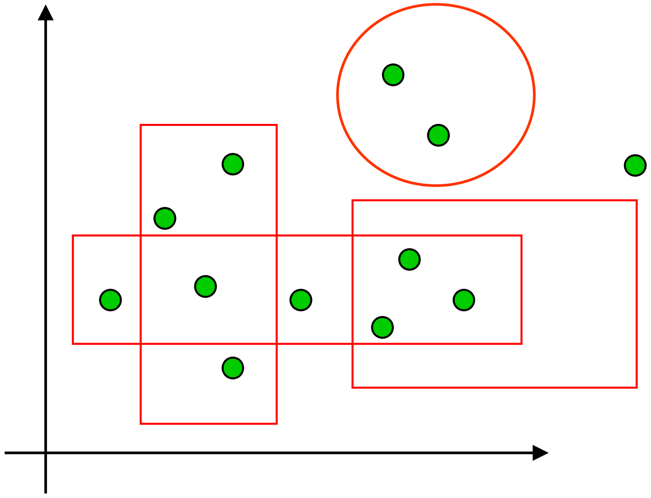
Wyszukiwanie geometryczne

Przeszukiwanie obszarów

- ✓ Dane
  - *P* – zbiór punktów,
  - *R* – obszar (region) - np. prostokąt, wielokąt, okrąg …
- ➢ Szukane
  - punkty (podzbiór *P*) leżące wewnątrz regionu *R*
    (lub ich liczba bądź też pewna funkcja agregująca)



Przeszukiwanie obszarów

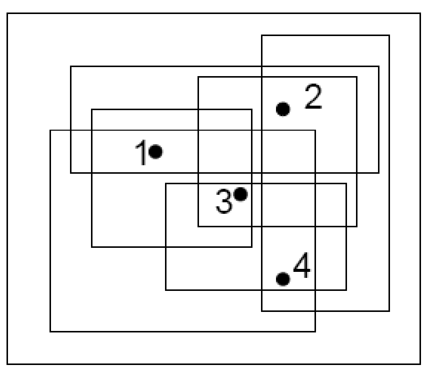
Ocena rozwiązania (struktury danych)

- czas potrzebny na realizację zapytania,
- pamięć potrzebna na przechowywanie danych,
- czas wstępnego przetworzenia,
- *czas aktualizacji*

Przeszukiwanie obszarów

<u>Podstawowe spostrzeżenie</u>

Liczba zbiorów, jakie mogą być efektem wyszukiwania obszaru geometrycznego jest znacznie mniejsza od liczby wszystkich możliwych podzbiorów P (zbioru potęgowego)



R = { {},
{1}, {2}, {3}, {4},
{1,2}, {1,3}, {2,3},{2,4},{3,4},
{1,2,3}, {1,3,4}, {2,3,4} ,
{1,2,3,4} }

$\cancel{\{1,4\}} \quad \cancel{\{1,2,4\}}$

Obszary prostokątne ortogonalne

- Regiony zdefiniowane jako prostokąty o bokach zgodnych z osiami współrzędnych
  - mogą być rozdzielone na zestaw wyszukiwań jednowymiarowych
- Typowe podejście – reprezentacja *P* jako kolekcji **podzbiorów kanonicznych** $\{S_1, S_2, \ldots, S_k\}$ (*k* zależy od *n* oraz typu regionu) takich, że dowolny zbiór wynikowy może być wyznaczony jako rozłączna suma podzbiorów kanonicznych (podzbiory te mogą się wzajemnie nakładać)

Wybór podzbiorów kanonicznych

- Wiele możliwości, wpływających na złożoność czasową i pamięciową
- Przykłady
  - n 1-elementowych zbiorów $\{p_i\}$
    - efektywne pamięciowo – O(n),
    - wynik zawierający k elementów – k podzbiorów (nieefektywne dla zliczania wyników)
  - zbiór potęgowy dla P
    - każde zapytanie reprezentowalne przez 1 podzbiór
    - możemy mieć 2^n zbiorów do przechowania

Wybór podzbiorów kanonicznych

- **Przypadek 1-wymiarowy**
  - zbiór punktów $P = \{p_1, p_2, \ldots, p_n\}$ na prostej
  - region – przedział $[x_{min}, x_{max}]$
  - oczekiwana złożoność czasowa: O(log n + k), gdzie k jest liczbą wynikowych punktów (wrażliwość na rozmiar wyniku)
  - dla zadania wyznaczenia liczby punktów w przedziale, można uzyskać złożoność O(log n)

<u>w jaki sposób?</u>

Przeszukiwanie 1-wymiarowe

Metoda 1

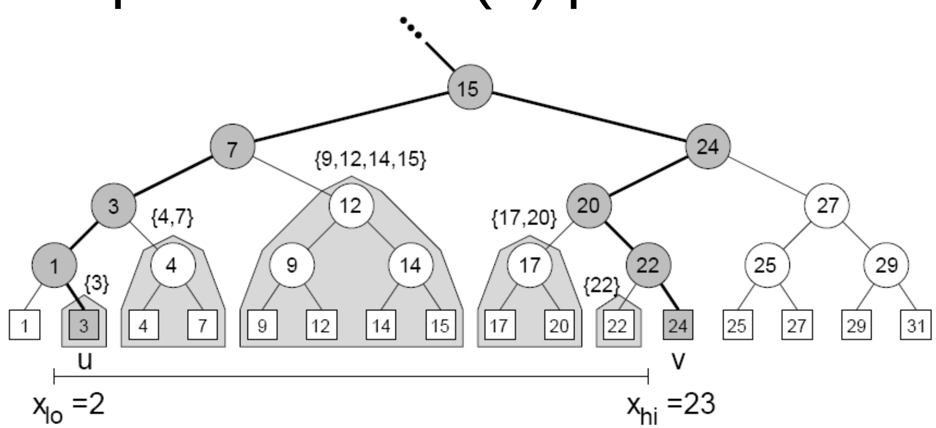
- wstępnie posortować punkty rosnąco
- wyznaczyć najmniejszy punkt $p_i \geq x_{min}$
- wyznaczyć największy punkt $p_j \leq x_{max}$
- zwrócić wszystkie punkty pomiędzy p_i i p_j

Wada: metoda nie da się uogólnić dla wyższych wymiarów…

Przeszukiwanie 1-wymiarowe

Metoda 2

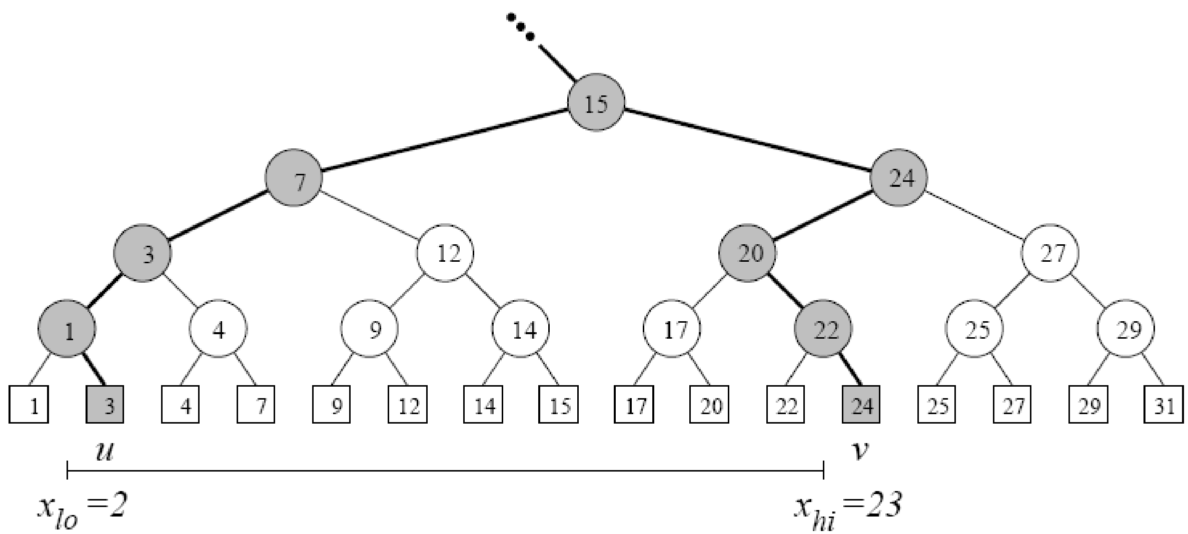
- posortować punkty rosnąco
- zapisać je w liściach zrównoważonego drzewa binarnego
- każdy element drzewa nie będący liściem zostaje oznaczony największą wartością występującą w lewym poddrzewie
- każdy element drzewa może być powiązany (jawnie lub nie) z podzbiorem punktów – O(*n*) podzbiorów kanonicznych



Przeszukiwanie 1-wymiarowe

– odszukać pierwszy z lewej liść *u* o wartości większej lub równej x_{min}

– odszukać pierwszy z prawej liść *v* o wartości większej lub równej x_{max}

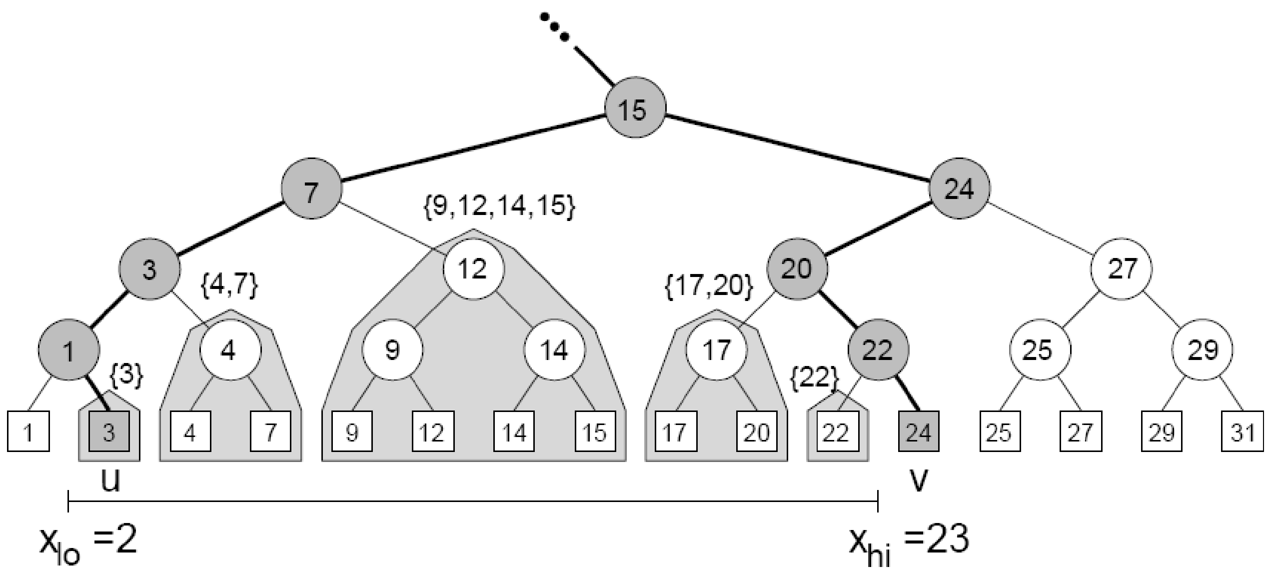


Przeszukiwanie 1-wymiarowe

– wszystkie obiekty w poddrzewach na prawo od lewej ścieżki

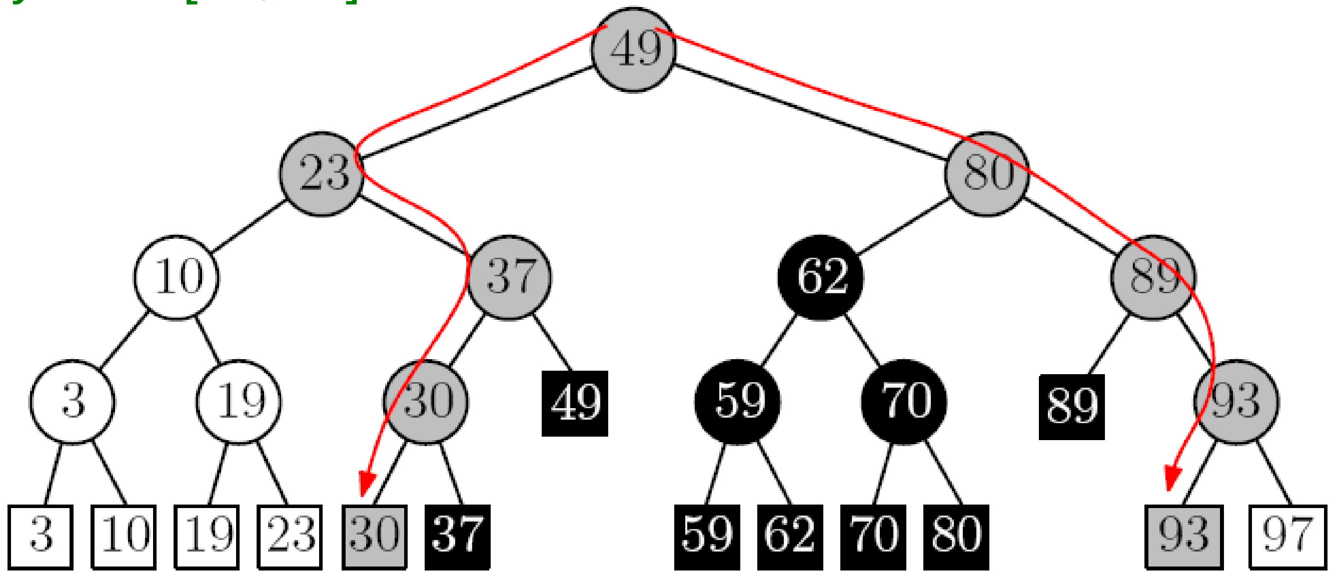
– wszystkie obiekty w poddrzewach na lewo od prawej ścieżki

– obiekty w liściach *u* i *v*, zależnie od wartości



Przeszukiwanie 1-wymiarowe

Zapytanie: [25, 90]



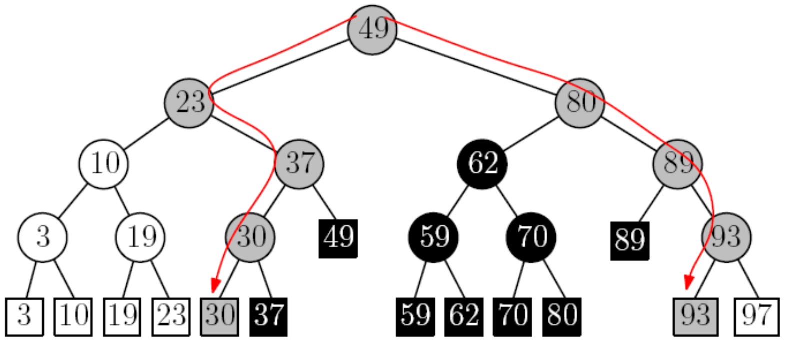
Białe węzły – nigdy nie odwiedzone w trakcie zapytania
Szare węzły – odwiedzone; być może należą do „*odpowiedzi*"
Czarne węzły – całe poddrzewo jest *wyjściem*

Przeszukiwanie 1-wymiarowe

– złożoność pamięciowa: *O(n)*

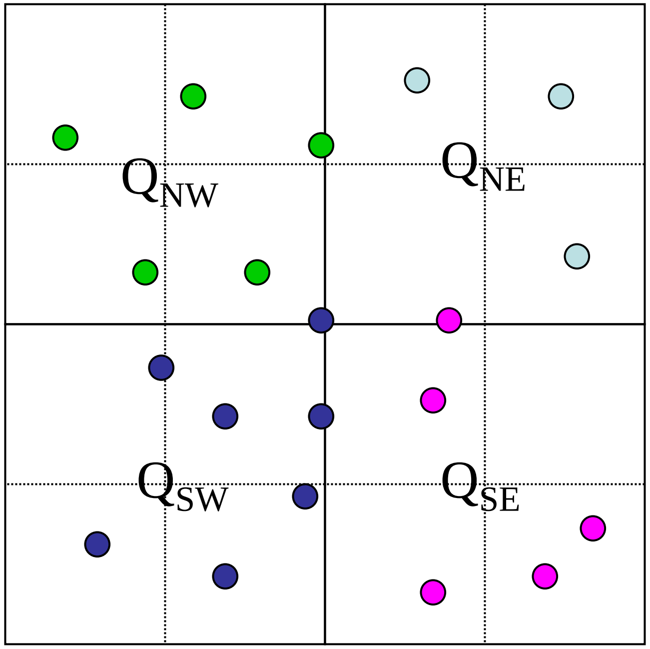
– złożoność czasowa

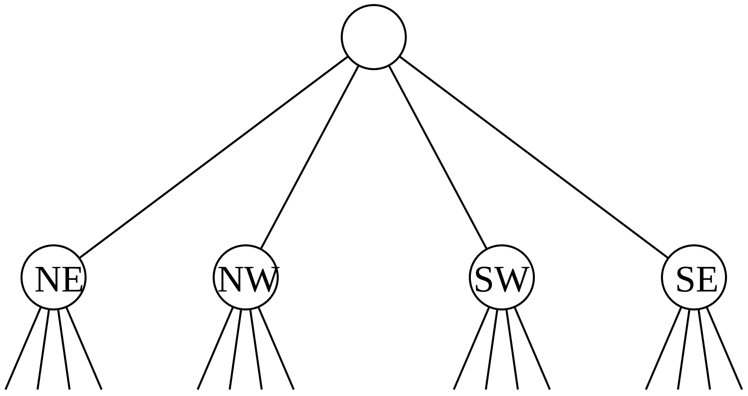
- podanie liczby obiektów: *O(log n)*
- ustalenie zbioru obiektów: *O(log n + k)*



Przeszukiwania k-wymiarowe

drzewa ćwiartkowe (quadtree, octree)

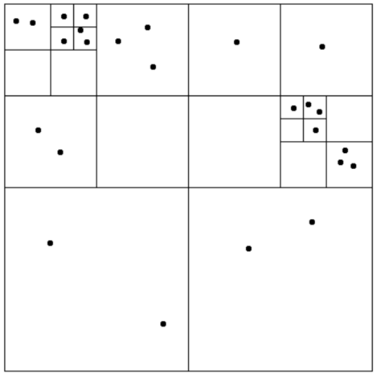


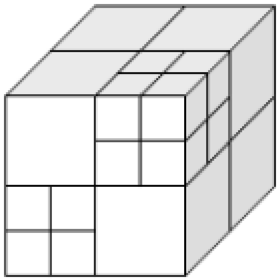


Przeszukiwania k-wymiarowe

drzewa ćwiartkowe (quadtree, octree)

- łatwe w implementacji
- przydatne w wielu zastosowaniach
- dla przeszukiwania może być bardzo nieefektywne w pesymistycznym przypadku

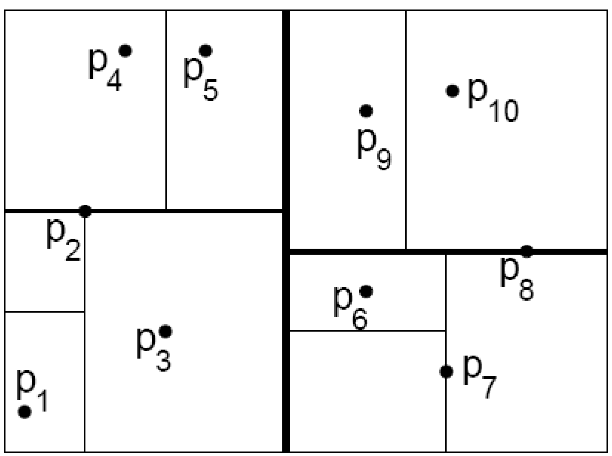


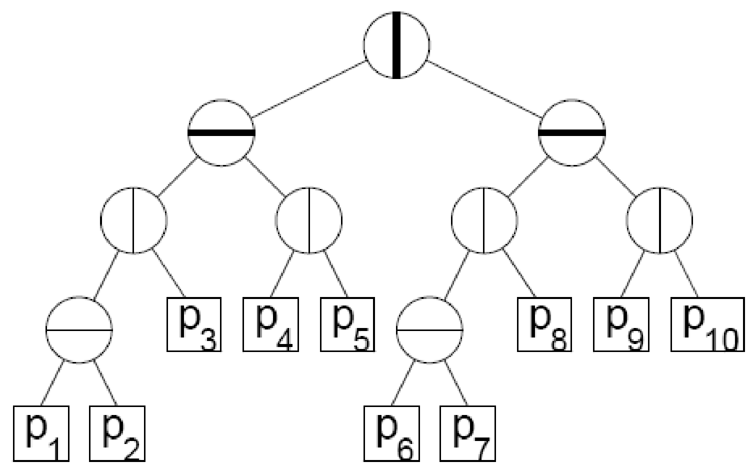


Przeszukiwanie k-wymiarowe

kd-drzewa (kd-trees, *k*-d trees)

- uogólnienie drzewa przeszukiwania 1-wymiarowego
- praktyczne i łatwe w implementacji
- użyteczne w wielu problemach przeszukiwania

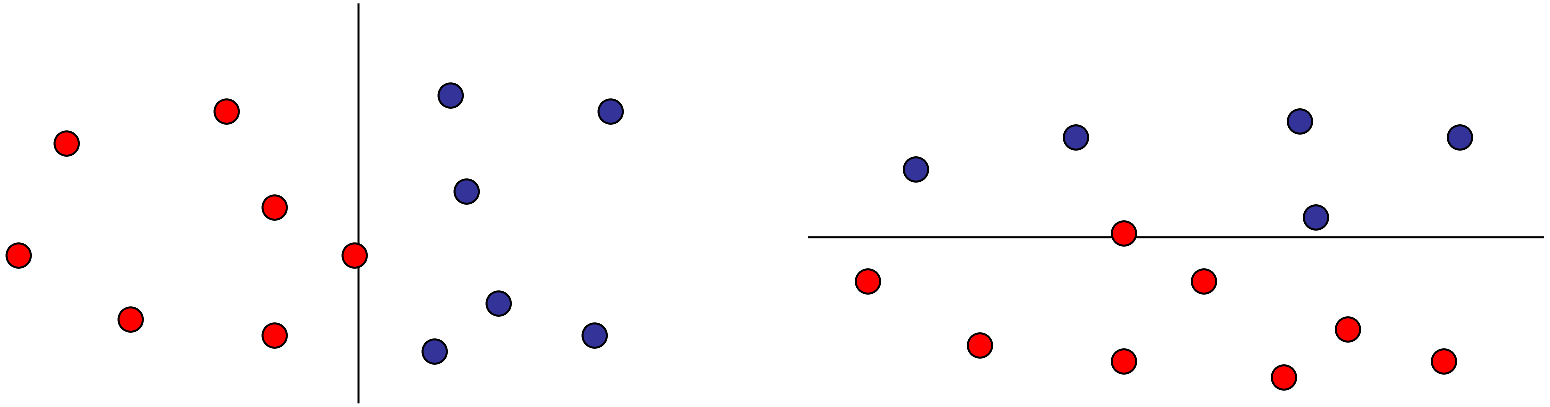


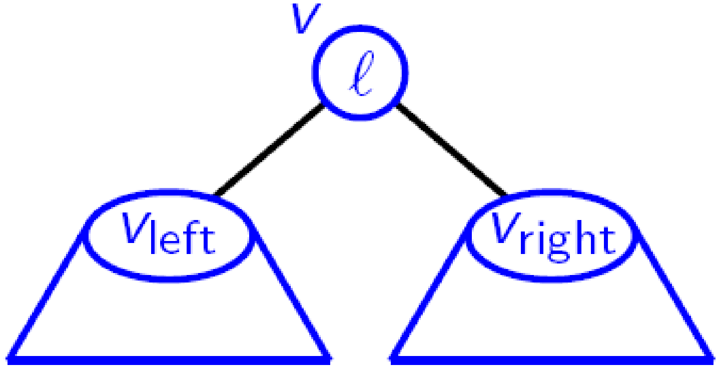


Struktura kd-drzewa

- liście
  - obiekty geometryczne – punkty
- węzły
  - wymiar, względem którego wykonany jest podział
  - wartość współrzędnej podziału w wybranym wymiarze
    - obiekty mniejsze w lewym poddrzewie
    - obiekty większe w prawym poddrzewie
    - obiekty równe w lewym lub prawym poddrzewie (dla zrównoważenia)
  - liczba obiektów w danym poddrzewie

Tworzenie kd-drzewa





Tworzenie kd-drzewa

Wejście: zbiór P i obecna głębokość *depth*

Wyjście: korzeń kd-drzewa dla zbioru P

Początkowo: P – cały zbiór
depth=0

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;

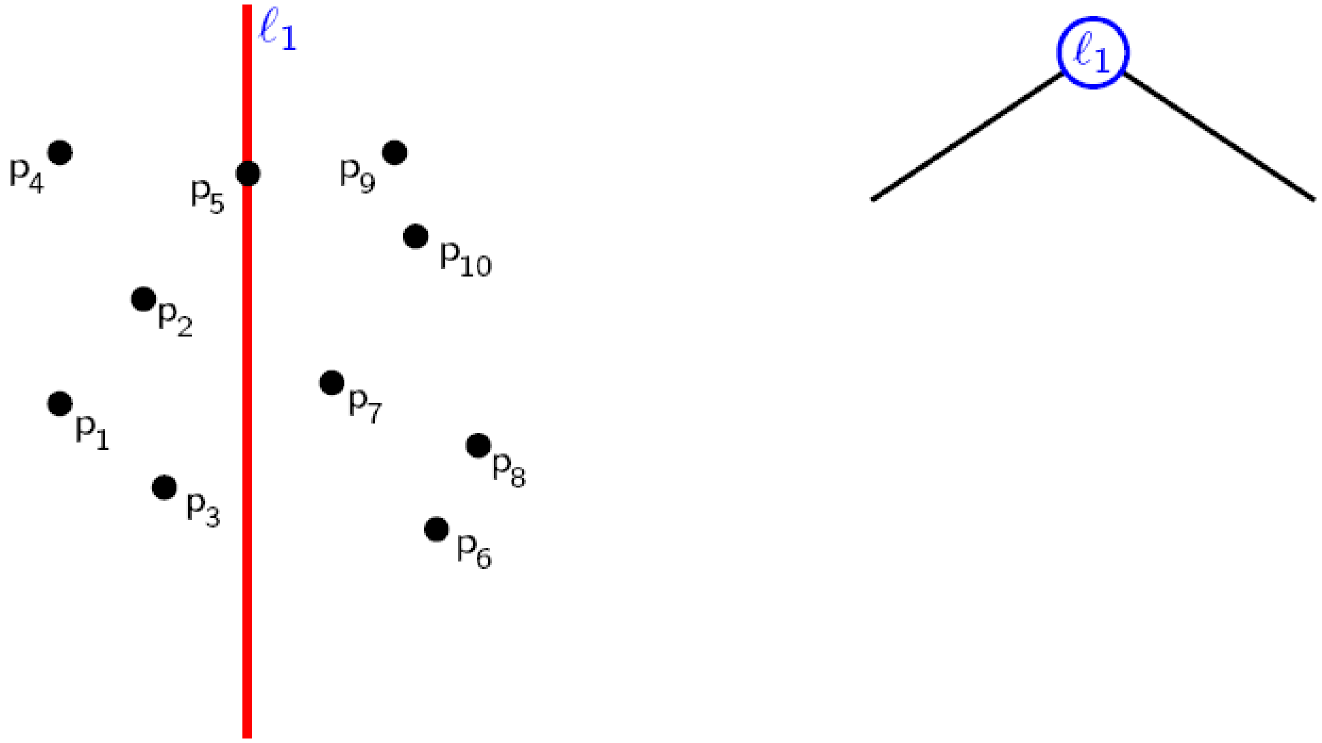
$v_l \leftarrow$ BUILDTREE(P_1,depth+1);

$v_p \leftarrow$ BUILDTREE(P_2,depth+1);

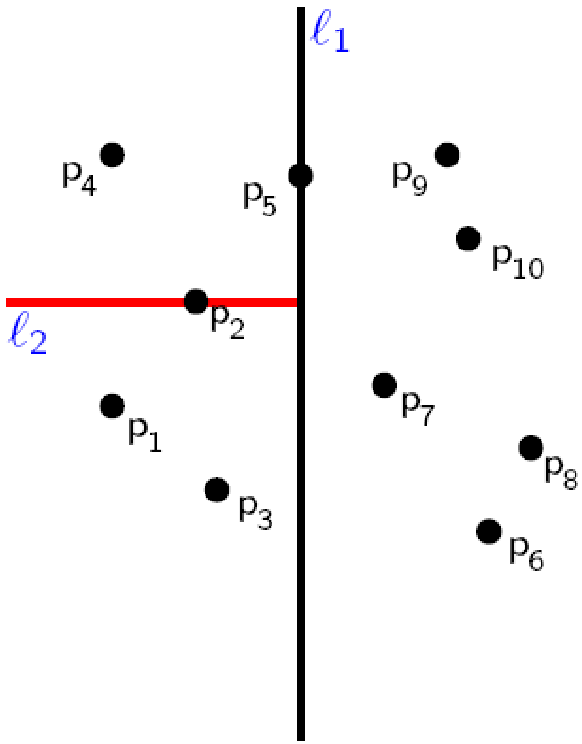
stwórz węzeł v – ojca v_l i v_p oraz zapamiętaj w nim l;

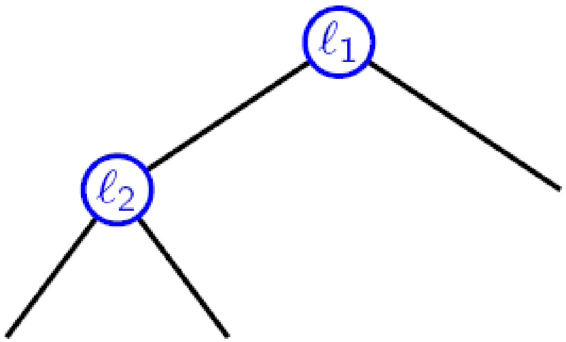
return v

Tworzenie kd-drzewa

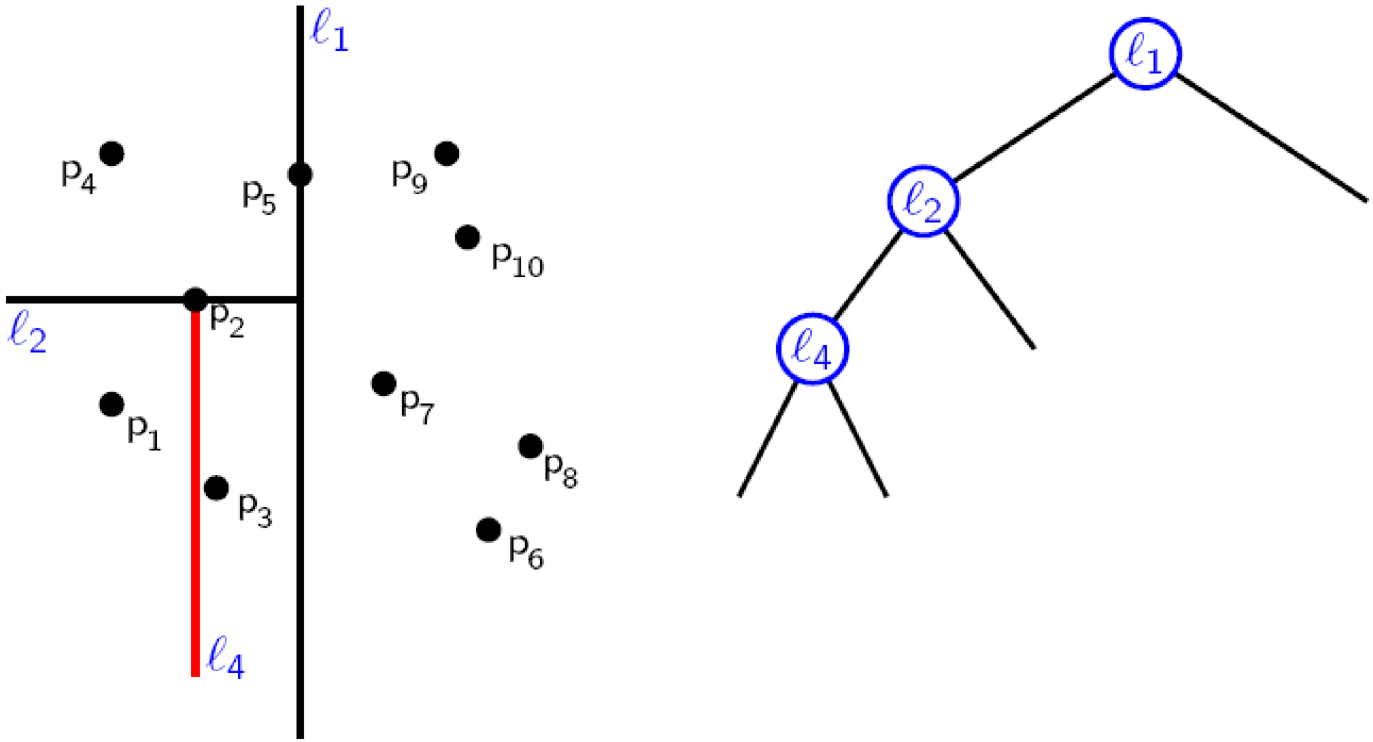


Tworzenie kd-drzewa

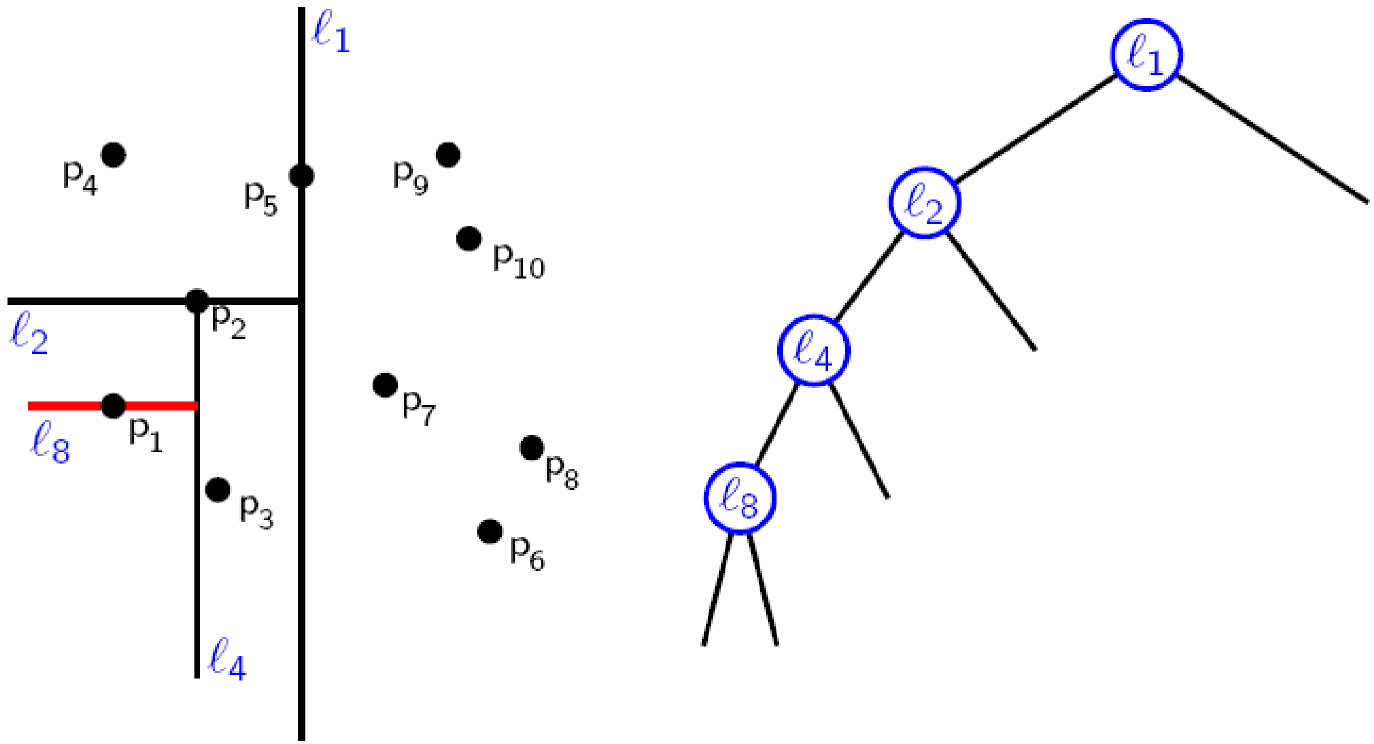




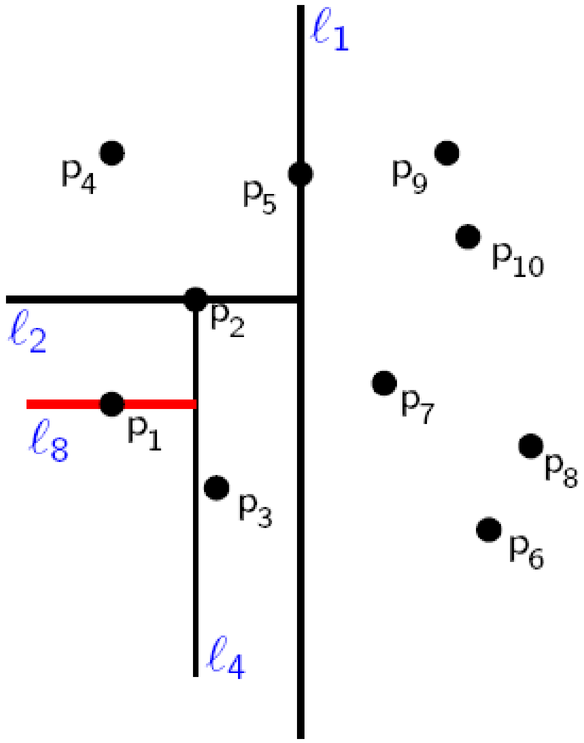
Tworzenie kd-drzewa

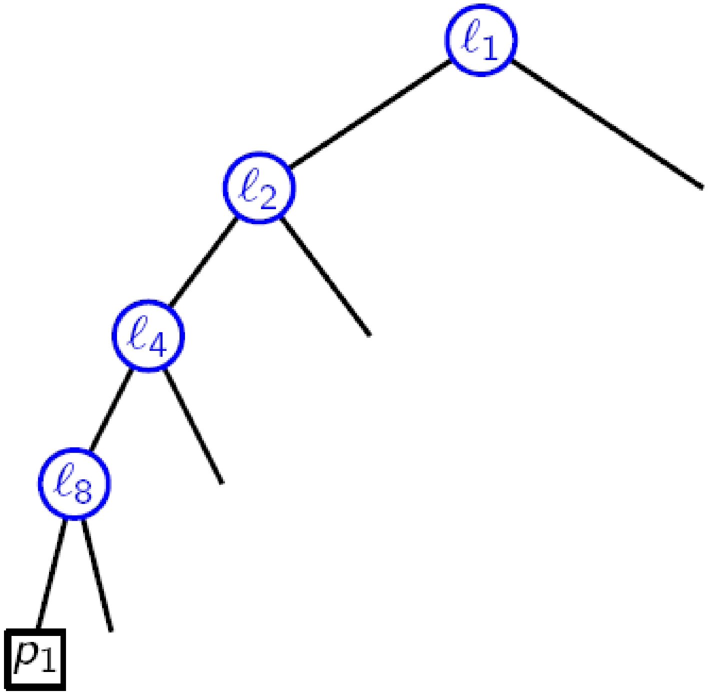


Tworzenie kd-drzewa

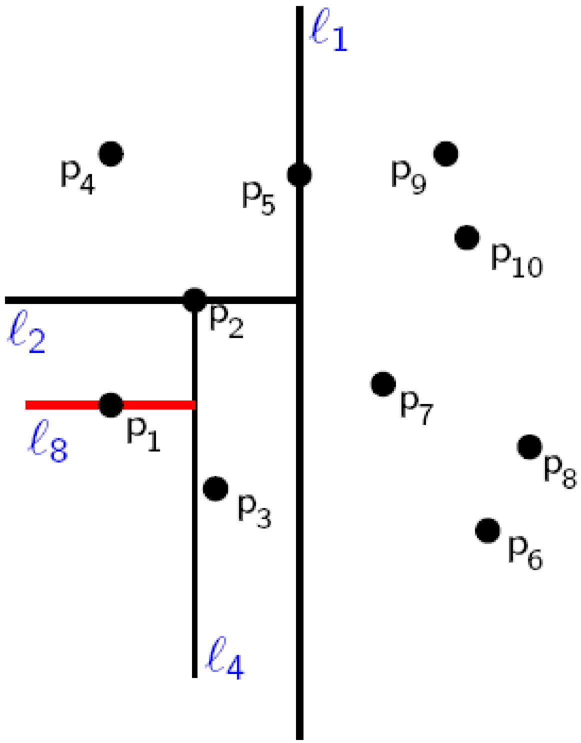


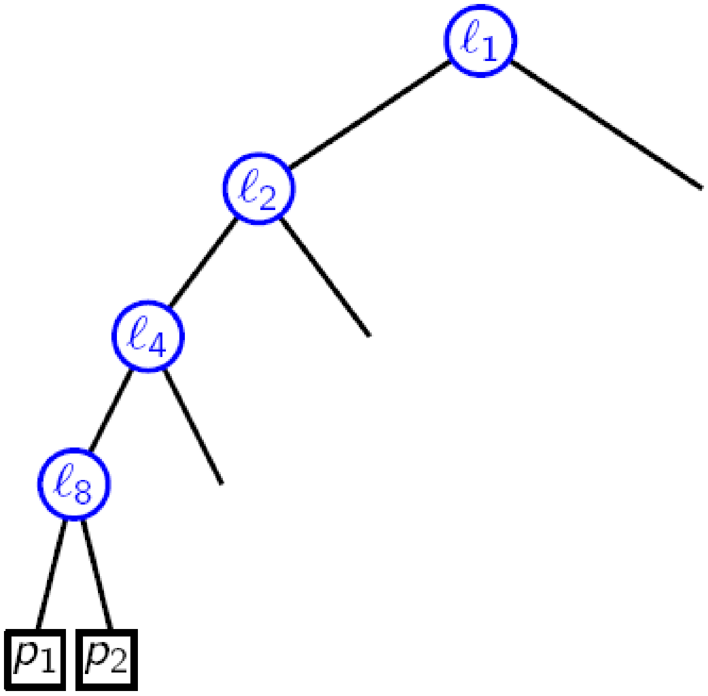
Tworzenie kd-drzewa



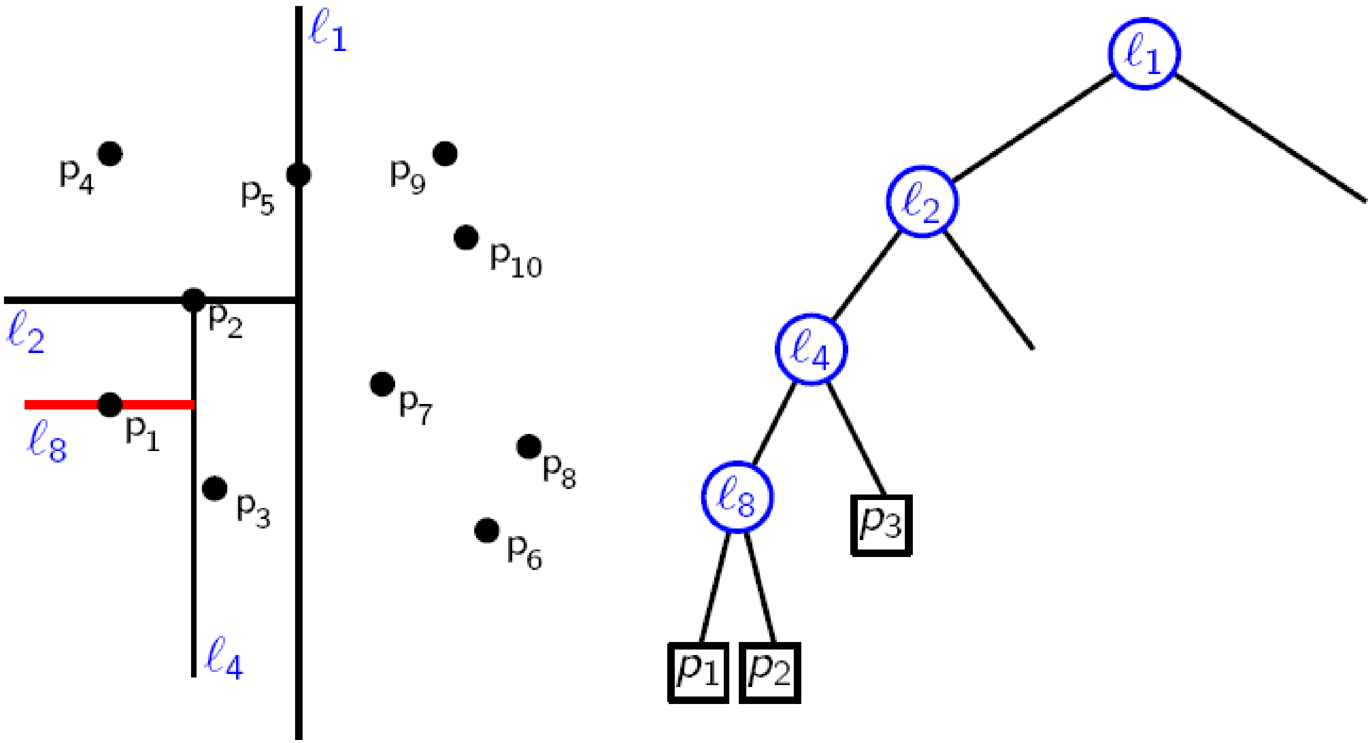


Tworzenie kd-drzewa

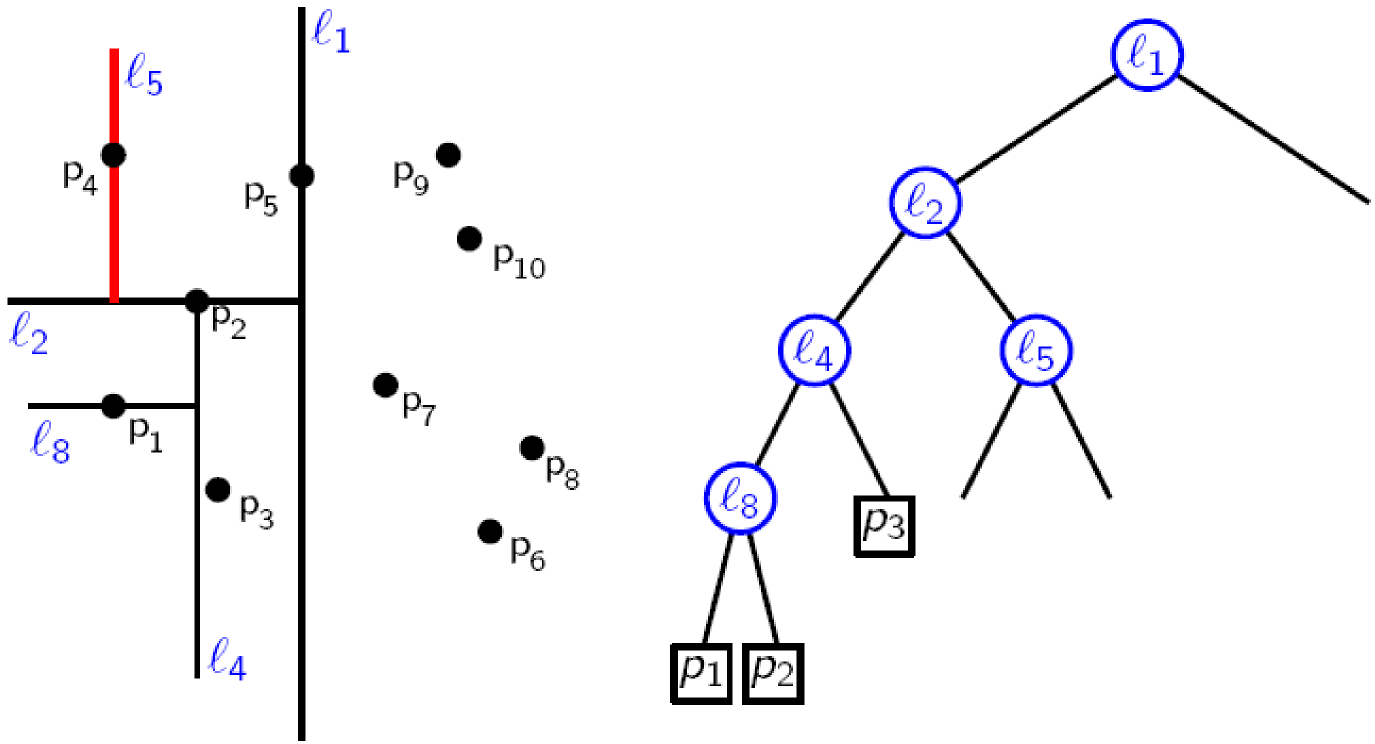




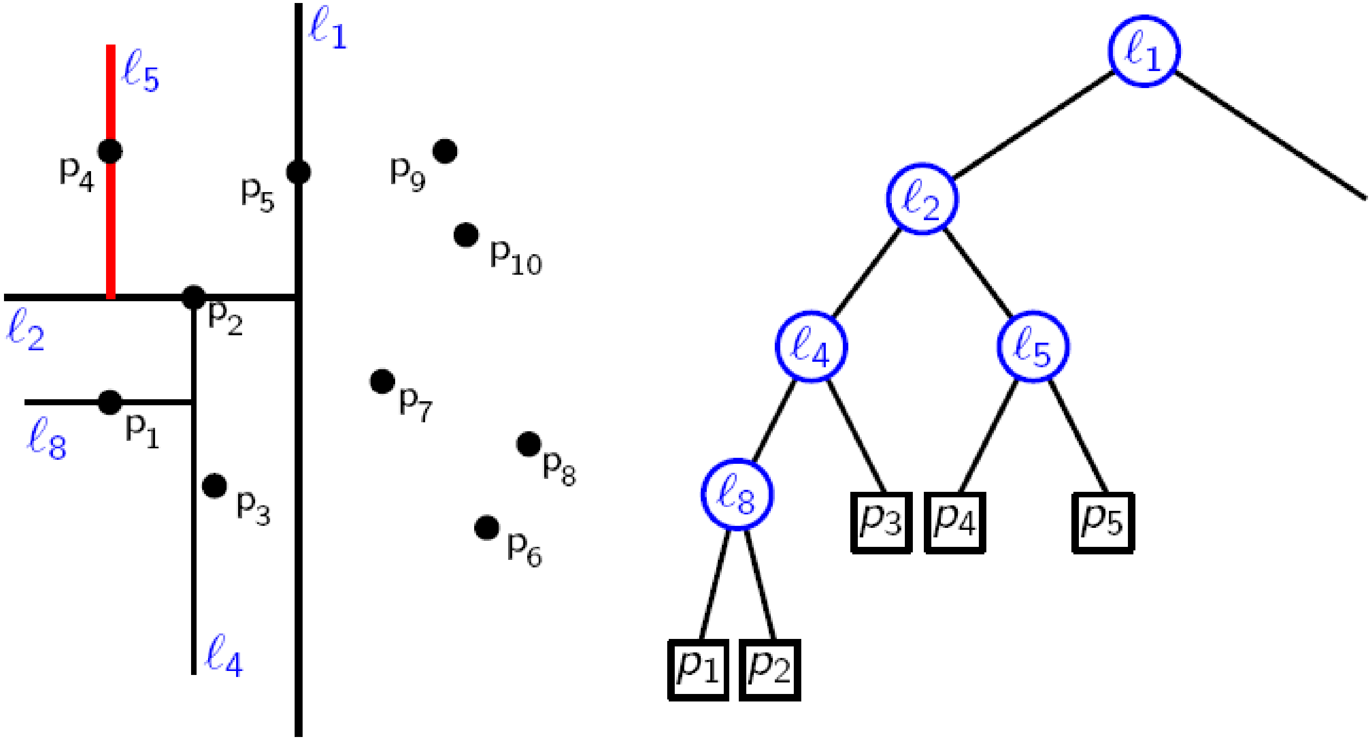
Tworzenie kd-drzewa



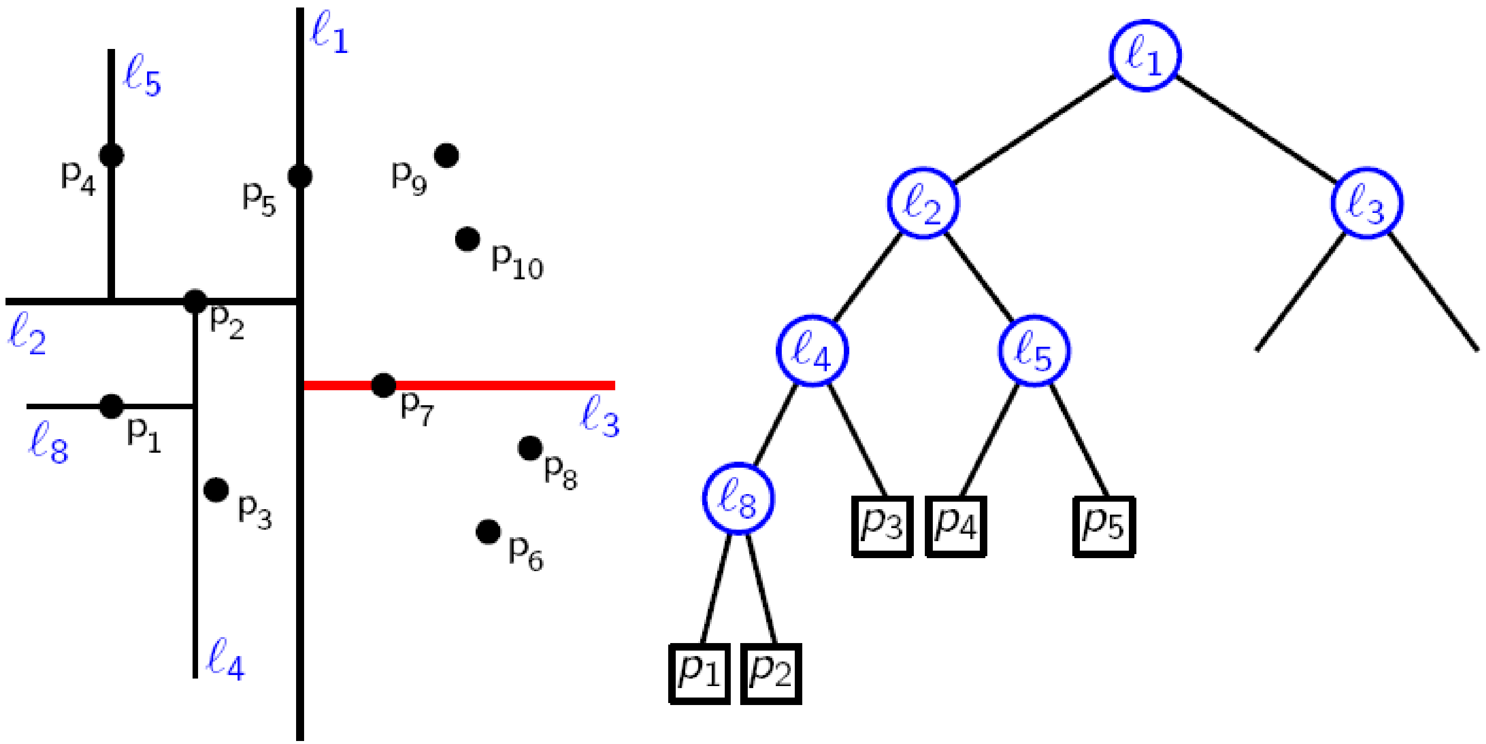
Tworzenie kd-drzewa



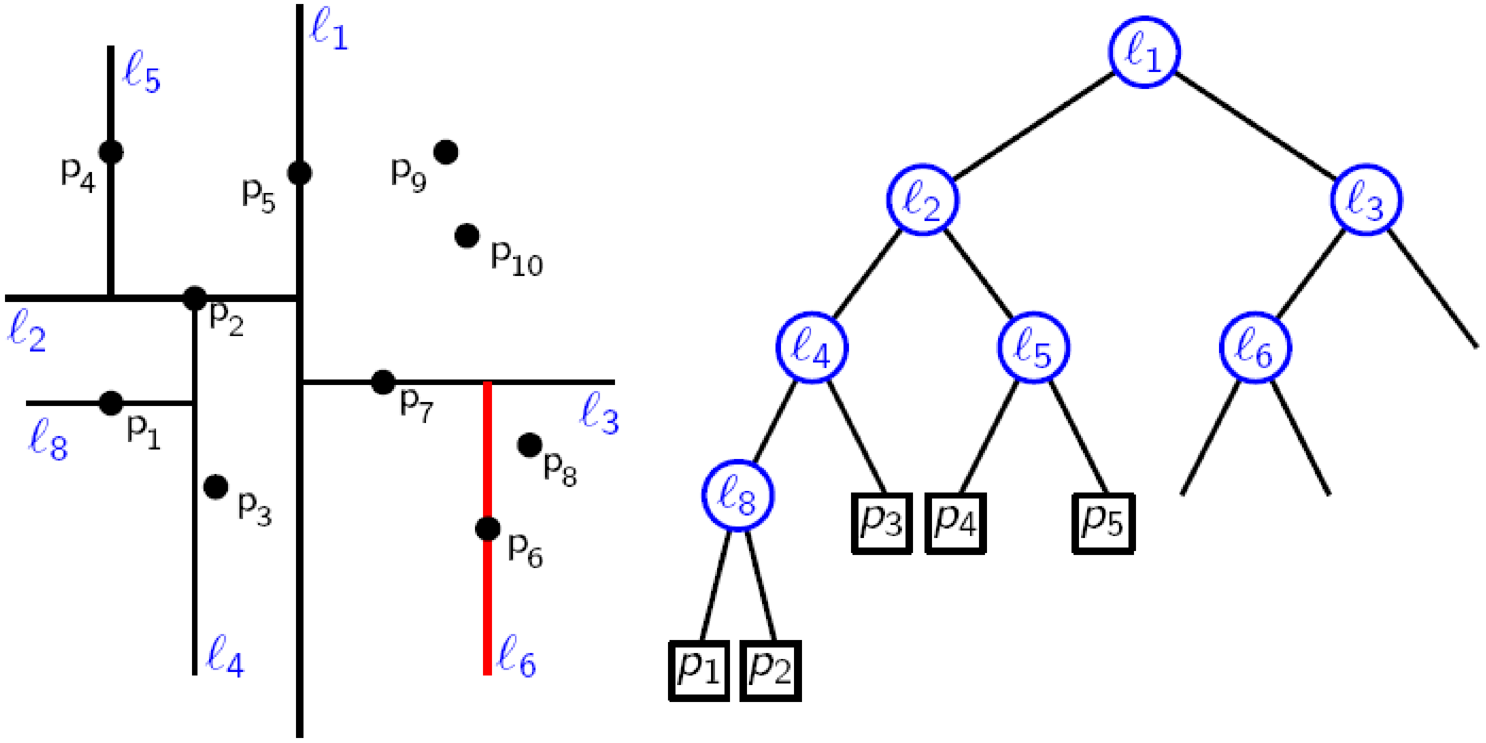
Tworzenie kd-drzewa



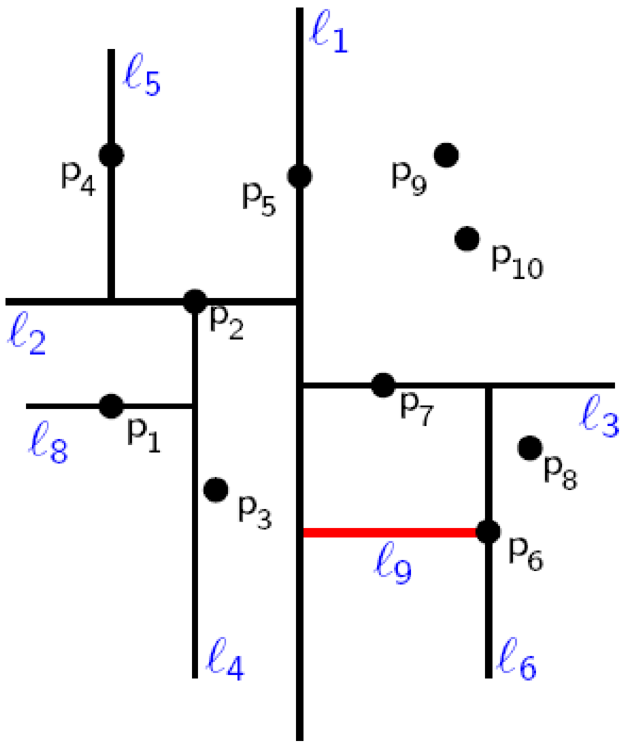
Tworzenie kd-drzewa

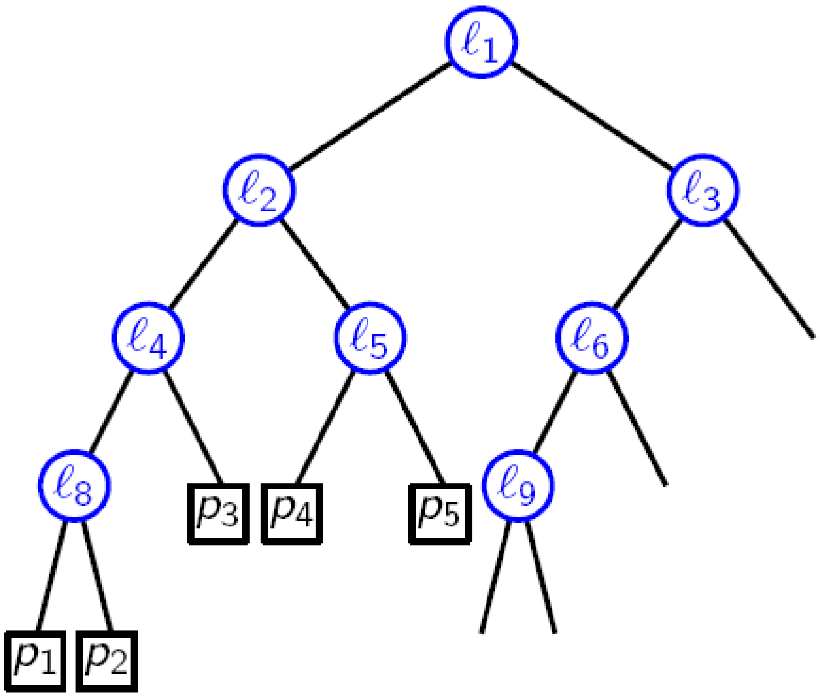


Tworzenie kd-drzewa

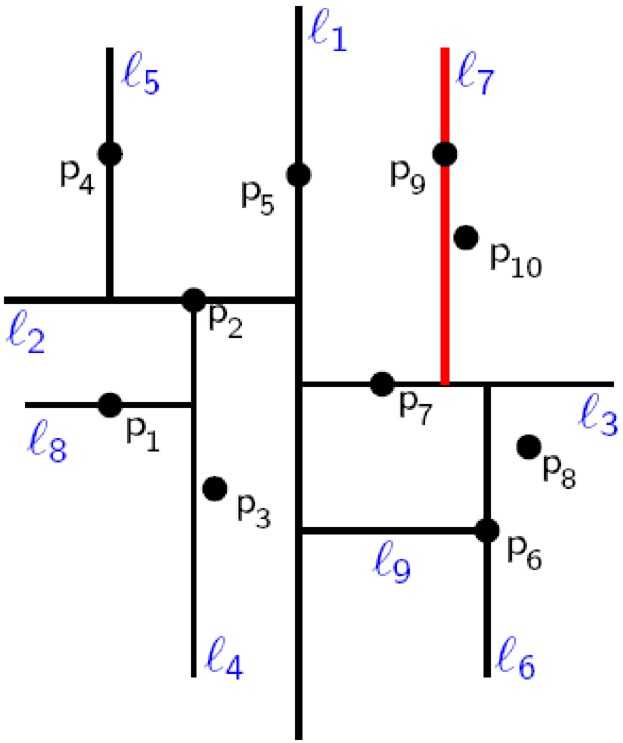


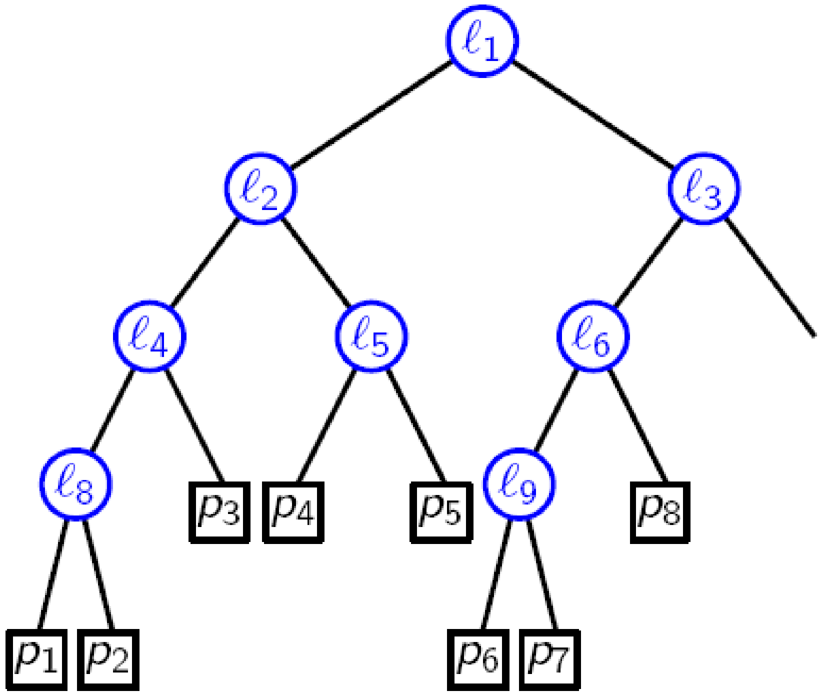
Tworzenie kd-drzewa



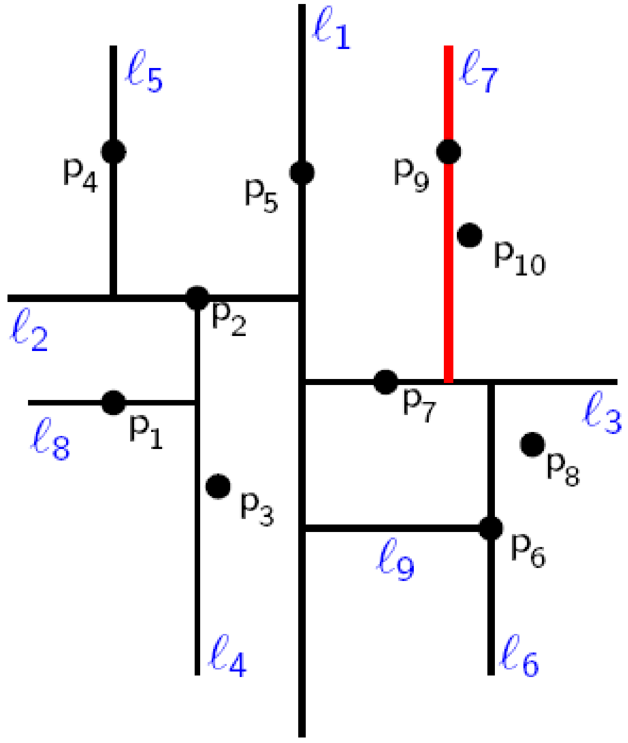


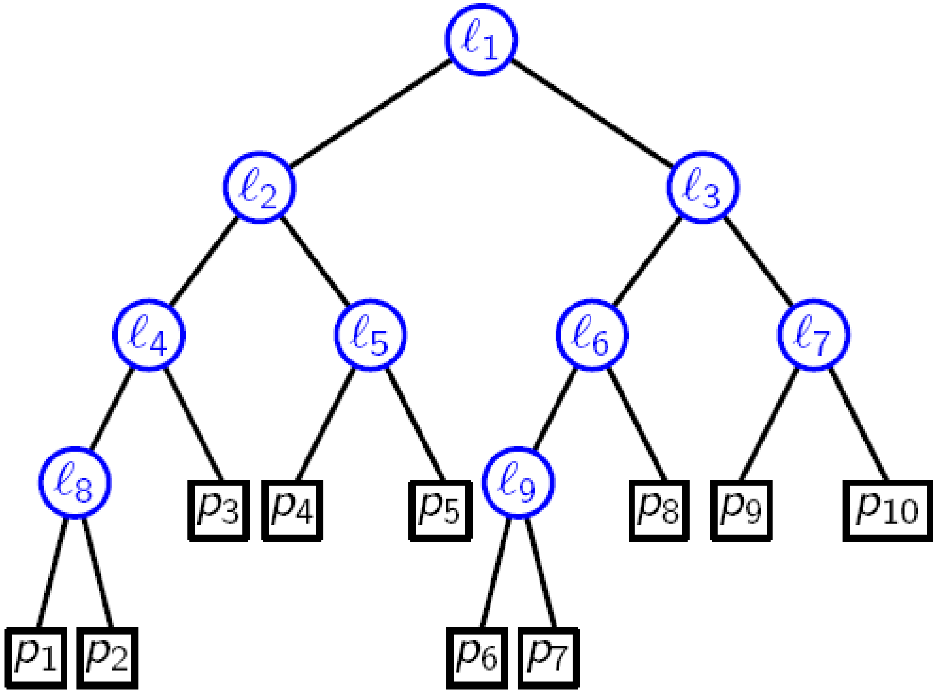
Tworzenie kd-drzewa





Tworzenie kd-drzewa





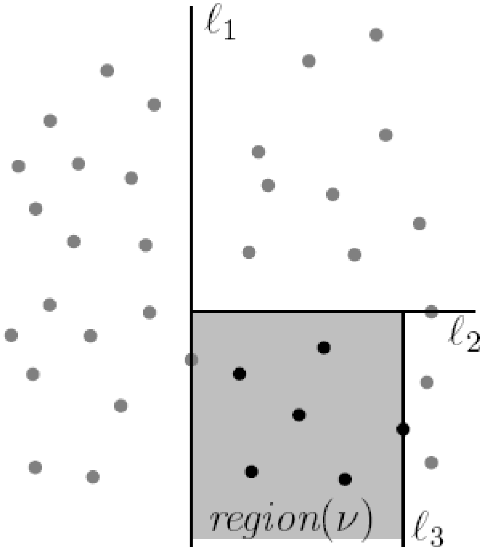
Tworzenie kd-drzewa

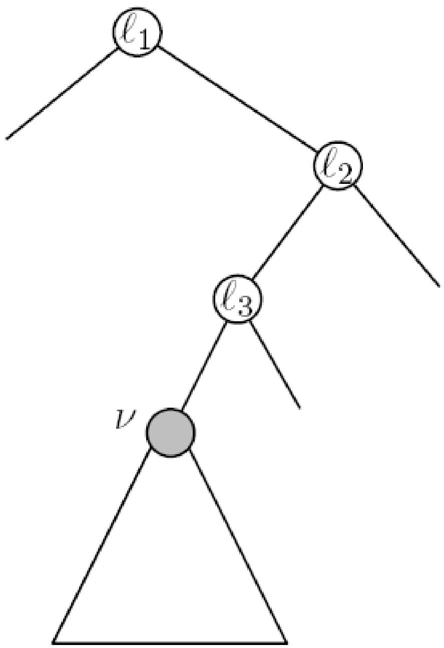
- Jak wybrać wymiar dla podziału?
  - kolejne wymiary
    - nie trzeba przechowywać tej informacji jawnie w drzewie
    - mogą pojawić się wydłużone podobszary
  - wymiar, dla którego współrzędne punktów mają największą różnicę
    - pozwala uzyskać lepszą strukturę drzewa
- Jak wybrać wartość dla podziału?
  - mediana względem wybranego wymiaru
    - dla zapewnienia głębokości drzewa $O(\log n)$

Tworzenie kd-drzewa

- Złożoność czasowa $O(n \log n)$
- Najkosztowniejszy krok
  - ustalenie mediany dla podziału
    - *k* list/tablic punktów (wskaźników), posortowanych według wartości współrzędnych dla kolejnych wymiarów
      - wstępne sortowanie $O(n \log n)$
      - ustalenie mediany $O(1)$
      - podział list $O(n)$

kd-drzewa

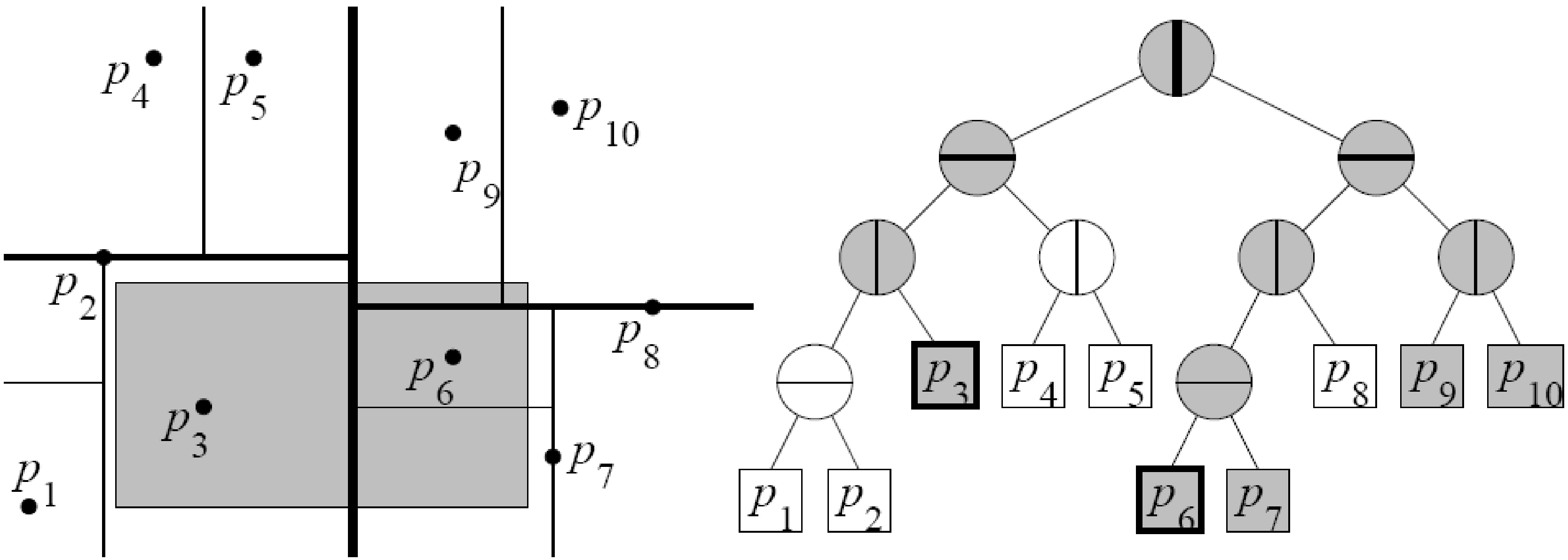




W jaki sposób znamy *region*(*v*)?

- opcja 1: zapamiętany w wierzchołku *v*
- opcja 2: obliczany przy przechodzeniu po drzewie

Przeszukiwanie kd-drzewa



Białe węzły – region R nie przecina *region*(*v*)

Szare węzły – region R przecina *region*(*v*), ale *region*(*v*) $\not\subseteq$R

Czarne węzły – *region*(*v*) $\subseteq$ R

Przeszukiwanie kd-drzewa

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

SEARCHKD(*v*,*R*)

if v jest liściem ***then***
    if $v \in R$ ***then*** zwróć v
    else
      if *region* (ls(v)) $\subseteq R$
        then zwróć wszystkie liście poddrzewa o korzeniu w ls(v)
        else if *region*(ls(v)) przecina R ***then***
                SEARCHKD(ls(v),R)
      if *region* (rs(v)) $\subseteq$ R
        then zwróć wszystkie liście poddrzewa o korzeniu w rs(v)
        else if *region* (rs(v)) przecina R ***then***
                SEARCHKD(rs(v),R)

Przeszukiwanie kd-drzewa

- Dla zrównoważonego drzewa
  - złożoność czasowa zliczania: $O(\sqrt{n})$
  - złożoność czasowa wyszukiwania: $O(\sqrt{n} + k)$
  - złożoność pamięciowa: $O(n)$
- Jak wykazać?
  - liczba odwiedzanych węzłów: $O(\sqrt{n})$
  - liczba odwiedzanych węzłów = liczba przecinanych komórek drzewa

Przeszukiwanie kd-drzewa

- **Lemat**:
  dla zrównoważonego kd-drzewa z zamiennym podziałem, dowolna pionowa lub pozioma prosta przecina $O(\sqrt{n})$ *komórek*.
- Załóżmy linię pionową $x=x_0$
  - dla podziału pionowego, prosta przecina lewy lub prawy podwęzeł
  - dla podziału poziomego, prosta przecina oba
  - ponieważ podział jest zamienny, podwaja się co dwa poziomy – przecina co najwyżej 2 węzły na 2 poziomie, 4 na 4 poziomie, 2^i na 2i poziomie

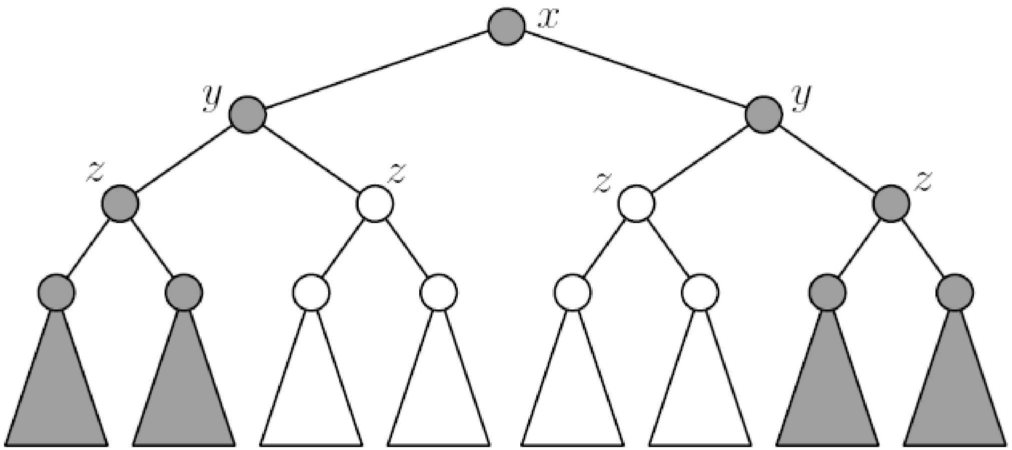
Przeszukiwanie kd-drzewa

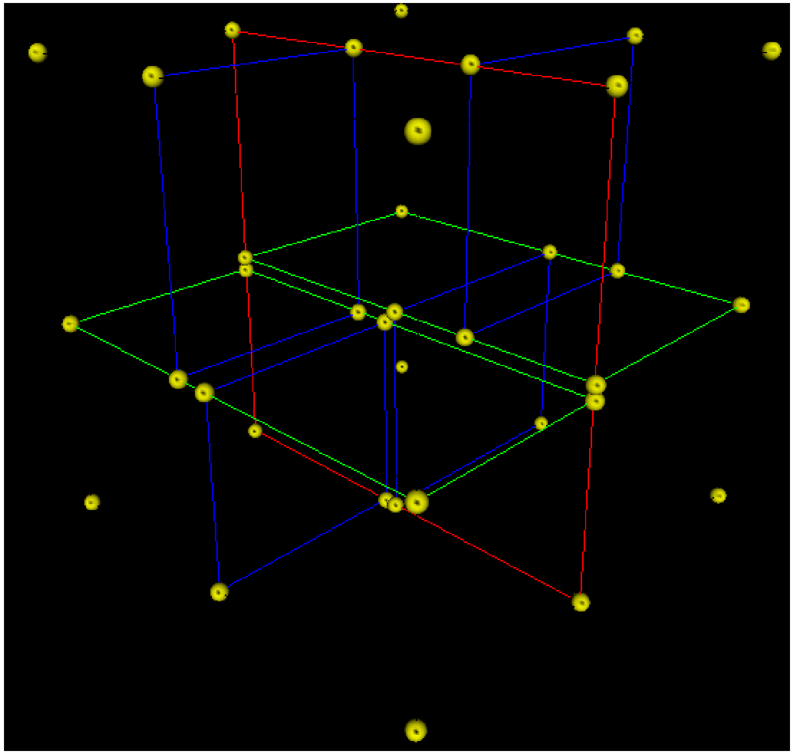
- Ponieważ wzrost jest potęgowy, suma jest zdominowana przez ostatni składnik – liczbę przeciętych komórek na najniższym poziomie drzewa.
- Drzewo jest zrównoważone – log *n* poziomów
- Liczba komórek przeciętych na najniższym poziomie przez prostą: $2^{(\log n)/2} = 2^{\log \sqrt{n}} = \sqrt{n}$
- Liczba komórek przeciętych przez prostokąt

- … stąd złożoność obliczeniowa całego procesu

$$O(4\sqrt{n}) = O(\sqrt{n})$$

kd-drzewa w wyższym wymiarze





Zakresy prostokątne ortogonalne

Ortogonalne **drzewa obszarów**

- wielopoziomowe drzewa wyszukiwania – dekompozycja złożonego zapytania na skończoną liczbę prostszych zapytań o zakres
- zamiana przeszukiwania d-wymiarowego na zestaw zapytań 1-wymiarowych

Drzewa obszarów

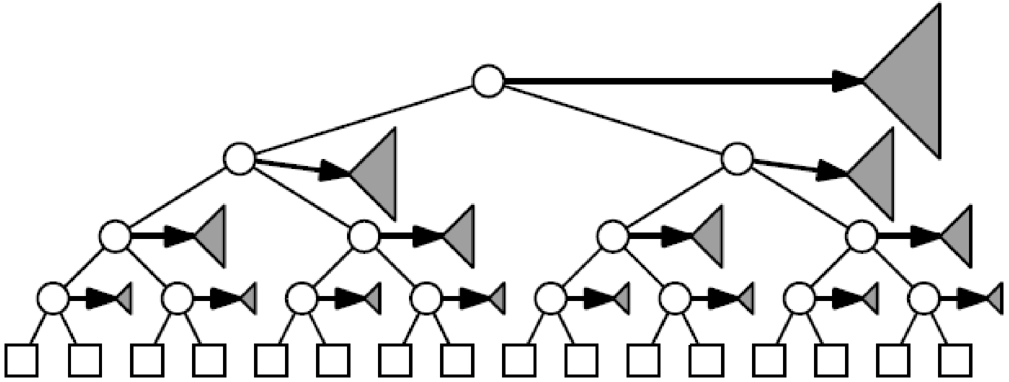
Przykład 2d

- zapytanie o prostokątny zakres – dwa zapytania o przedziały: $[x_{\mathrm{min}}, x_{\mathrm{max}}]$ oraz $[y_{\mathrm{min}}, y_{\mathrm{max}}]$
- zakładamy, że mamy przygotowane 1-wymiarowe drzewo przeszukiwania obszaru dla współrzędnej x
  - zrównoważone drzewo binarne
  - z każdym węzłem niejawnie powiązany podzbiór kanoniczny
  - odpowiedź na zapytanie – suma niewielkiej liczby $m=O(\log n)$ podzbiorów $\{S_1, S_2, \ldots, S_m\}$

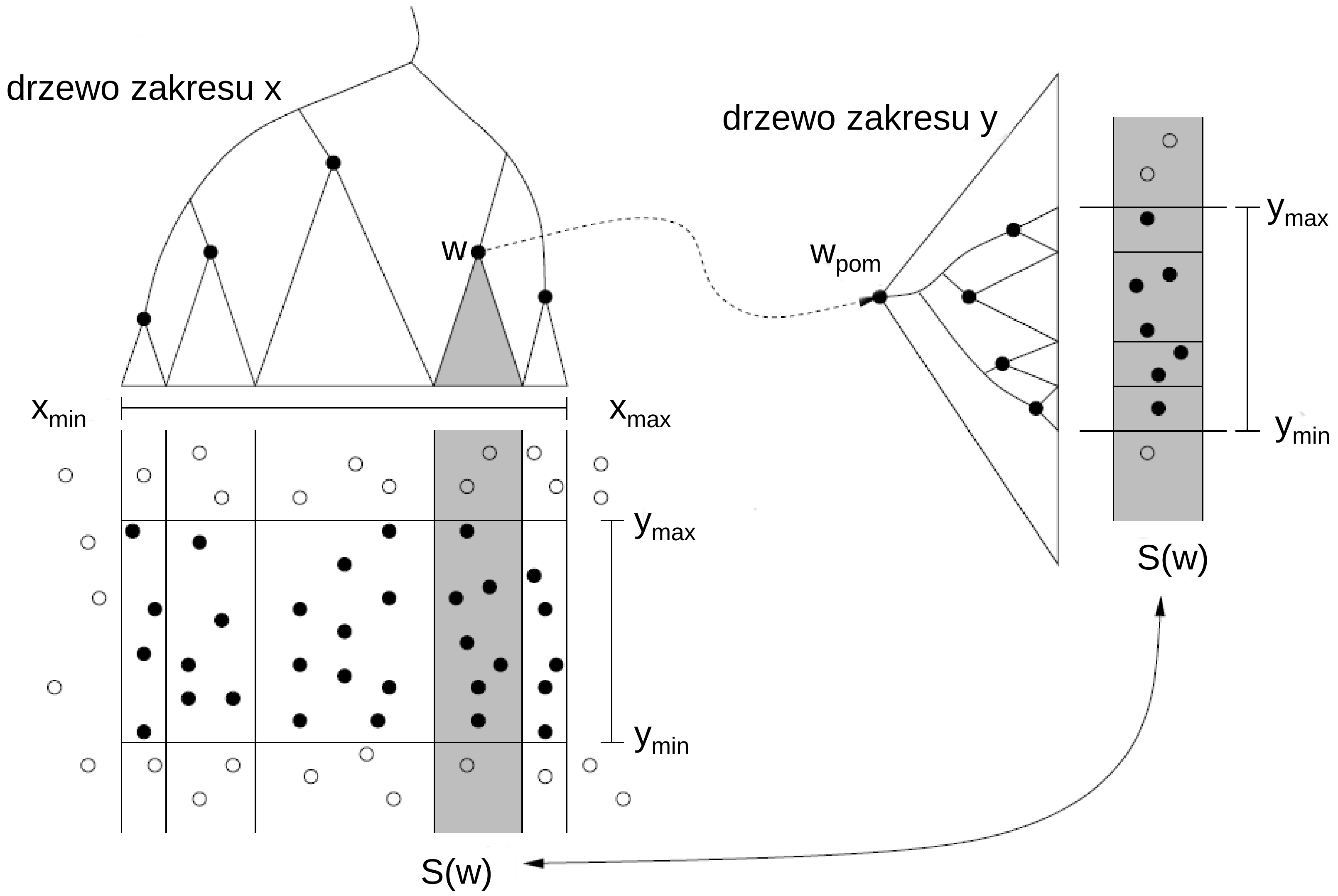
Drzewa obszarów

Drugi poziom

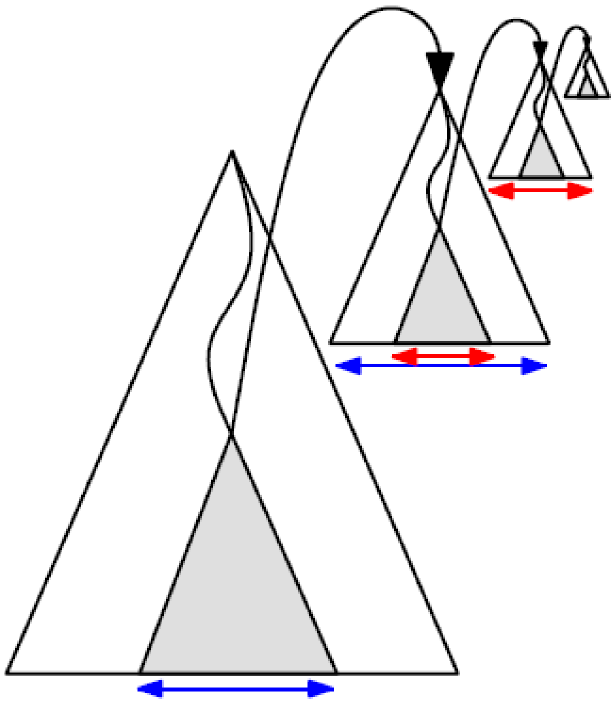
– dla każdego węzła *w* tego drzewa przeszukiwania zakresu *x* tworzone jest pomocnicze drzewo w_{pom} będące drzewem przeszukiwania zakresu *y* dla wszystkich punktów w kanonicznych zbiorach związanych z węzłem *w*



Drzewa obszarów



Drzewa obszarów



Dla d-wymiarowych przeszukiwań obszaru
d poziomów drzew

Drzewa obszarów

Złożoność pamięciowa

- Pierwsze drzewo zakresu x: $O(n)$
- Suma drzew drugiego poziomu: $O(n \log n)$
  - liczba elementów w drzewie jest proporcjonalna do liczby liści, czyli liczby punktów w drzewie
  - każdy punkt należy do pomocniczych drzew dla wszystkich swoich przodków w drzewie
  - drzewo jest zrównoważone – każdy punkt (liść) ma $O(\log n)$ przodków, stąd końcowe oszacowanie

$O(n \log n)$ na płaszczyźnie

$O(n \log^{(d-1)} n)$ w przestrzeni d-wymiarowej

Konstrukcja drzewa

- Stworzenie drzewa wyszukiwania obszaru pierwszego poziomu: $O(n \log n)$
- Tworzenie drzew drugiego poziomu i kolejnych …

- Strukturę można zbudować w czasie

$$O(n \log^{(d-1)} n)$$

Przeszukiwanie

Złożoność czasowa

- wyznaczenie węzłów reprezentujących podzbiory kanoniczne dla zakresu 1-wymiarowego: $O(\log n)$
- $O(\log n)$ kanonicznych podzbiorów, dla każdego kolejne wyszukiwanie $O(\log n)$ – razem $O(\log^2 n)$
  - *wyznaczenie elementów tych zbiorów – dodatkowe k*
  - *wyznaczenie liczby elementów – wstępnie zliczone sumy podzbiorów*
- złożoność czasowa w *przestrzeni d-wymiarowej* :

  $O(\log^d n + k)$

Przeszukiwanie

- Złożoność czasowa
  - dla płaszczyzny (d=2) mieliśmy $O(\log^2 n)$
- Dlaczego?
  - przeszukiwanie drzewa pierwszego poziomu $O(\log n)$
  - dla każdego odwiedzanego węzła, przeszukiwanie drzewa drugiego poziomu $O(\log n)$
- Jak uzyskać $O(\log^{(d-1)} n)$?
  - przeszukiwanie drzew drugiego poziomu w czasie $O(1)$
  - „kaskadowanie cząstkowe” (*fractional cascading*)

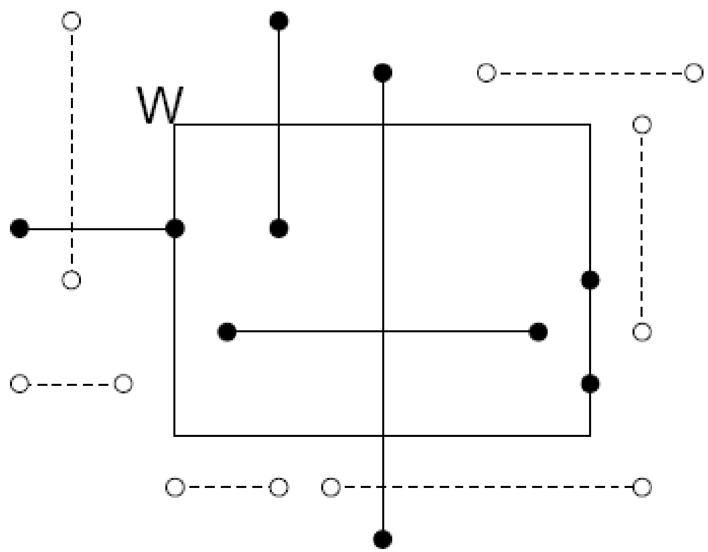
Drzewa przedziałów

- Przeszukiwanie zbiorów obiektów innych niż punkty
- Przykład – odcinki (poziome i pionowe) na płaszczyźnie
  - odcinek jest reprezentowany przez parę wierzchołków
  - odcinki mogą się przecinać

Drzewa przedziałów

Zapytanie o ***okienkowanie***

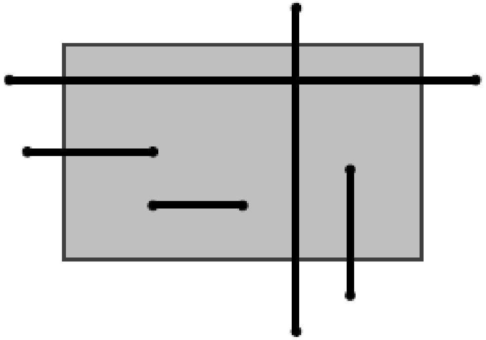
– znaleźć wszystkie odcinki przecinające ortogonalny prostokąt W (także odcinki leżące wewnątrz)



Drzewa przedziałów

Trzy przypadki

1. odcinki, których oba wierzchołki leżą wewnątrz W
2. odcinki, których jeden wierzchołek leży wewnątrz W
3. odcinki, których żaden wierzchołek nie leży wewnątrz W

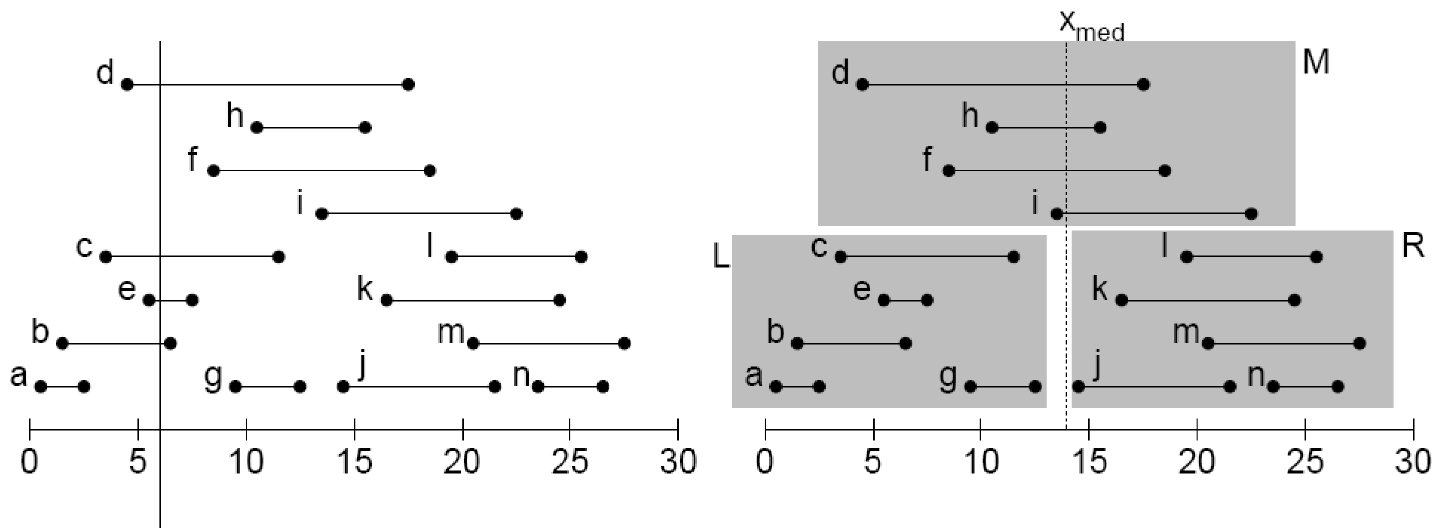


Drzewa przedziałów

- Przypadek 1 i 2 – drzewo przeszukiwania zakresu dla 2*n* punktów
  - odcinki mające oba wierzchołki wewnątrz W będą zgłaszane dwukrotnie
    - posortowanie odcinków wynikowych i usunięcie powtórzeń
    - zaznaczanie odcinków w trakcie wyszukiwania
- Przypadek 3 – odcinki, które przecinają W, ale których wierzchołki nie leżą wewnątrz W
  - poziomy odcinek przecina dowolną pionową linię łączącą górę i dół okna
  - odcinki pionowe analogicznie do poziomych

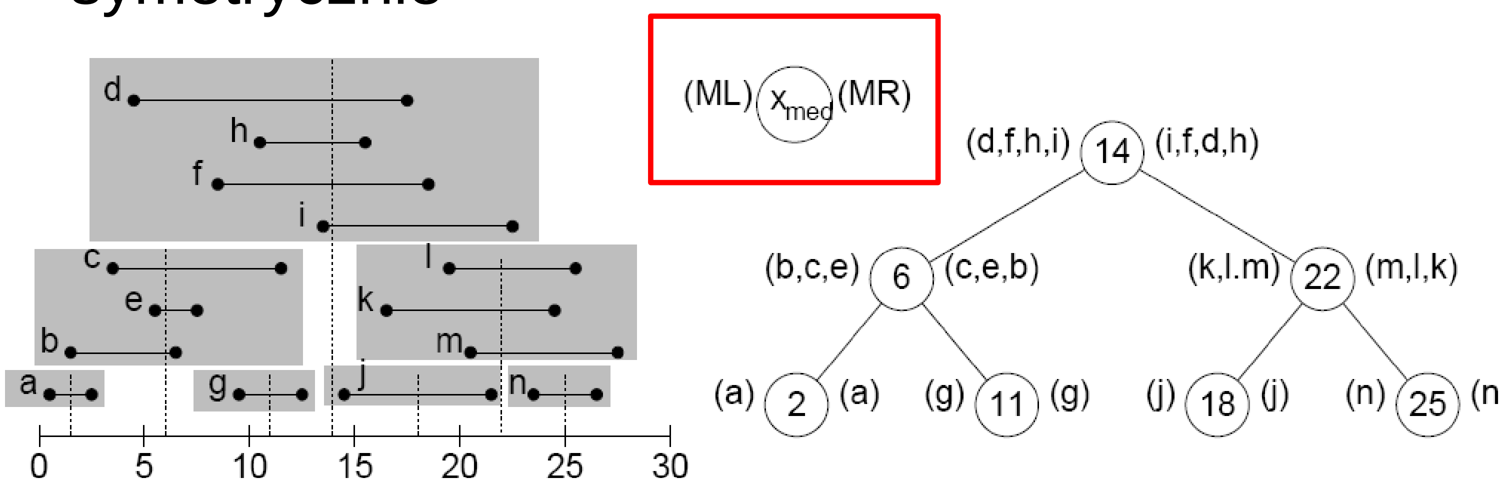
Drzewa przedziałów

- Odszukanie poziomych odcinków przeciętych przez zadaną linię pionową (np. lewy bok okna W)
- Struktura binarnego drzewa podziałów
  - sortujemy wierzchołki (dla współrzędnej x)
  - x_{med} – mediana $2n$ wierzchołków
  - podział na trzy grupy: M, L, R
    - jak przechowywać odcinki w M?



Drzewa przedziałów

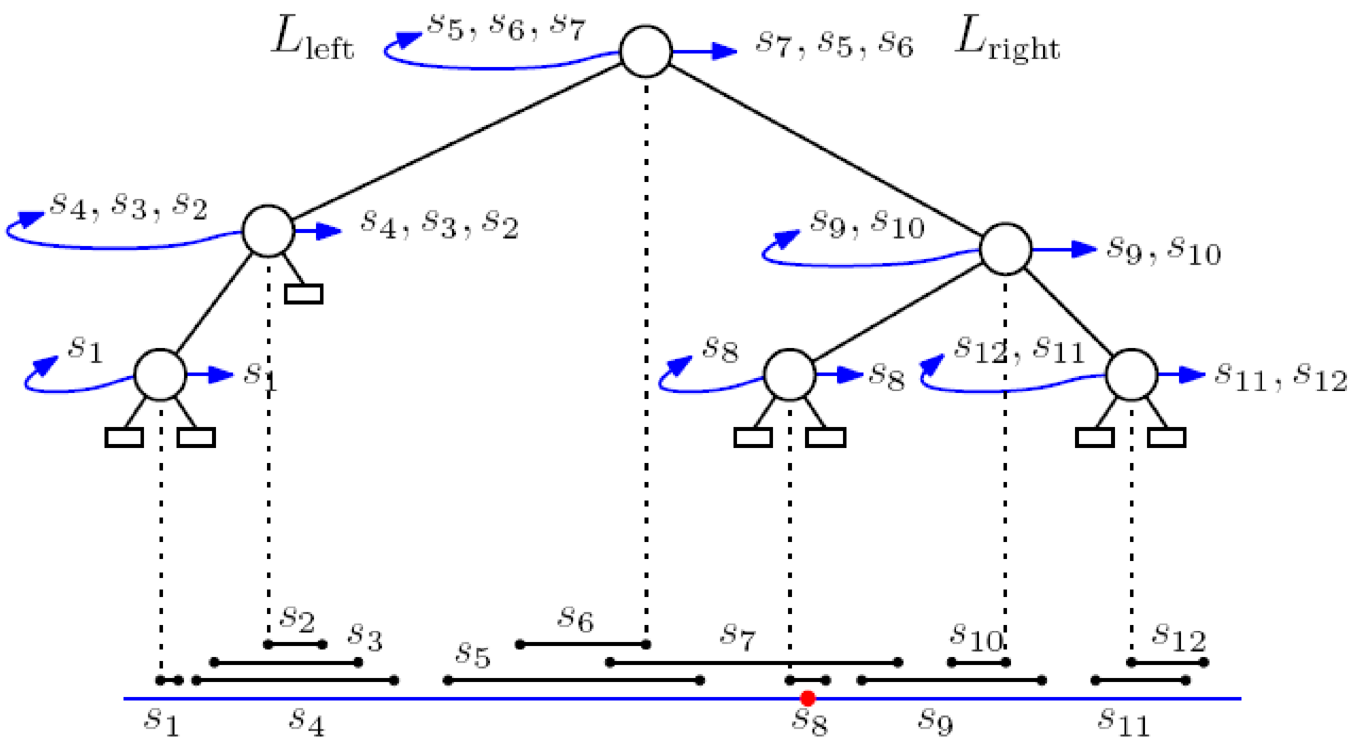
- dla $x_q <= x_{med}$
  - sortujemy odcinki rosnąco według lewego wierzchołka
  - przy przeglądaniu odcinków przerywamy, jeśli lewy wierzchołek jest większy od x_q
- dla $x_q > x_{med}$
  - symetrycznie



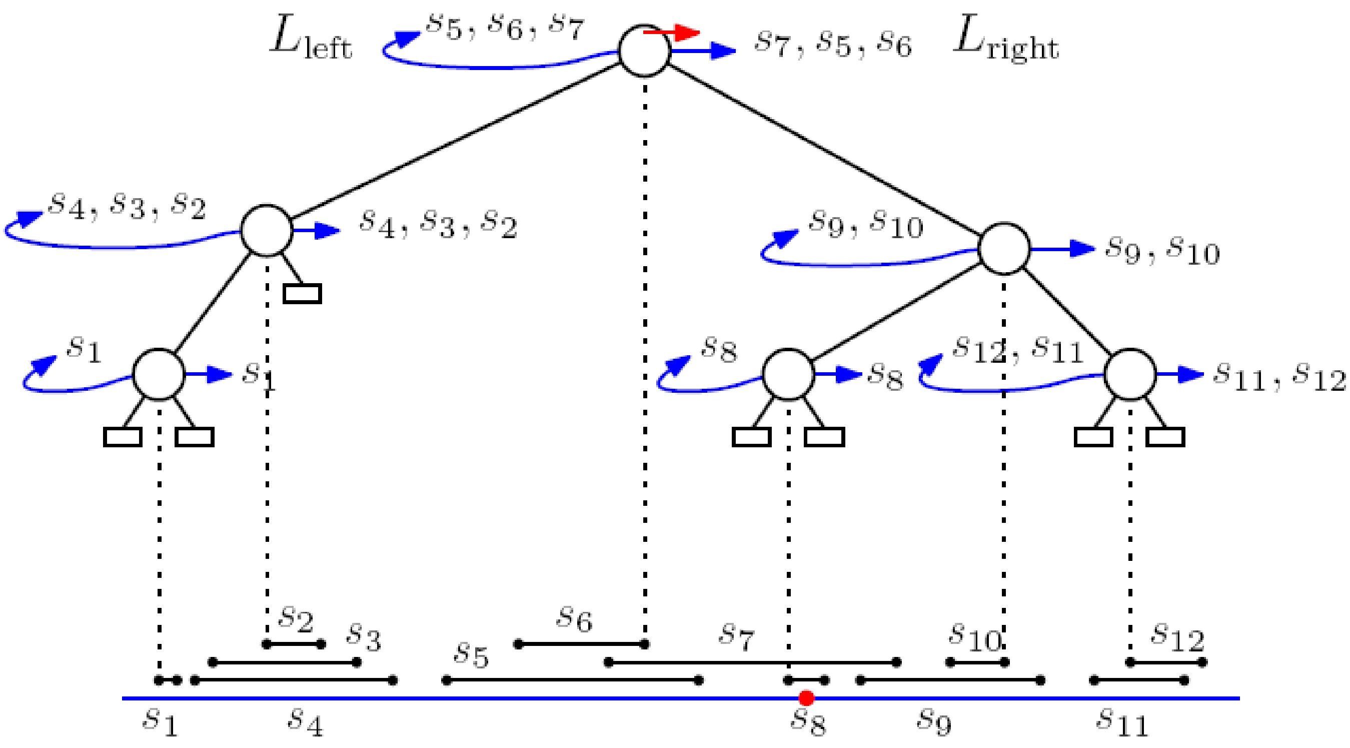
Konstrukcja drzewa

- Rekursywnie, poprzez kolejne podziały zbioru wszystkich odcinków
- Głębokość drzewa $O(\log n)$
  - dla $2n$ wierzchołków, każdy podział dzieli je na dwie grupy L i R o rozmiarze nie większym niż połowa (odliczając odcinki w M)
- Wyliczanie mediany wierzchołków oraz sortowanie odcinków według lewego i prawego wierzchołka
  - wstępne posortowanie tych wartości i zapisanie ich w trzech osobnych listach
- Złożoność czasowa konstrukcji $O(n \log n)$

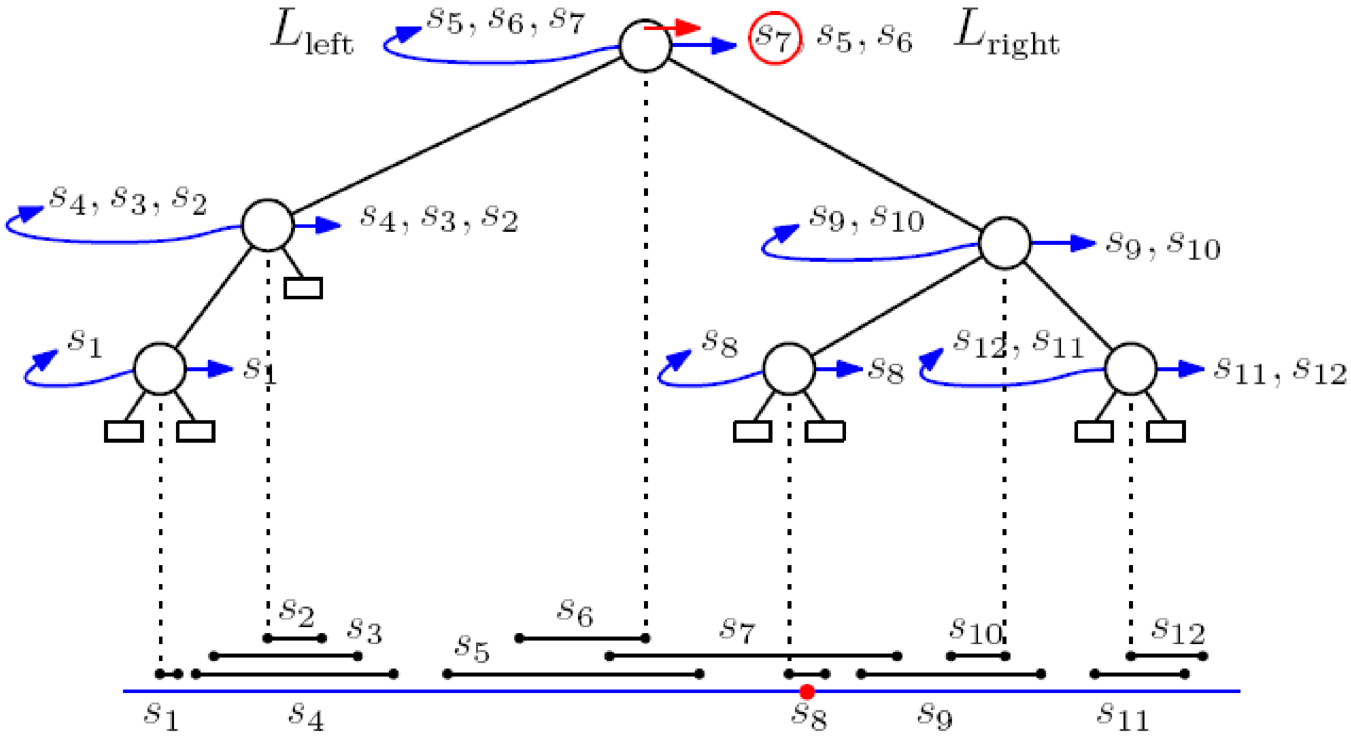
Przeszukiwanie dla prostej



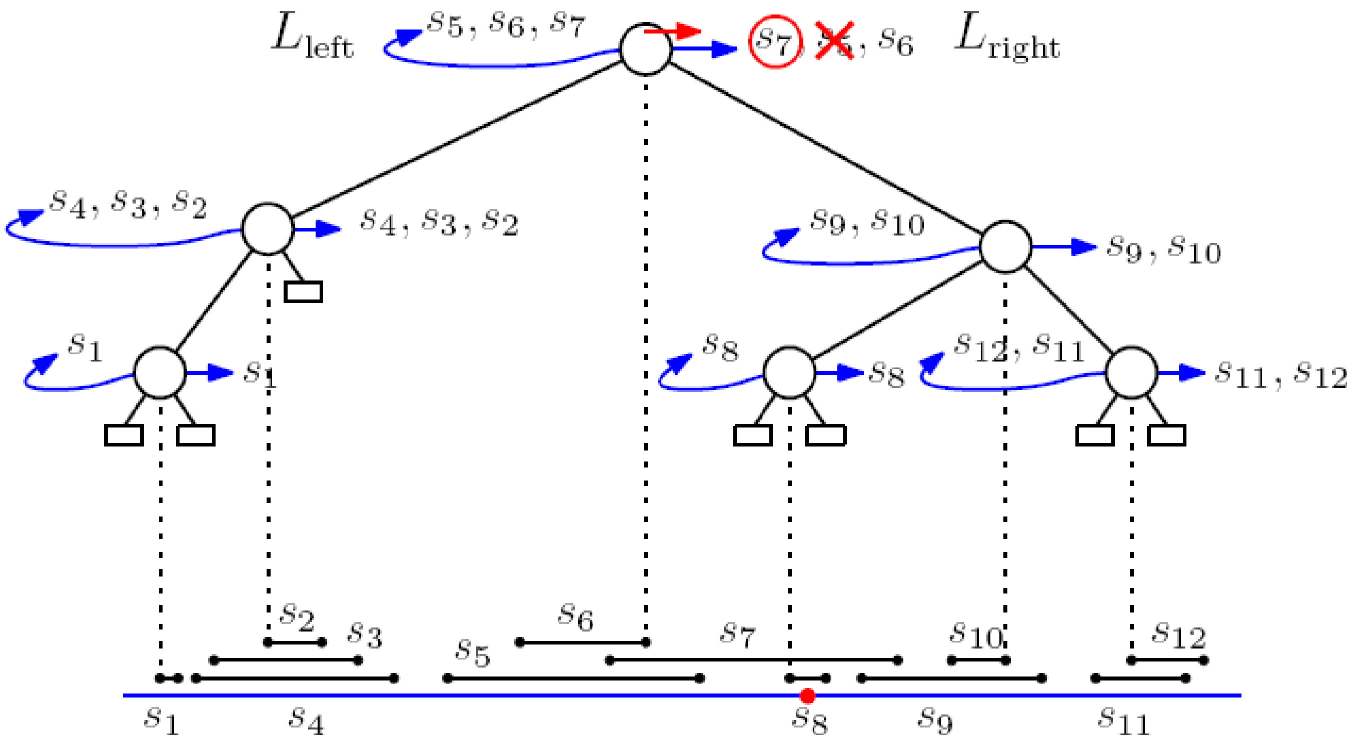
Przeszukiwanie dla prostej



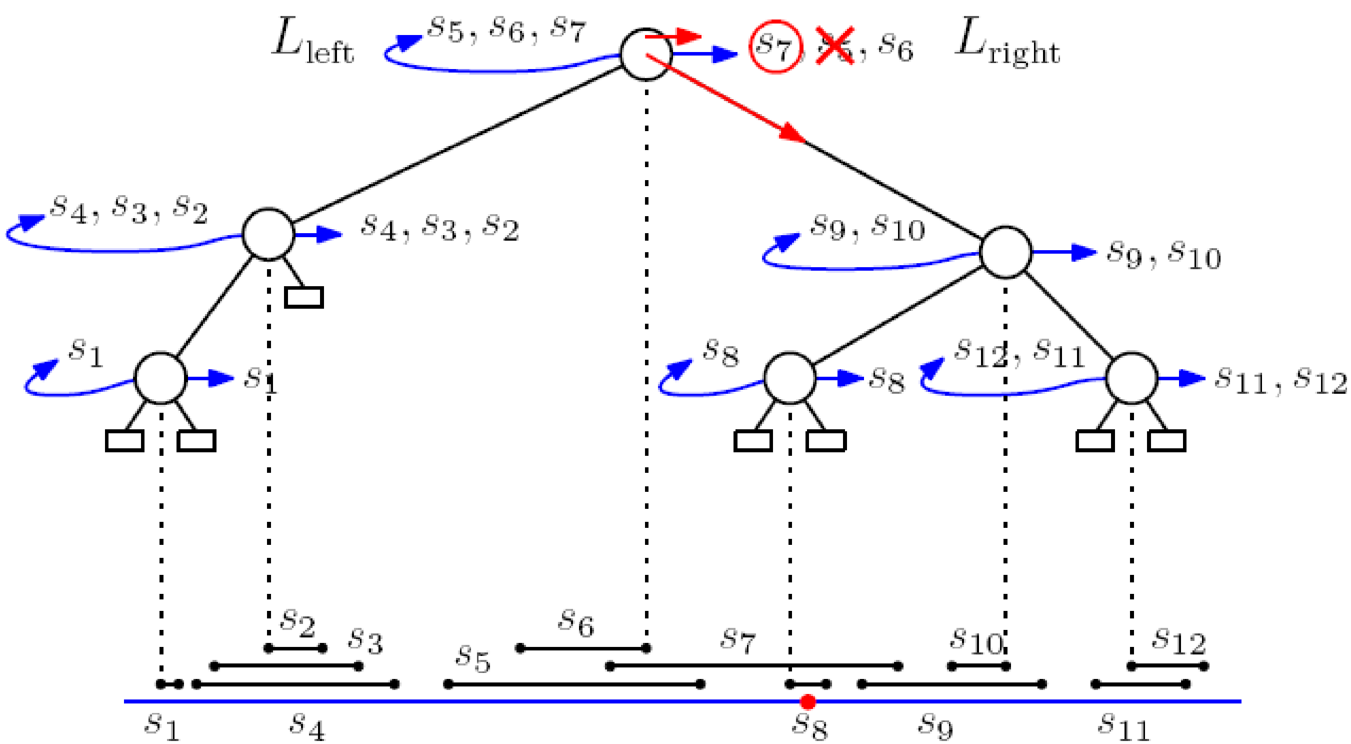
Przeszukiwanie dla prostej



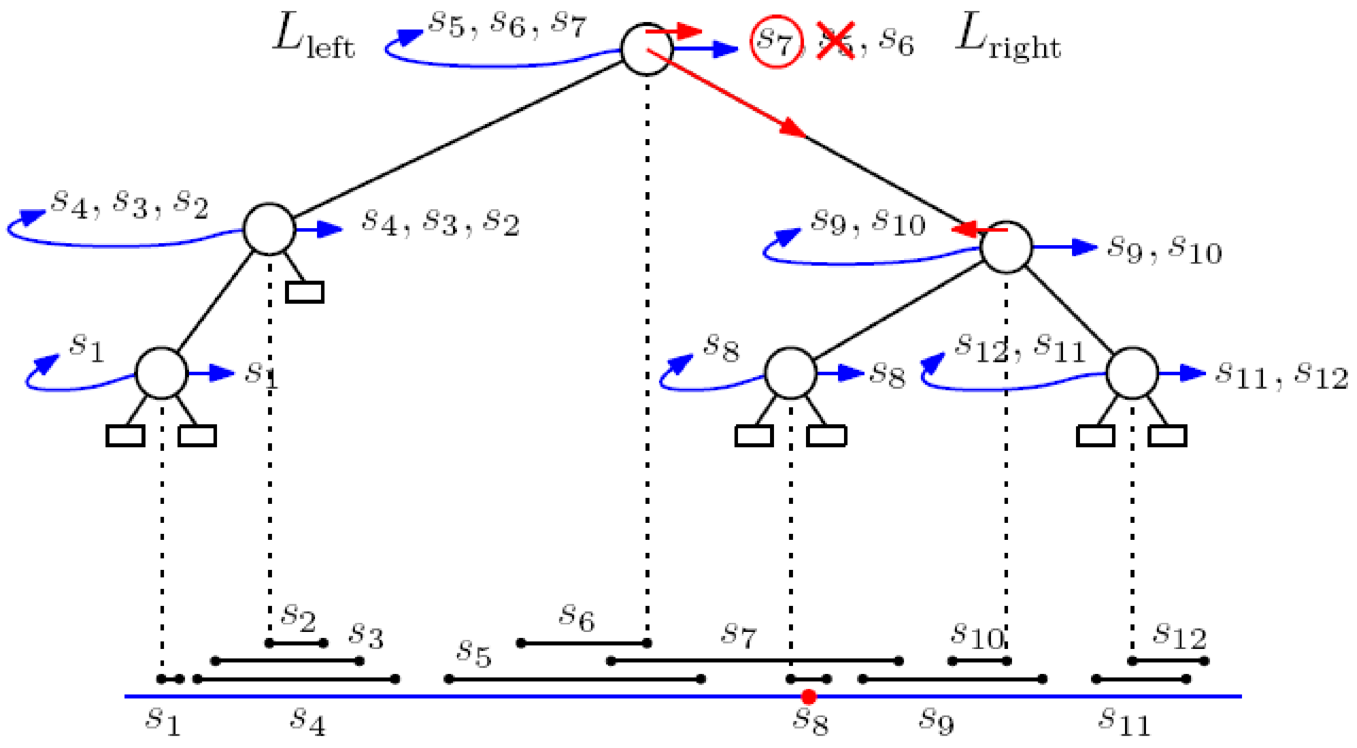
Przeszukiwanie dla prostej



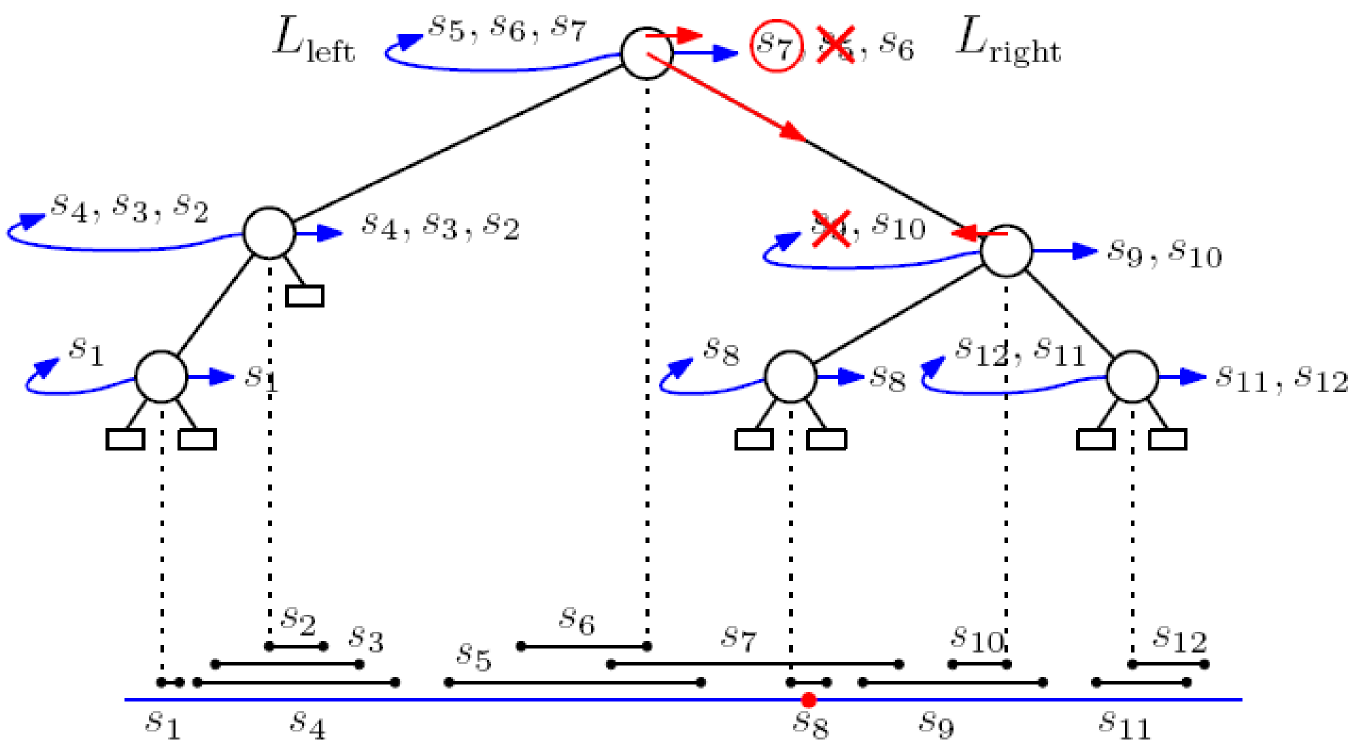
Przeszukiwanie dla prostej



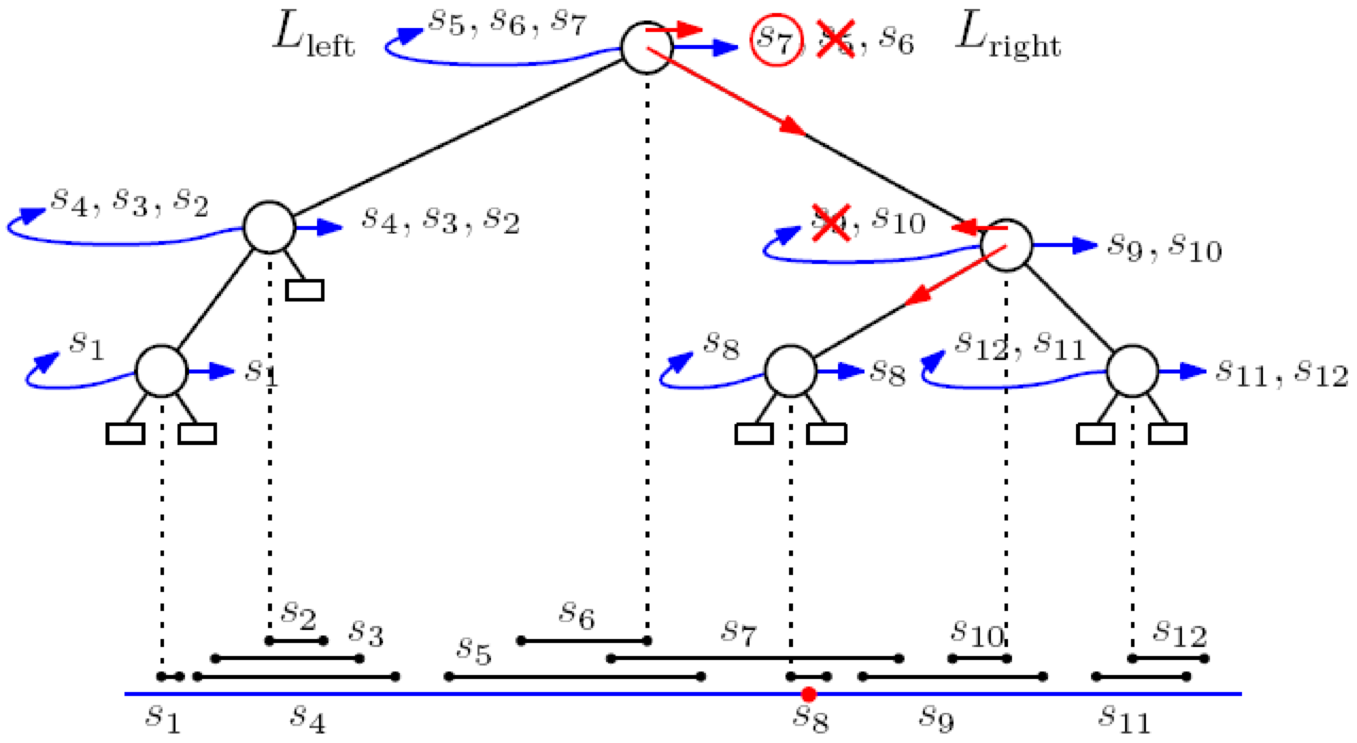
Przeszukiwanie dla prostej



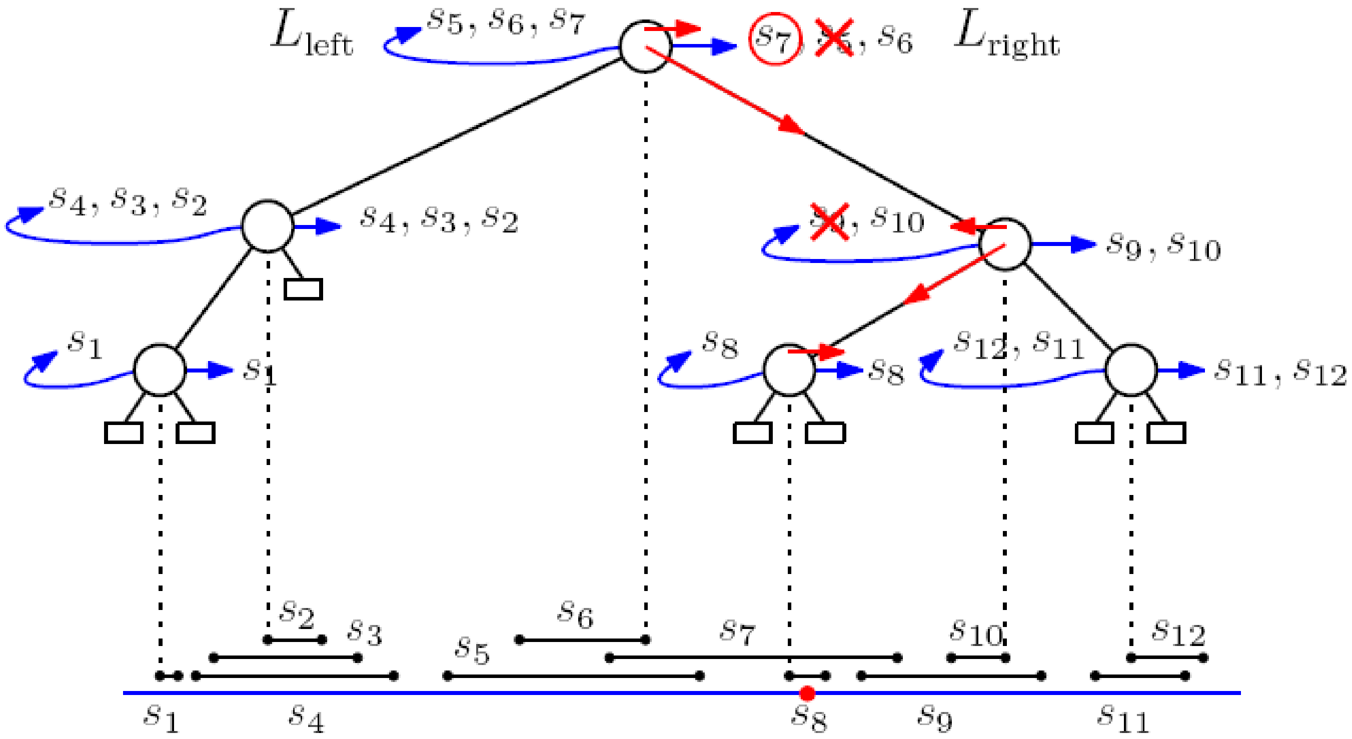
Przeszukiwanie dla prostej



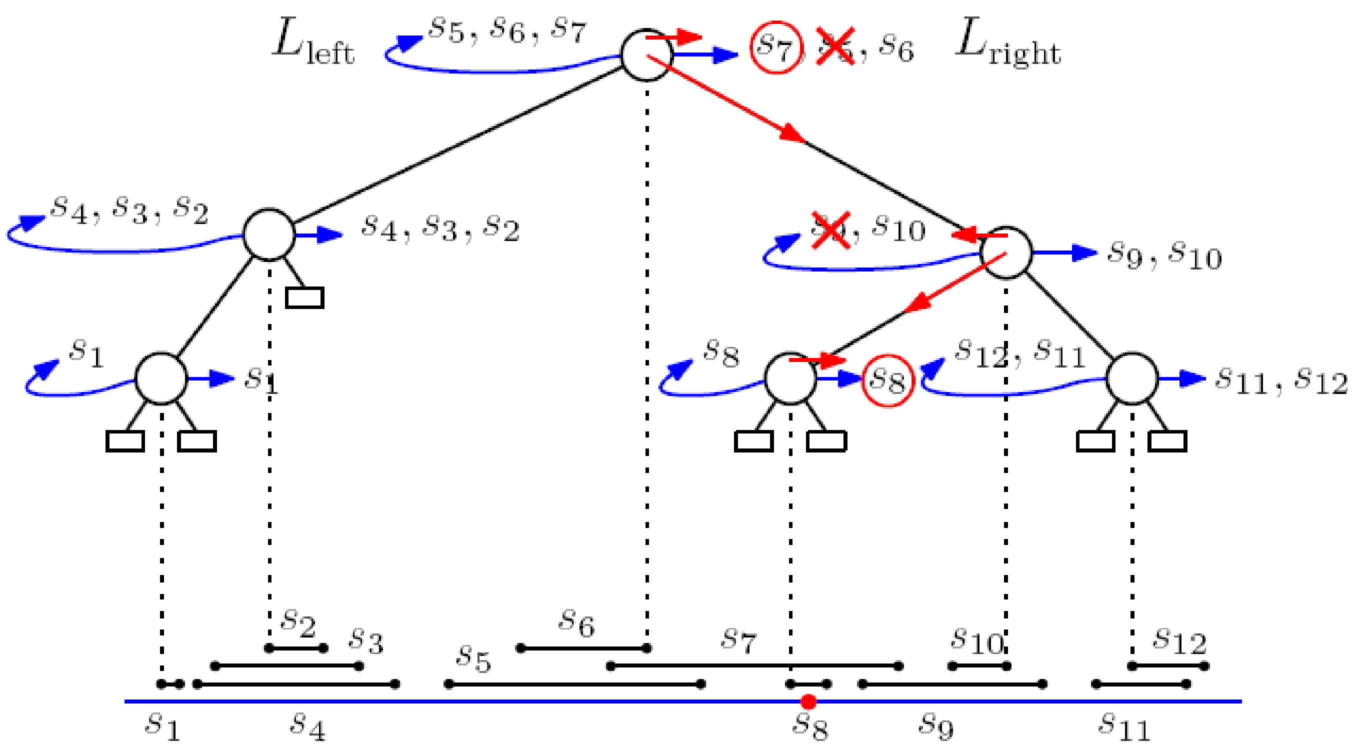
Przeszukiwanie dla prostej



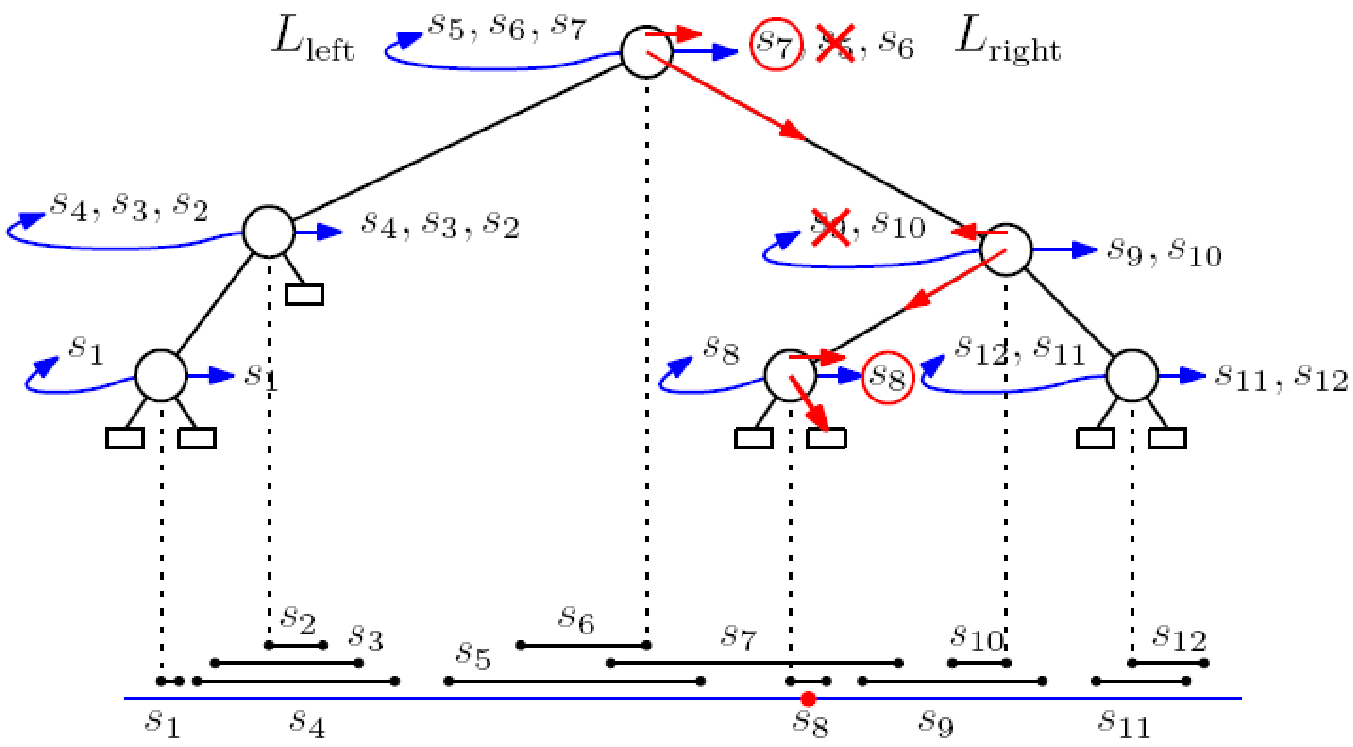
Przeszukiwanie dla prostej

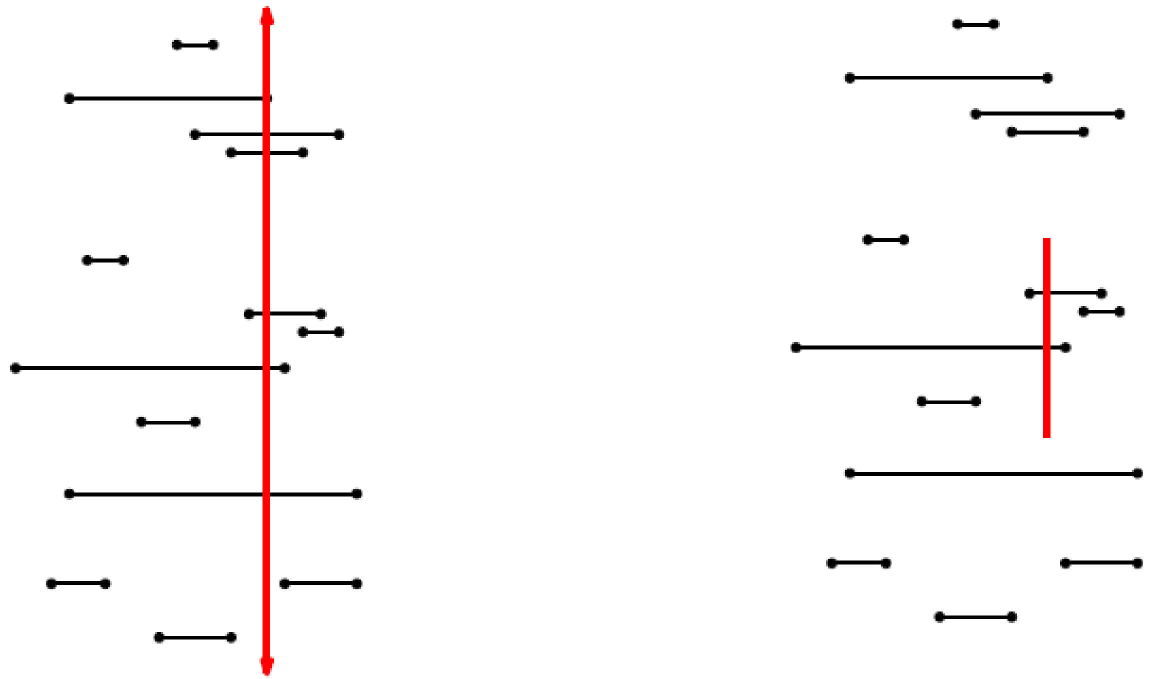


Przeszukiwanie dla prostej



Przeszukiwanie dla prostej

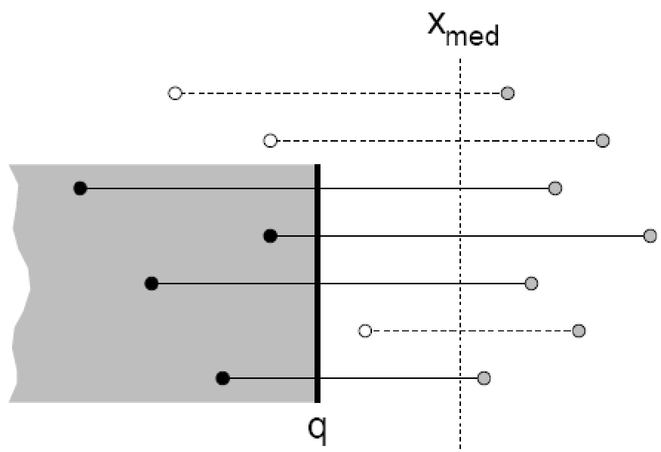




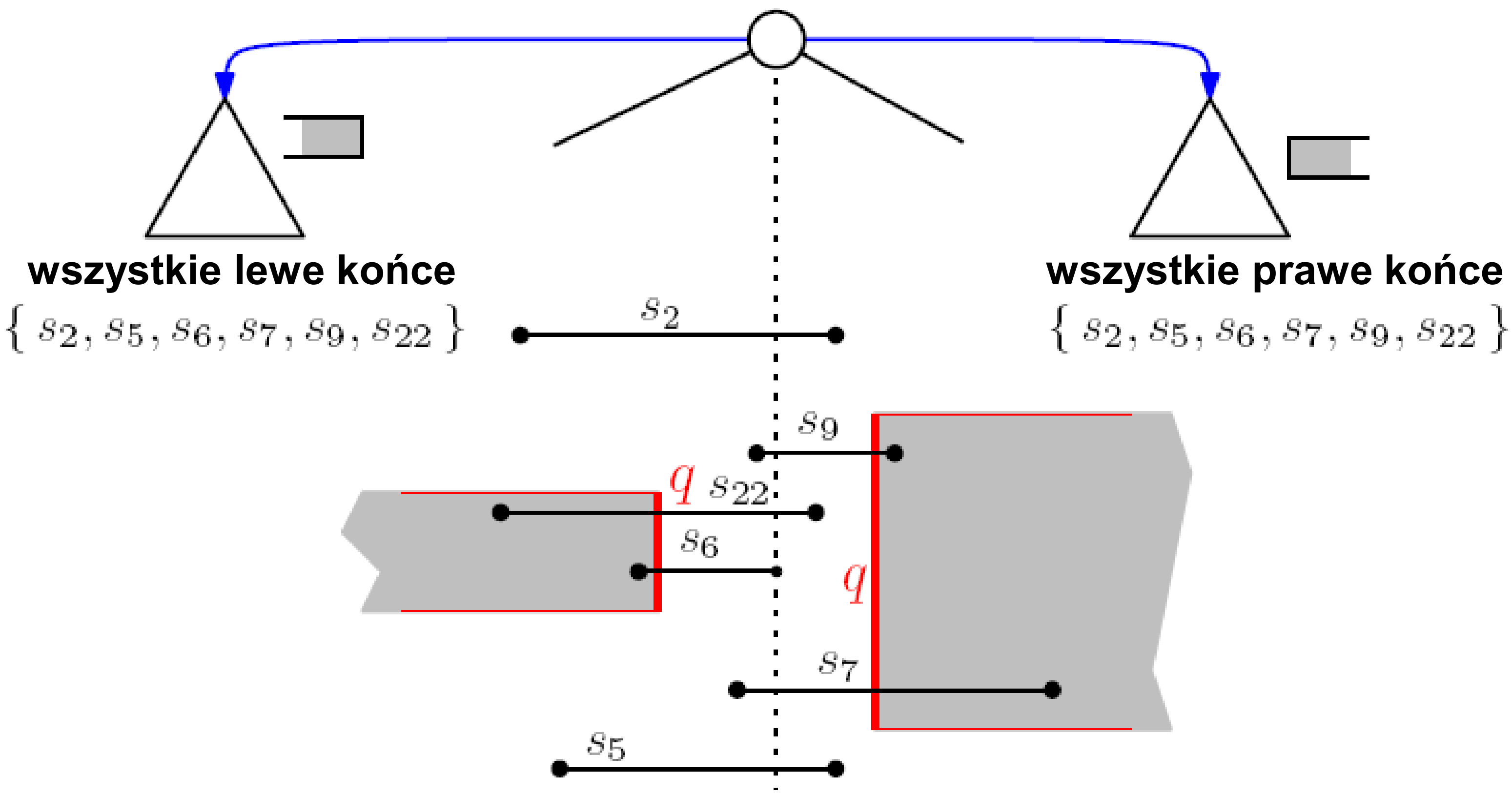
Drzewa przedziałów

Przecięcia odcinków przez pionowy **odcinek**

ML i MR – zamiast list posortowanych według lewych/prawych wierzchołków stosujemy drzewo obszarów dla lewych/prawych wierzchołków



Drzewa przedziałów



Drzewa przedziałów

Efektywność

- złożoność czasowa przeszukiwań
  $$O(\log^2 n + k)$$
- złożoność pamięciowa $O(n \log n)$
- złożoność czasowa konstrukcji $O(n \log n)$