1|b_1)$ i $(a_2|b_2)$, mamy

$$(a_1|b_1)<(a_2|b_2) \Leftrightarrow a_1 < a_2 \text{ lub } (a_1 = a_2 \text{ i } b_1 < b_2).$$

Załóżmy teraz, że dany jest zbiór P zawierający n punktów na płaszczyźnie. Punkty są różne, ale wiele punktów może mieć takie same współrzędne x lub y. Zastępujemy każdy punkt $p:=(p_x,p_y)$ przez nowy punkt $p':=((p_x|p_y),(p_y|p_x))$, który ma liczby złożone jako wartości współrzędnych. W ten sposób otrzymujemy nowy zbiór P' zawierający n punktów. Pierwsze współrzędne dowolnych dwóch punktów z P' są różne. To samo pozostaje prawdą dla drugiej współrzędnej.

Przypuśćmy teraz, że chcemy podać punkty, które leżą w obszarze $R:=[x_1,x_2]\times[y_1,y_2]$. To znaczy, że musimy również określić obszar zapytania w przestrzeni liczb złożonych. Przekształcony obszar jest definiowany w następujący sposób:

$$R' := [(x_1|-\infty),(x_2|+\infty)]\times[(y_1|-\infty),(y_2|+\infty)].$$

Pozostaje do udowodnienia, że nasze podejście jest poprawne, to znaczy, że punkty z P', które wyliczamy zadając pytanie odnośnie R', dokładnie odpowiadają punktom z P, które leżą w R.

Lemat.

Niech p będzie punktem a R prostokątnym obszarem. Wtedy $p \in R \Leftrightarrow p' \in R'$.

Dowód.

Niech $R:=[x_1,x_2]\times[y_1,y_2]$ i $p:=(p_x,p_y)$. Z definicji, p leży w R wtedy i tylko wtedy, gdy $x_1 \leq p_x \leq x_2$ i $y_1 \leq p_y \leq y_2$. Łatwo można zobaczyć, że zachodzi to wtedy i tylko wtedy, gdy $(x_1|-\infty) \leq (p_x|p_y) \leq (x_2|+\infty)$ i $(y_1|-\infty) \leq (p_y|p_x) \leq (y_2|+\infty)$, czyli wtedy i tylko wtedy, gdy p' leży w R'.

Zauważmy, że nie potrzeba faktycznie pamiętać przekształconych punktów. Wystarczy pamiętać tylko punkty ze zbioru P, pod warunkiem, że będziemy porównywać ich współrzędne w przestrzeni liczb złożonych.

Podejście korzystające z liczb złożonych można również zastosować w wyższych wymiarach.

Reasumując, tam, gdzie będzie nam to potrzebne w celu stworzenia odpowiedniej struktury danych, możemy zakładać, że wszystkie współrzędne punktów ze zbioru P są różne.

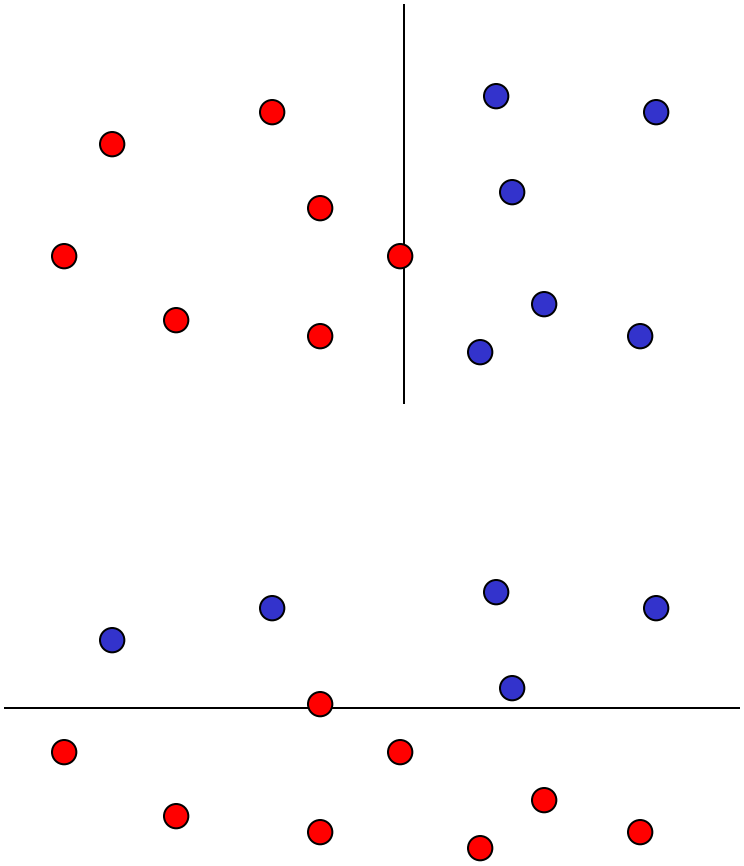
Kd drzewo

Kd drzewo to drzewo binarne, którego liśćmi są punkty z P a węzłami wewnętrznymi - proste równoległe do osi układu współrzędnych. Synami węzła odpowiadającego prostej pionowej są węzły odpowiadające prostym poziomym i vice versa.

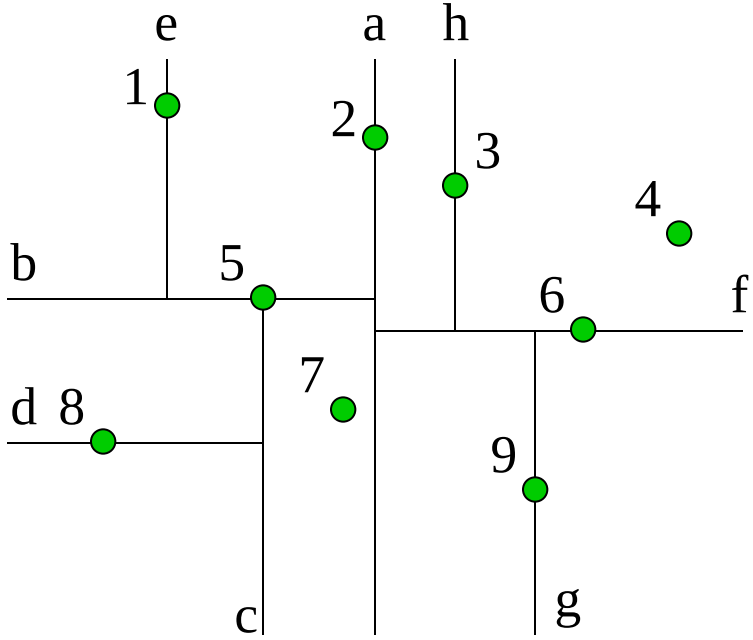
Jako korzeń wybieramy prostą pionową przechodzącą przez punkt z danego zbioru będący jego medianą względem współrzędnych x-owych. Dzielimy dany zbiór na dwa podzbiory, do których należą punkty odpowiednio nie większe od mediany i większe od niej (operujemy na liczbach złożonych).

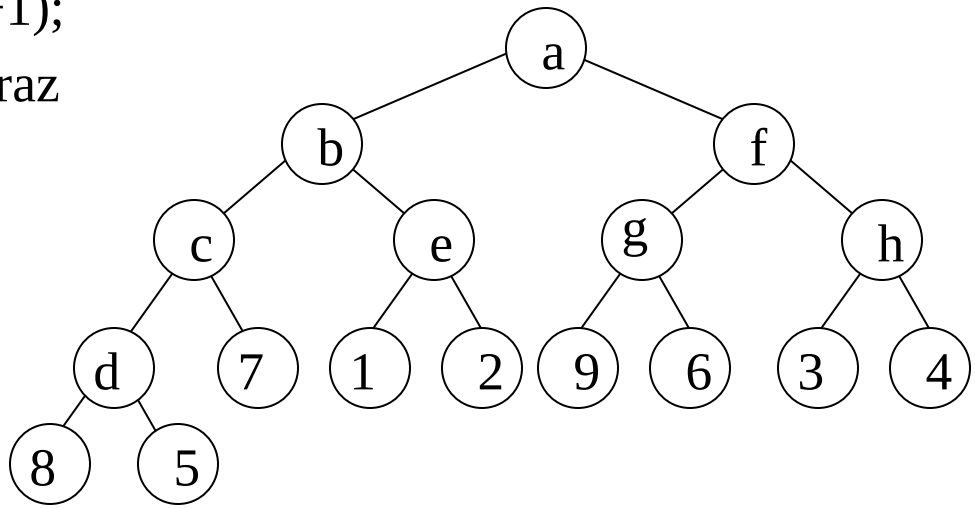
Następnie w ten sam sposób dzielimy każdy podzbiór względem y-ów itd.

Przyjmijmy, że pionowa (pozioma) prosta będzie dzielić zbiór punktów P na zbiór P_2 punktów leżących na prawo (powyżej) od tej prostej i zbiór $P_1 := P - P_2$.



procedure BUILDTREE(P,depth)
if P zawiera tylko jeden punkt
    then return liść pamiętający ten punkt
    else if depth jest parzyste
        then podziel P pionową prostą
            l na zbiory P_1 i P_2
        else podziel P poziomą prostą
            l na zbiory P_1 i P_2;
    $v_l \leftarrow$ BUILDTREE(P_1,depth+1);
    $v_p \leftarrow$ BUILDTREE(P_2,depth+1);
    stwórz węzeł v – ojca v_l i v_p oraz
    zapamiętaj w nim l;
return v





Niech ls(v) (rs(v)) oznacza lewego (prawego) syna wierzchołka v, a obszar(l) jest obszarem, który dzieli prosta l.

procedure SEARCHKD(v,R)

if v jest liściem

then if v ∈ R ***then*** zwróć v

else if obszar(ls(v)) ⊆ R

then zwróć wszystkie liście pod-drzewa o korzeniu w ls(v)

else if obszar(ls(v)) przecina R

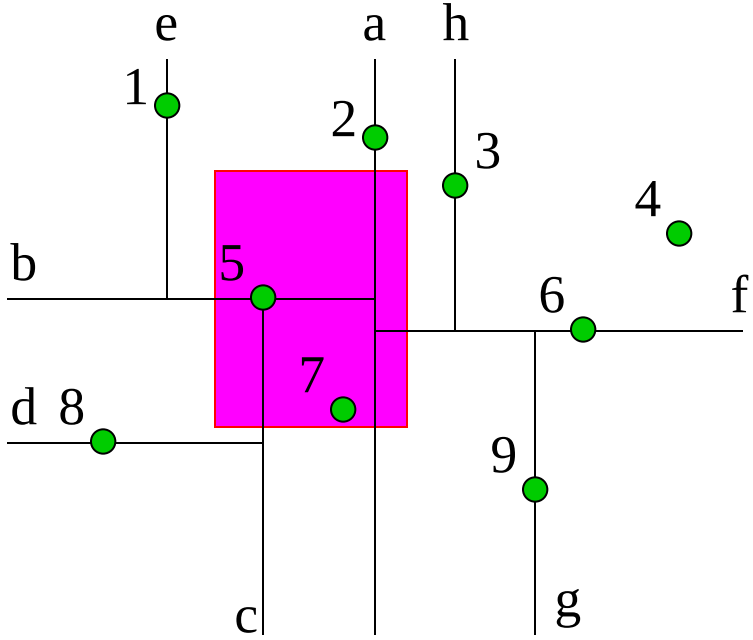
then SEARCHKD(ls(v),R)

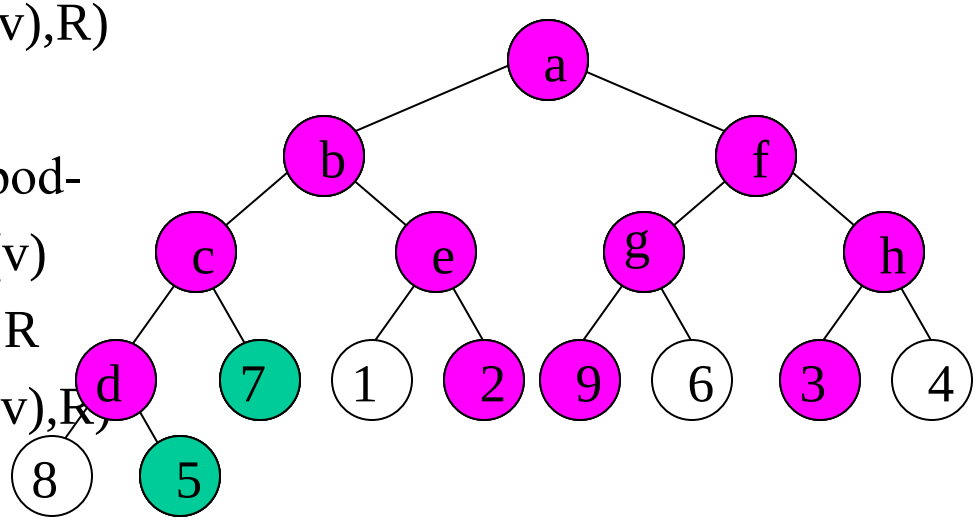
if obszar(rs(v)) ⊆ R

then zwróć wszystkie liście pod-drzewa o korzeniu w rs(v)

else if obszar(rs(v)) przecina R

then SEARCHKD(rs(v),R)





Lemat.

Kd drzewo ma rozmiar liniowy względem rozmiaru zbioru i można je zbudować w czasie O(n log n).

Dowód.

Czas potrzebny na stworzenie kd drzewa jest opisany przez następujące równania rekurencyjne: $T(n) = O(1)$ dla $n = 1$ oraz $T(n) = O(n) + 2T(\lceil n/2 \rceil)$ dla $n > 1$, czyli $T(n) = O(n \log n)$.

Lemat.

Wykorzystując kd drzewa, znalezienie wszystkich punktów z n-elementowego zbioru P należących do danego prostokąta R wymaga czasu $O(\sqrt{n} + k)$, gdzie k jest liczbą znalezionych punktów.

Dowód.

Policzmy, ile porównań potrzebujemy w celu znalezienia obszarów przeciętych przez brzeg prostokąta R. Obszary wewnątrz prostokąta zawierają elementy zbioru, więc jest ich O(k).

Rozważmy jedną z prostych zawierających bok prostokąta R i stwórzmy równania rekurencyjne na liczbę porównań po wykonaniu dwóch kroków wyszukiwania. Mają one postać: $Q(n) = O(1)$ dla $n = 1$ oraz $Q(n) = 3+2Q(\lceil n/4 \rceil)$ dla $n > 1$, a ich rozwiązaniem jest $Q(n)=O(n^{1/2})$. .

Konstrukcję kd drzewa możemy uogólnić również na wyższe wymiary.

Dla n-elementowego zbioru punktów P trzymamy wtedy drzewo rozmiaru O(n), które można skonstruować w czasie O(n log n).

Czas odpowiedzi na zapytanie o punkty należące do prostopadłościanu R w d wymiarowej przestrzeni wynosi $O(k+n^{(d-1)/d})$, gdzie k jest liczbą znalezionych punktów.

Drzewa obszarów (range tree).

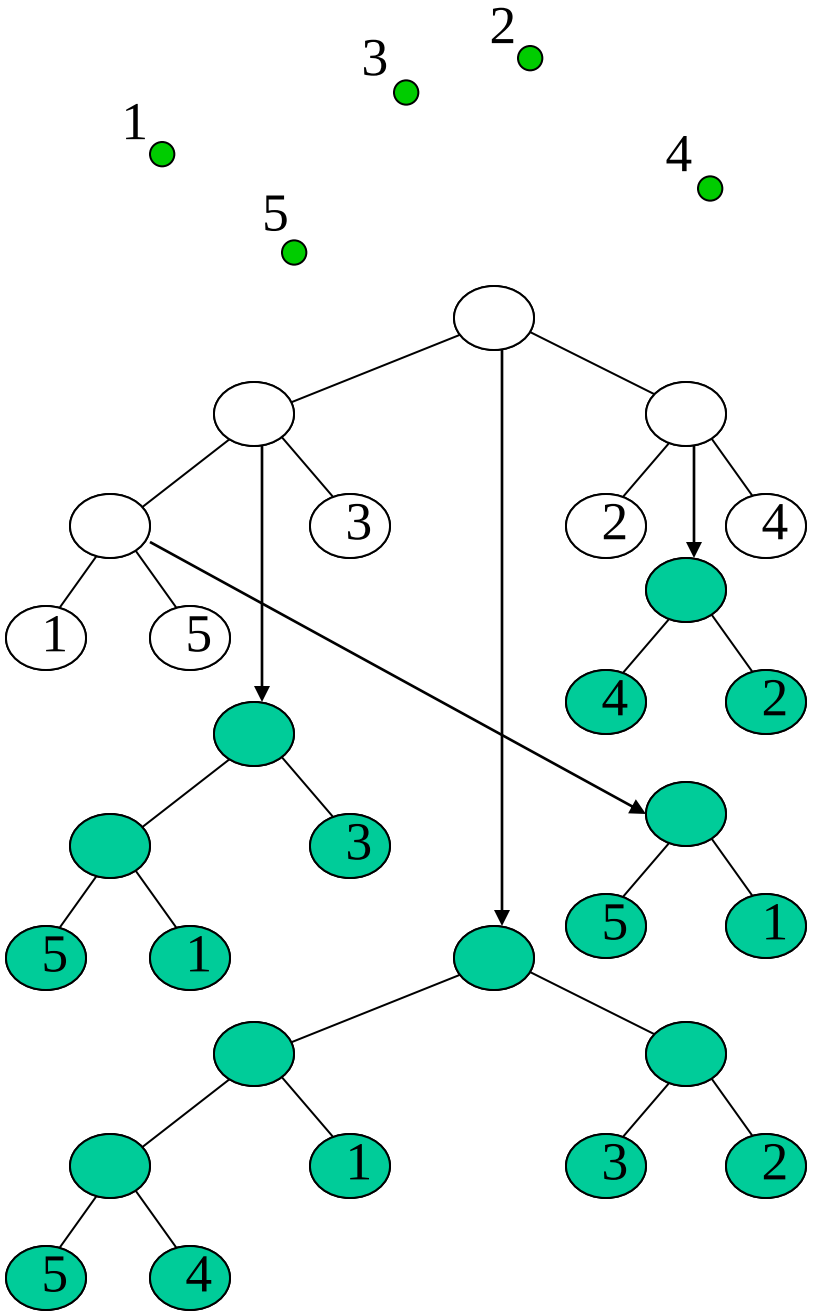
Drzewa obszarów tworzymy następująco:

Punkty z danego zbioru P uporządkowane względem współrzędnych x-owych umieszczamy w liściach wzbogaconego, zrównoważonego drzewa poszukiwań binarnych T.

Dodatkowo każdy wierzchołek v drzewa T wskazuje na wzbogacone, zrównoważone drzewo poszukiwań binarnych, w którym wszystkie punkty poddrzewa drzewa T o korzeniu v uporządkowane są względem współrzędnej y-owej.

W węzłach drzewa T (oraz drzew stowarzyszonych) pamiętamy wartość współrzędnej x (y) wyznaczającej podział na prawe i lewe poddrzewa.

{# na przykładzie nie ma struktur stowarzyszonych dla liści ani wartości współrzędnych w węzłach #}



Niech x_m (y_m) oznacza odpowiednią współrzędną mediany zbioru. Niech H_P oznacza strukturę stowarzyszoną dla zbioru P.

procedure BUILDRT(P)

if P zawiera tylko jeden punkt v

then stwórz liść w T i strukturę $H_{\{v\}}$

else podziel P na dwa zbiory P_l i P_r

względem x_m ;

$v_l \leftarrow$ BUILDRT(P_l);

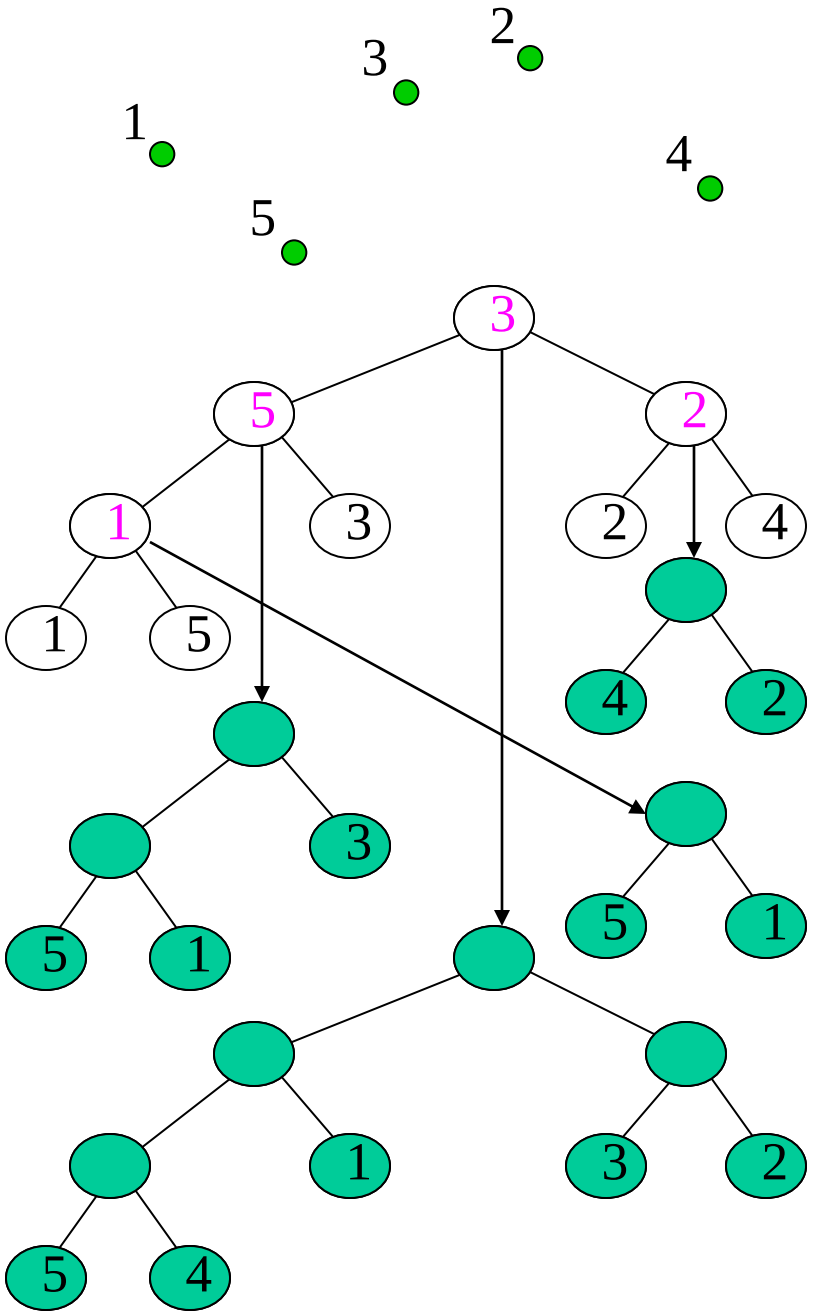
$v_p \leftarrow$ BUILDRT(P_r);

stwórz H_P, zapamiętaj w niej punkty i ich współrzędne y-owe (scalanie);

stwórz węzeł v będący ojcem v_l i v_p, zapamiętaj w nim x_m i połącz z H_P ;

return v ;

{# struktur $H_{\{v\}}$ nie ma na rysunku z uwagi na brak miejsca, w węzłach powinny być współrzędne zamiast etykiet punktów #}



Niech R := $[x_1,x_2]\times[y_1,y_2]$ oraz v_x określa węzeł rozejścia się ścieżek poszukiwania x_1 i x_2.

procedure SEARCHRT(T,R)

znajdź v_x ;

if v_x jest liściem

then sprawdź czy odpowiadający mu punkt należy do R

else $v \leftarrow ls(v_x)$;

while v nie jest liściem ***do***

if $x_1 \le x_v$

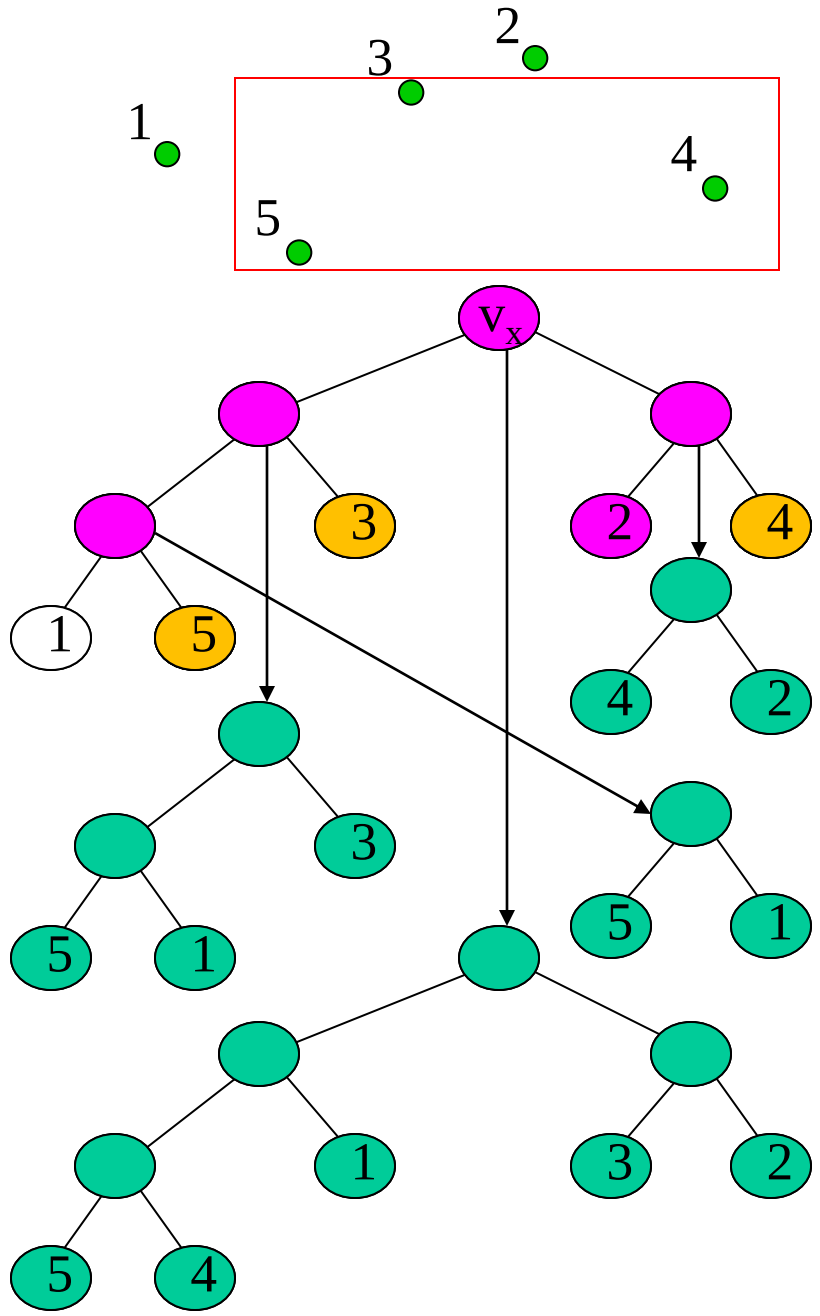
then znajdź w strukturze stowarzyszonej dla rs(v) punkty z R;

$v \leftarrow ls(v)$

else $v \leftarrow rs(v)$;

sprawdź czy punkt z liścia jest w R ;

wykonaj symetryczne operacje do powyższych zaczynając od $v \leftarrow rs(v_x)$;



Lemat.

Drzewo obszarów dla n-elementowego zbioru punktów na płaszczyźnie wymaga O(n log n) pamięci i może zostać skonstruowane również w czasie O(n log n).

Lemat.

Znalezienie z pomocą drzewa obszarów wszystkich punktów z n-elementowego zbioru P należących do danego prostokąta R wymaga czasu $O(k + \log^2 n)$, gdzie k jest liczbą znalezionych punktów.

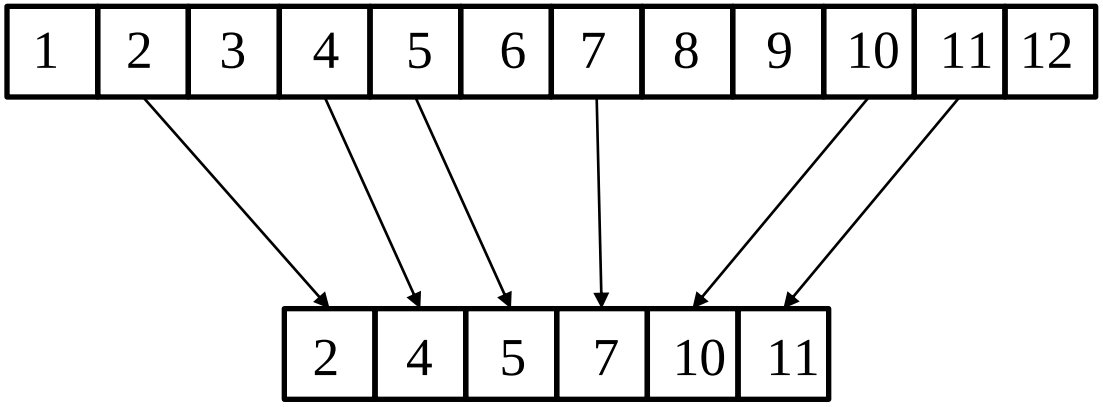
Konstrukcję drzewa obszarów można również rozszerzyć na wyższe wymiary.

Dla n-elementowego zbioru P w d wymiarowej przestrzeni drzewo obszarów ma rozmiar $O(n \log^{d-1} n)$ i można je zbudować w czasie $O(n \log^{d-1} n)$. Odpowiedź na zapytanie o punkty należące do prostopadłościanu R otrzymujemy w czasie $O(k + \log^d n)$, gdzie k jest liczbą znalezionych punktów.

Kaskadowanie (cascading).

Jeśli mamy dwa uporządkowane zbiory liczb, z których jeden jest podzbiorem drugiego, to chcąc wyszukać w każdym z nich ten sam element nie musimy tego robić dwukrotnie.

Zamiast tego, tworząc zbiory możemy dodać wskaźniki łączące odpowiednie pary elementów.



W taki sam sposób możemy określić zależności między drzewami stowarzyszonymi w strukturze drzewa obszarów.

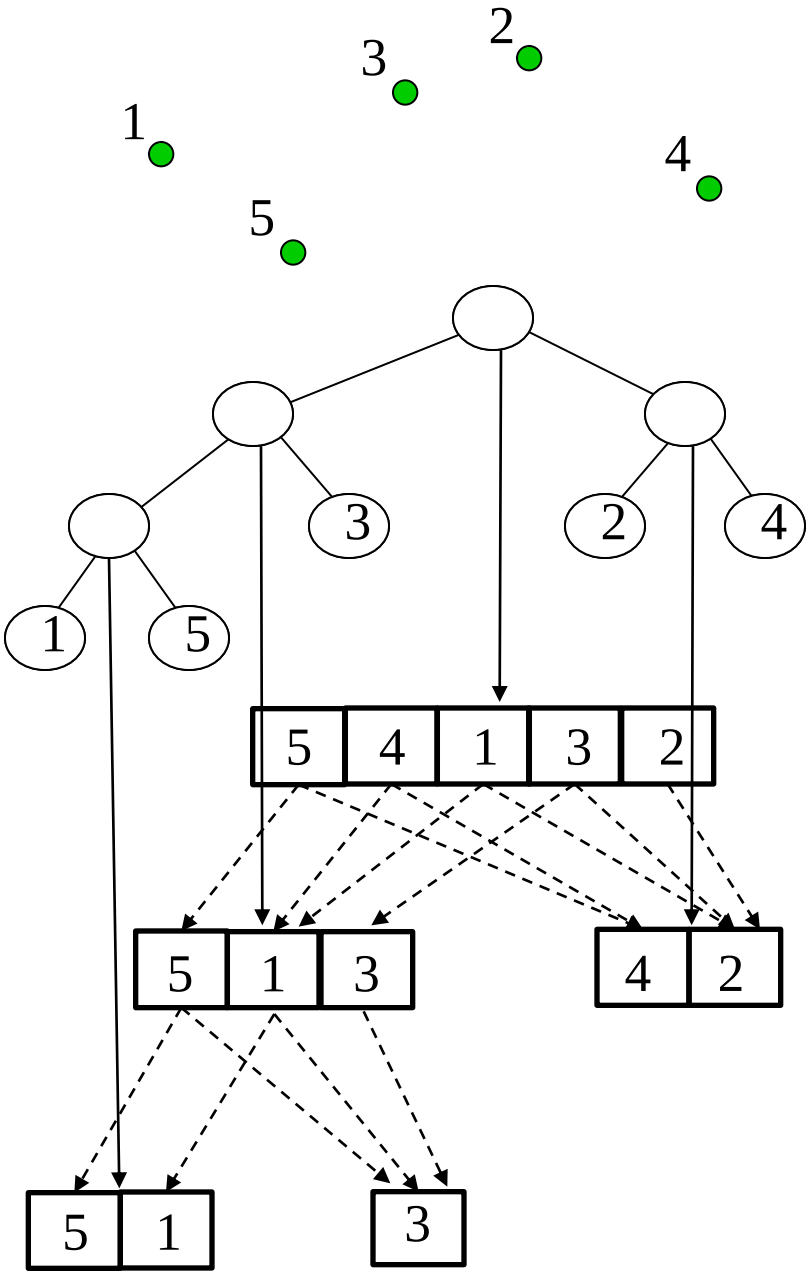
Warstwowe drzewo obszarów (layered range tree).

Na ostatnim poziomie wyszukiwania zamiast drzew stowarzyszonych stosujemy tablice z dowiązaniami pomiędzy nimi.

Wskaźniki z komórki zawierającej wartość v prowadzą do najmniejszej nie mniejszej niż v wartości w każdym z poddrzew.

Przechodzimy z tablicy do tablicy zgodnie ze ścieżką wyznaczoną na poprzednim poziomie wyszukiwania określając jednocześnie elementy należące do zakresu wyznaczonego przez ostatnią badaną współrzędną.

Wyszukiwanie binarne wykonujemy tylko w tablicy odpowiadającej momentowi rozejścia się ścieżek wyszukiwań zakresu dla poprzedniego poziomu.



Konstrukcję warstwowego drzewa obszarów można również rozszerzyć na wyższe wymiary.

Lemat.

Dla n-elementowego zbioru P w $d \geq 2$ wymiarowej przestrzeni warstwowe drzewo obszarów ma rozmiar $O(n \log^{d-1} n)$ i można je zbudować w czasie $O(n \log^{d-1} n)$. Odpowiedź na zapytanie o punkty należące do prostopadłościanu R otrzymujemy w czasie $O(k + \log^{d-1} n)$, gdzie k jest liczbą znalezionych punktów.

Drzewa przeszukiwań priorytetowych

(priority search tree).

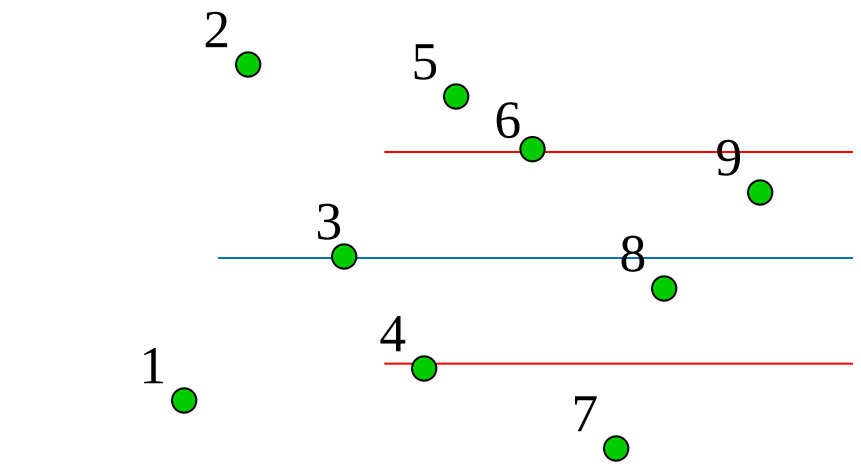
Załóżmy, że prostokąt R jest lewostronnie nieograniczony ($R := [-\infty,x]\times[y_1,y_2]$).

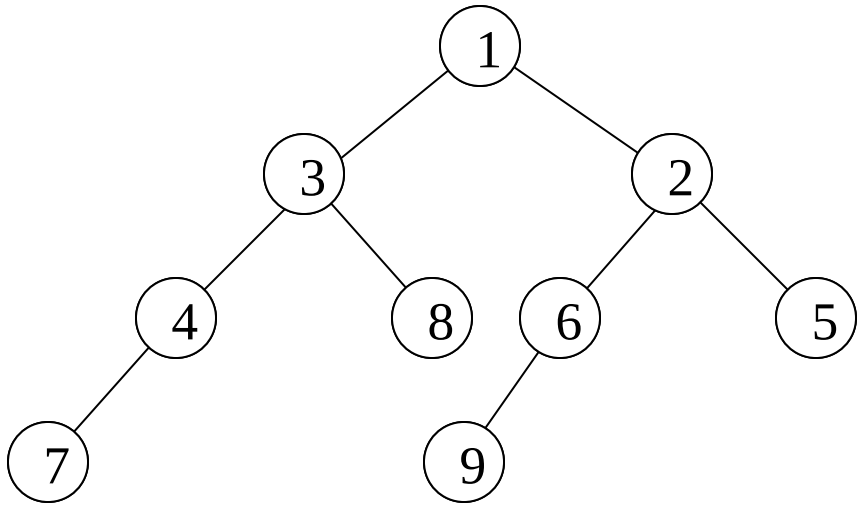
Niech p_1 będzie punktem o minimalnej x-owej współrzędnej, a y_m - medianą y-owych współrzędnych pozostałych punktów.

Niech $P_d = \{p \in P\text{-}\{p_1\}: y_p \leq y_m\}$

oraz $P_g = \{p \in P\text{-}\{p_1\}: y_p > y_m\}$

Tworzymy drzewo binarne T, którego korzeniem jest p_1 a jego synami są korzenie drzew stworzonych odpowiednio dla zbiorów P_d i P_g. W węzłach przechowujemy informacje o wartościach median.





Niech v_y określa punkt rozejścia się ścieżek poszukiwania y_1 i y_2 , a p(v) oznacza punkt odpowiadający wierzchołkowi v.

procedure SEARCHPST(T, R)

znajdź v_y ;

for każdy węzeł v na ścieżkach poszukiwań y_1 i y_2 ***do***

  if p(v) ∈ R ***then*** zwróć p(v) ;

for każdy węzeł v na ścieżce poszukiwań y_1 w lewym poddrzewie v_y ***do***

  if w v ścieżka poszukiwań idzie w lewo

    then wyszukaj punkty należące do R

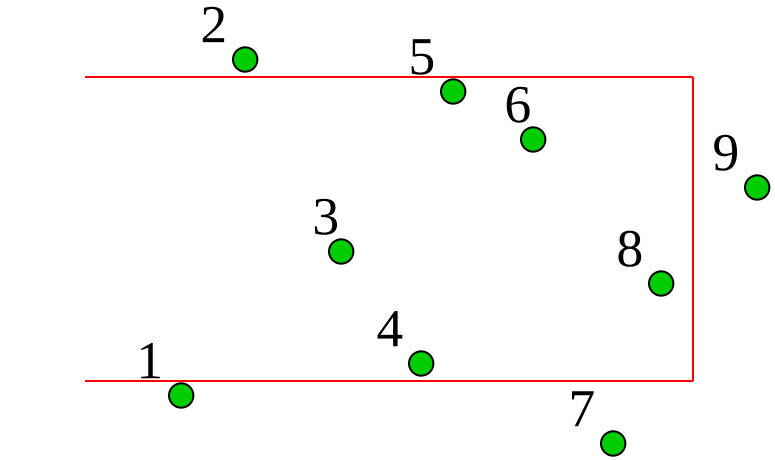
      w poddrzewie o korzeniu w rs(v);

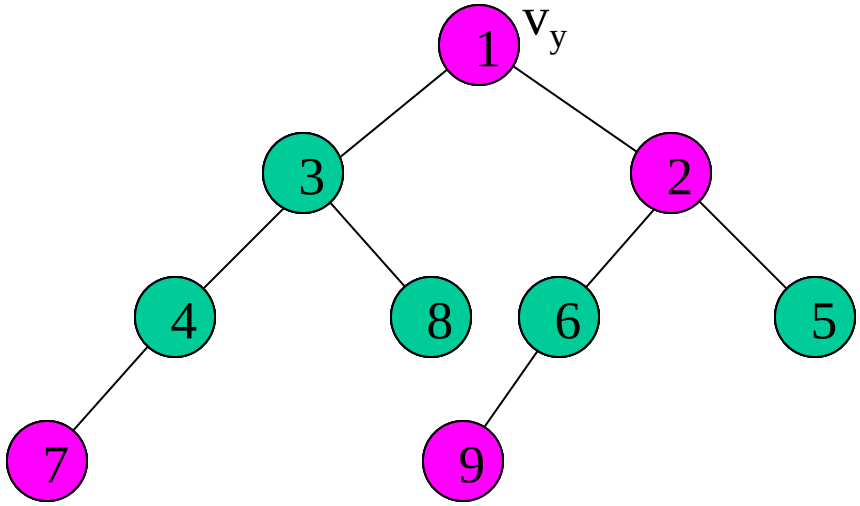
for każdy węzeł v na ścieżce poszukiwań y_2 w prawym poddrzewie v_y ***do***

  if w v ścieżka poszukiwań idzie w prawo

    then wyszukaj punkty należące do R

      w poddrzewie o korzeniu w ls(v);





Lemat.

Drzewo przeszukiwań priorytetowych dla n-elementowego zbioru P punktów na płaszczyźnie używa O(n) pamięci i można je zbudować w czasie O(n log n).

Lemat.

Wyszukiwanie punktów należących do R w poddrzewie nieodwiedzonego syna węzła v na ścieżce poszukiwań y_1 lub y_2 wymaga czasu $O(1+k_v)$, gdzie k_v oznacza liczbę punktów z R znalezionych w poddrzewie.

Dowód.

Jeśli wierzchołek w poddrzewie odpowiada punktowi z R, to wszystkie punkty leżące na ścieżce z korzenia (syna v) do w też są w R. Zatem liczba sprawdzeń jest proporcjonalna do liczby znalezionych punktów + sprawdzenia dla v (gdy nie znaleziono żadnego punktu).

Lemat.

Stosując drzewo przeszukiwań priorytetowych możemy określić wszystkie punkty z P należące do R w czasie O(k + log n), gdzie k jest liczbą znalezionych punktów.

Drzewo przedziałów (interval tree).

Rozważmy teraz przypadek, gdy zamiast punktów mamy do czynienia z n-elementowym zbiorem I pionowych lub poziomych odcinków. Chcemy znaleźć te z nich, które przecinają prostokąt $R := [x_1,x_2]\times[y_1,y_2]$.

Te odcinki, które mają przynajmniej jeden koniec wewnątrz prostokąta możemy znaleźć wykorzystując wcześniej omówione struktury.

Dlatego teraz będziemy tylko badać odcinki przecinające przeciwległe boki prostokąta. Załóżmy, że I jest zbiorem odcinków poziomych.

Niech x_m będzie medianą x-owych współrzędnych końców odcinków z I.

Zdefiniujmy: $I_l := \{[x_p,x_k] \in I: x_k < x_m \}$, $I_m := \{[x_p,x_k] \in I: x_p \le x_m \le x_k \}$, $I_r := \{[x_p,x_k] \in I: x_m < x_p \}$.

Stworzymy drzewo binarne, którego korzeniem będzie x_m a jego synami będą mediany (o ile istnieją) wierzchołków ze zbiorów I_l i I_r. Z korzeniem wiążemy dwie uporządkowane (ku medianie) listy odpowiednio lewych i prawych końców odcinków należących do I_m. To samo robimy ze zbiorami I_l i I_r.

procedure BUILDIT(I)

if I = ∅

then zwróć pusty liść

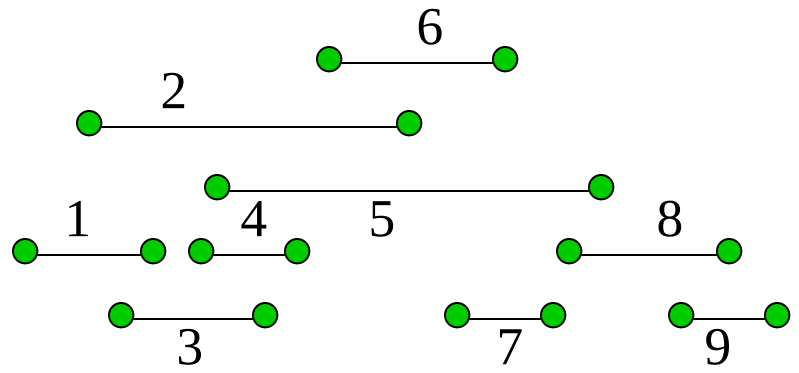
else stwórz węzeł v i zapamiętaj w nim x_m ;

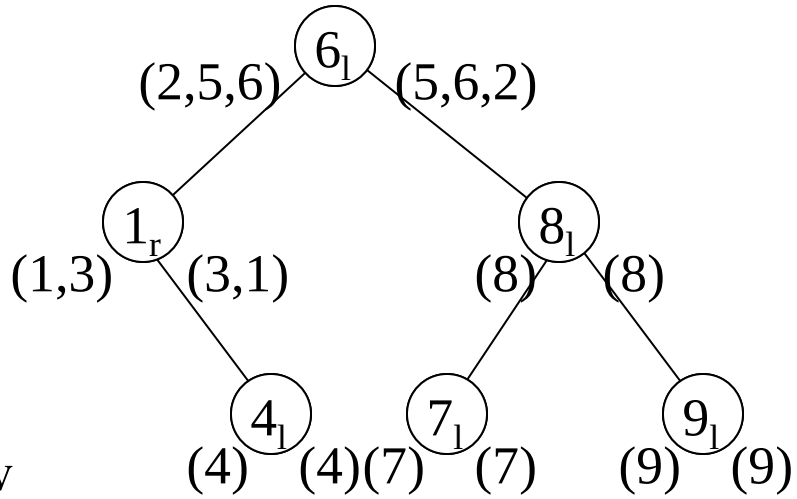
oblicz I_m, stwórz uporządkowane listy lewych (L_l) i prawych (L_r) końców odcinków z I_m i podłącz je do v;

ls(v) ← BUILDIT(I_l);

rs(v) ← BUILDIT(I_r);

return;

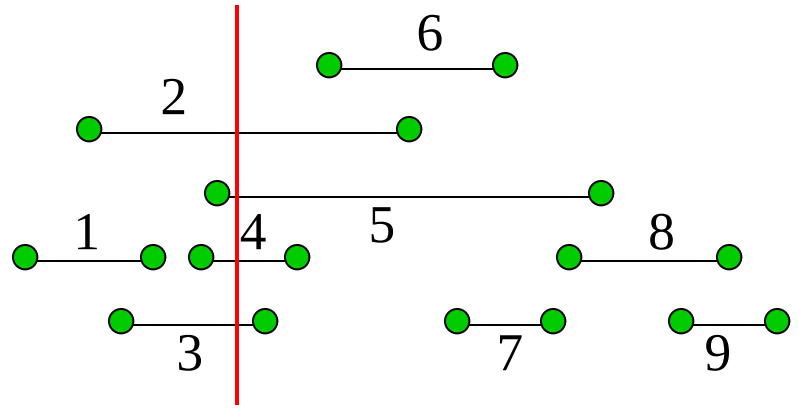


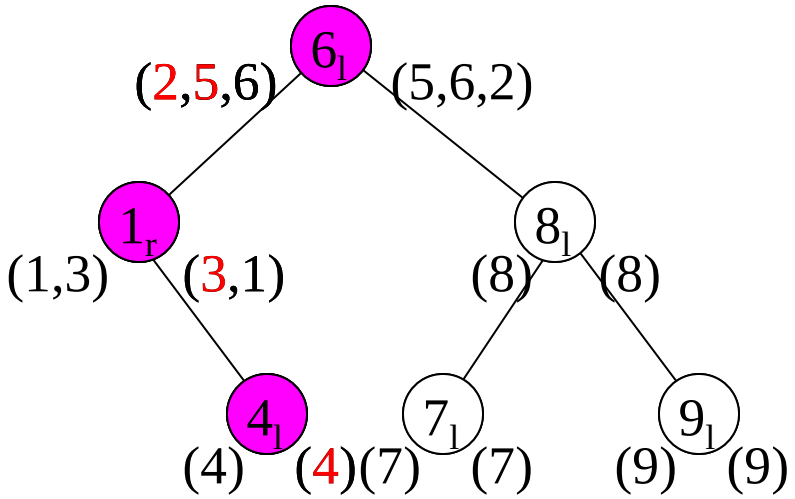


Niech v_x oznacza medianę x_m zapamiętaną w węźle v.

Szukamy odcinków, które zawierają dowolny punkt $q \in [x_1,x_2]$.

procedure SEARCHIT(v,q)
if v nie jest liściem
  then if $q < v_x$
    then badaj listę $L_l(v)$ zaczynając od najdalszego punktu; wypisuj odcinki, do których należy q, a gdy się skończą – zatrzymaj; SEARCHIT(ls(v),q);
    else badaj listę $L_r(v)$ zaczynając od najdalszego punktu; wypisuj odcinki, do których należy q, a gdy się skończą – zatrzymaj; SEARCHIT(rs(v),q);





Lemat.

Drzewo przedziałów dla n-elementowego zbioru I poziomych odcinków ma liniowy rozmiar i wysokość O(log n). Czas konstrukcji struktury wynosi O(n log n).

Lemat.

Znalezienie wszystkich poziomych odcinków z n-elementowego zbioru I przecinanych daną pionową prostą z wykorzystaniem drzewa przedziałów wymaga czasu O(k+log n), gdzie k jest liczbą znalezionych odcinków.

Jeśli chcemy znaleźć poziome odcinki przecinane przez pionowy odcinek (bok prostokąta R), to zamiast list L_l i L_r należy użyć np. drzew przeszukiwań priorytetowych.

Lemat.

Wszystkie poziome odcinki z n-elementowego zbioru I przecinane danym pionowym odcinkiem możemy znaleźć z pomocą drzewa przedziałów w czasie $O(k + \log^2 n)$, gdzie k jest liczbą znalezionych odcinków.

Dziękuję za uwagę.

Ćwiczenia 4.

1. Jak wyszukiwać punkty w jednowymiarowym obszarze ?

2. Jak zdefiniować liczby złożone w wyższych wymiarach ?

3. Udowodnij, że czas odpowiedzi w kd drzewie na zapytanie o punkty należące do kostki R w przestrzeni d-wymiarowej wynosi $O(k+n^{(d-1)/d})$, gdzie d jest stałą a k jest liczbą znalezionych punktów.

4. Częściowe skojarzenie, to zbiór punktów o określonej jednej (lub kilku) współrzędnej.

a) Pokaż, że dwuwymiarowe kd drzewo może odpowiedzieć na zapytanie o częściowe skojarzenie w pesymistycznym czasie $O(n^{1/2} + k)$, gdzie k jest liczbą znalezio-nych punktów.

b) Wytłumacz, jak skorzystać z dwuwymiarowego drzewa obszarów do odpowiedzi na zapytanie o częściowe skojarzenie. Jaki jest w efekcie czas zapytania ?

c) Opisz strukturę danych, która korzysta z liniowej pamięci i rozwiązuje dwuwymiarowe zapytanie o częściowe skojarzenie czasie $O(\log n + k)$. Jaki jest czas tworzenia struktury i odpowiedzi na klasyczne zapytanie ?

5. Zbuduj drzewo obszarów dla n-elementowego zbioru P w przestrzeni d-wymiarowej.

a) Jaki ma rozmiar i w jakim czasie można je zbudować ?

b) W jakim czasie znajdujemy odpowiedź na zapytanie o punkty należące do kostki R ?

6. Niech S_1 będzie zbiorem n rozłącznych półprostych pionowych, a S_2 zbiorem m rozłącznych półprostych poziomych. Opisz algorytm zamiatania, który policzy w czasie $O((n+m) \log (n+m))$ liczbę przecięć w $S_1 \cup S_2$.

7. W jaki sposób stworzyć listy pomocnicze w drzewie przedziałów w czasie $O(n \log n)$?

8. Niech P będzie zbiorem n punktów na płaszczyźnie posortowanych względem współrzędnej y. Pokaż, że z posortowania P wynika, że drzewo przeszukiwań priorytetowych dla punktów z P można zbudować w czasie $O(n)$.

9. Którą z poznanych struktur danych można wykorzystać do obsługi dynamicznie zmieniającego się zbioru punktów na płaszczyźnie ? Jaki byłby czas aktualizacji takiej struktury ?