Geometria Obliczeniowa

Przeszukiwanie przestrzeni

Prof. dr hab. inż. **Łukasz Madej**
Mgr inż. **Daniel Bachniak,** Mgr inż. **Mateusz Mojżeszko**
Katedra Informatyki Stosowanej i Modelowania
Wydział Inżynierii Metali i Informatyki Przemysłowej

Budynek B5
p. 716
lmadej@agh.edu.pl
home.agh.edu.pl/lmadej

Plan

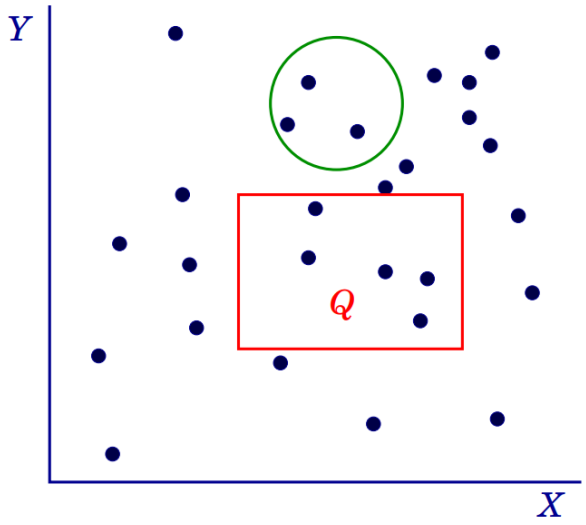
1. Przeszukiwanie przestrzeni,

2. Wyszukiwanie w jedym wymiarze:
   - Drzewo binarne,
   - Drzewo BST,
   - Range Tree:
     - Konstrukcja,
     - Wyszukiwanie,

3. Wyszukiwanie w wielu wymiarach:
   - Drzewa KD dla 2 wymiarów,
   - Drzewa KD dla wielu wymiarów.

Przeszukiwanie przestrzeni

Przeszukiwanie przestrzeni



- Algorytmy oraz **struktury danych** pozwalające na **szybkie przeszukiwanie** wartości (lub zakresów wartości),
- Przeszukiwane mogą być pojedyncze wartości, wielo-wymiarowe punkty, prostokąty i inne typy danych.

Wyszukiwanie liniowe - w najgorszym z możliwych przypadków przy wyszukaniu danego elementu konieczne jest przeglądnięcie wszystkich N dostępnych elementów:

- najprostszy algorytm wyszukiwania informacji w danych zapisanych w strukturze liniowej, np. w postaci listy,
- koncepcja zakłada porównanie każdego z elementu z pożądanym wzorcem,
- dedykowany dla zbiorów nieuporządkowanych.

Znajdź pozycje cyfry 17 w [3,6,2,1,98,4,17] ➡ Pozycja 6

Wyszukiwanie binarne – dla danych uporządkowanych:

- znajdujemy środek przedziału i porównujemy daną wartość z poszukiwanym wzorcem.
- w rezultacie wiadomo w której części leży wzorzec.
- dzielimy dany przedział ponownie na pół itd. aż do znalezienia poszukiwanego elementu.

Znajdź pozycje cyfry 17 w [1,2,3,**4**,6,17,98]

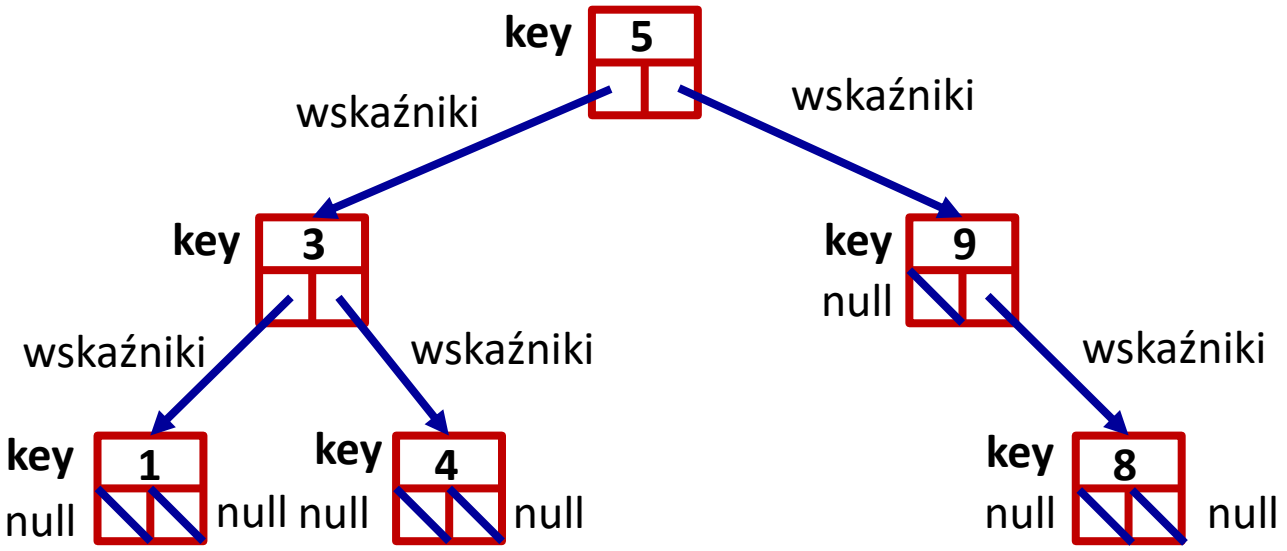
[6,**17**,98]

Wyszukiwanie binarne zazwyczaj jest szybsze niż liniowe – jednak należy uwzględnić narzut związany z posortowaniem listy.

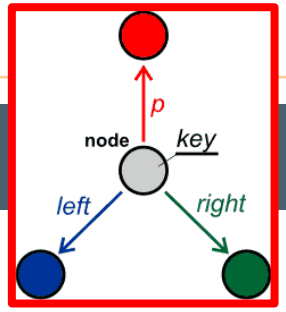
Przeszukiwanie w przestrzeni 1D

Binary Tree (drzewo binarne)

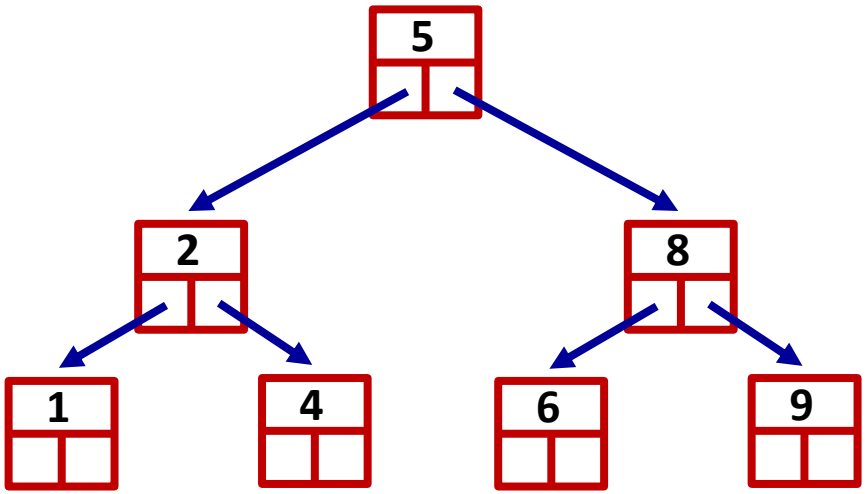
- Drzewo binarne składa się z węzłów,
- Każdy węzeł posiada wskaźniki do lewego (**left**) i prawego (**right**) „syna” oraz wartość (**key** lub data),
- Wskaźnik o wartości null (left, right) oznacza brak „potomstwa” – **Liść drzewa**.



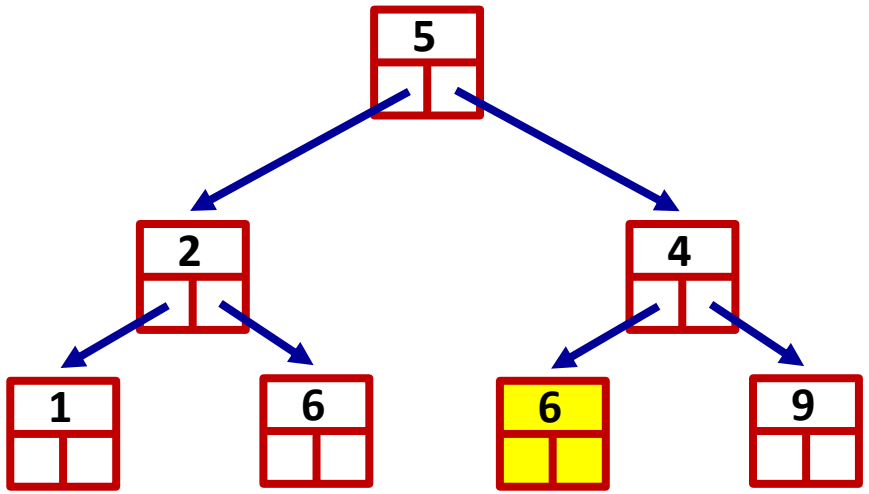
Binary Tree vs Binary Search Tree (BST)



- **Binary Search Tree** (lub Ordered Binary Tree): binarne drzewo poszukiwań,
- Jest typem drzewa binarnego gdzie węzły umieszczone są w określonym porządku. Dla każdego węzła:
  - **wszystkie elementy w lewym poddrzewie są mniejsze lub równe** względem niego,
  - **wszystkie elementy w prawym poddrzewie są większe** względem niego.



drzewo BST

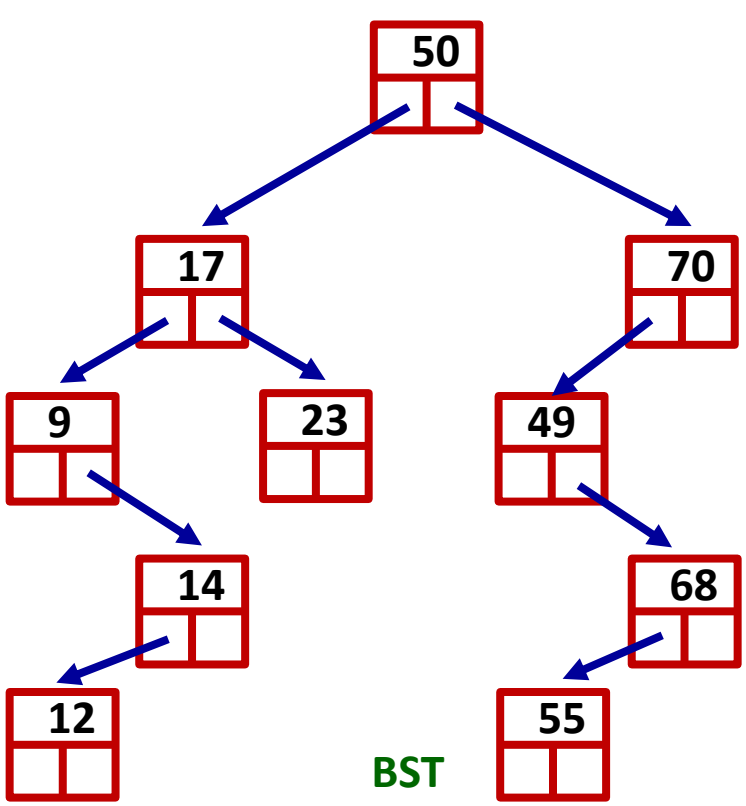


drzewo binarne (ale nie BST)

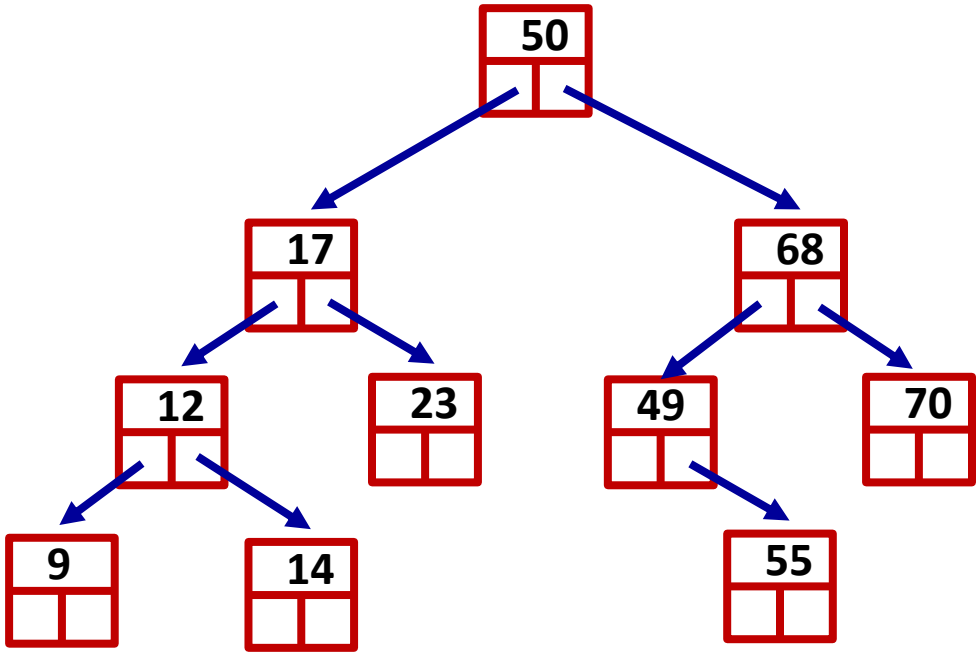
BST vs Range Tree

- **Range Tree** jest **zbalansowanym binarnym drzewem poszukiwań** (**Balanced Binary Search Tree**) dla zbioru jedno-wymiarowych punktów,

- Zbalansowane drzewo: **wysokość drzewa jest najmniejsza** z możliwych dla danego zbioru punktów.



BST



Range Tree

https://en.wikipedia.org

Konstrukcja drzewa Range Tree 1D

algorytm Top down - opis

Budowa drzewa:

- Posortuj punkty rosnąco (przed konstrukcją drzewa), następnie wywołaj metodę budowy drzewa rekurencyjnie,

- Jeśli metoda budowy drzewa otrzymała jako argument tylko jeden punkt to zwróć ten punkt (jako węzeł),

- Jeśli metoda budowy drzewa otrzymała więcej niż jeden punkt to:
  - Wyznacz **medianę** dla tych punktów,
  - Podziel punkty na dwa zbiory względem mediany,
  - Zwróć węzeł drzewa (punkt, którego kluczem jest mediana)
  - Lewe i prawe dziecko tego węzła są wyznaczone rekurencyjnie na podstawie dwóch wyznaczonych zbiorów.

Mediana - wartość cechy w szeregu uporządkowanym, powyżej i poniżej której znajduje się jednakowa liczba obserwacji.

Mediana ze zbioru liczb:

1. posortować punkty w kolejności od najmniejszej do największej
2. nadać numerację od 1 do n.
3. podzielić:
    1. jeżeli n jest nieparzyste, medianą jest wartość obserwacji w środku: (n+1)/2
    2. jeżeli n jest parzyste, wynikiem jest średnia arytmetyczna między dwiema środkowymi obserwacjami: n/2 i (n+1)/2

[3,6,2,1,98,4,17]

[1,2,3,4,6,17,98]

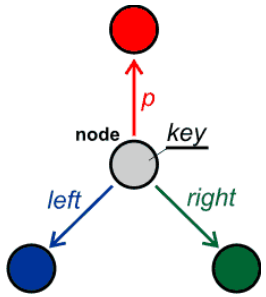
[1,2,3,**4**,6,17,98]

https://pl.wikipedia.org/wiki/Mediana

Konstrukcja drzewa Range Tree 1D

Algorytm – implementacja (C#)

```
public class Node1D {
        public double Key;

        public Node1D Left;
        public Node1D Right;
        public Node1D p;

}
```

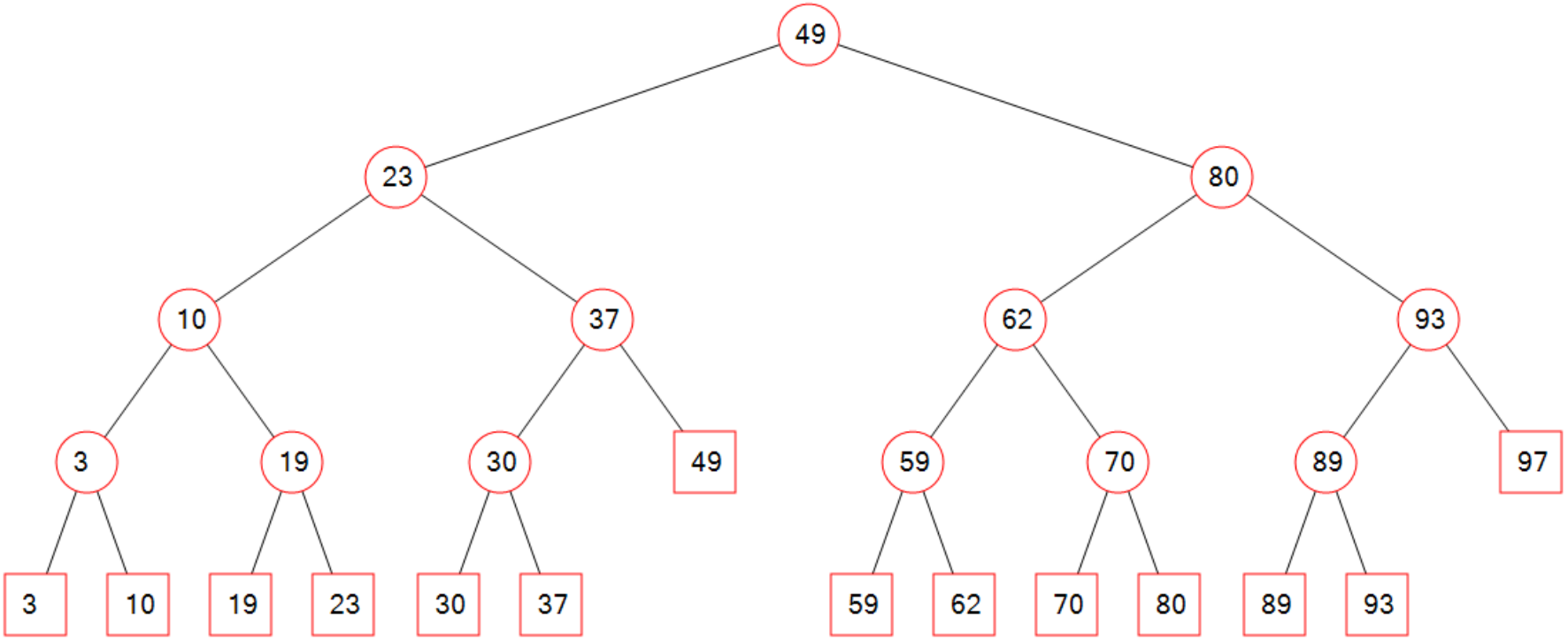



```
Node1D BuildRangeTree1D_TopDown(List<double> points, int begin, int end) {
        if (begin == end)
                return new Node1D() { Key = points[begin] };
        else {
                int center = (begin + end) / 2;
                Node1D node = new Node1D();
                node.Key = points[center];
                node.Left = BuildRangeTree1D_TopDown(points, begin, center);
                node.Right = BuildRangeTree1D_TopDown(points, center + 1,
                end);

                return node;
        }
}
```

Konstrukcja drzewa Range Tree 1D

algorytm– wynik działania

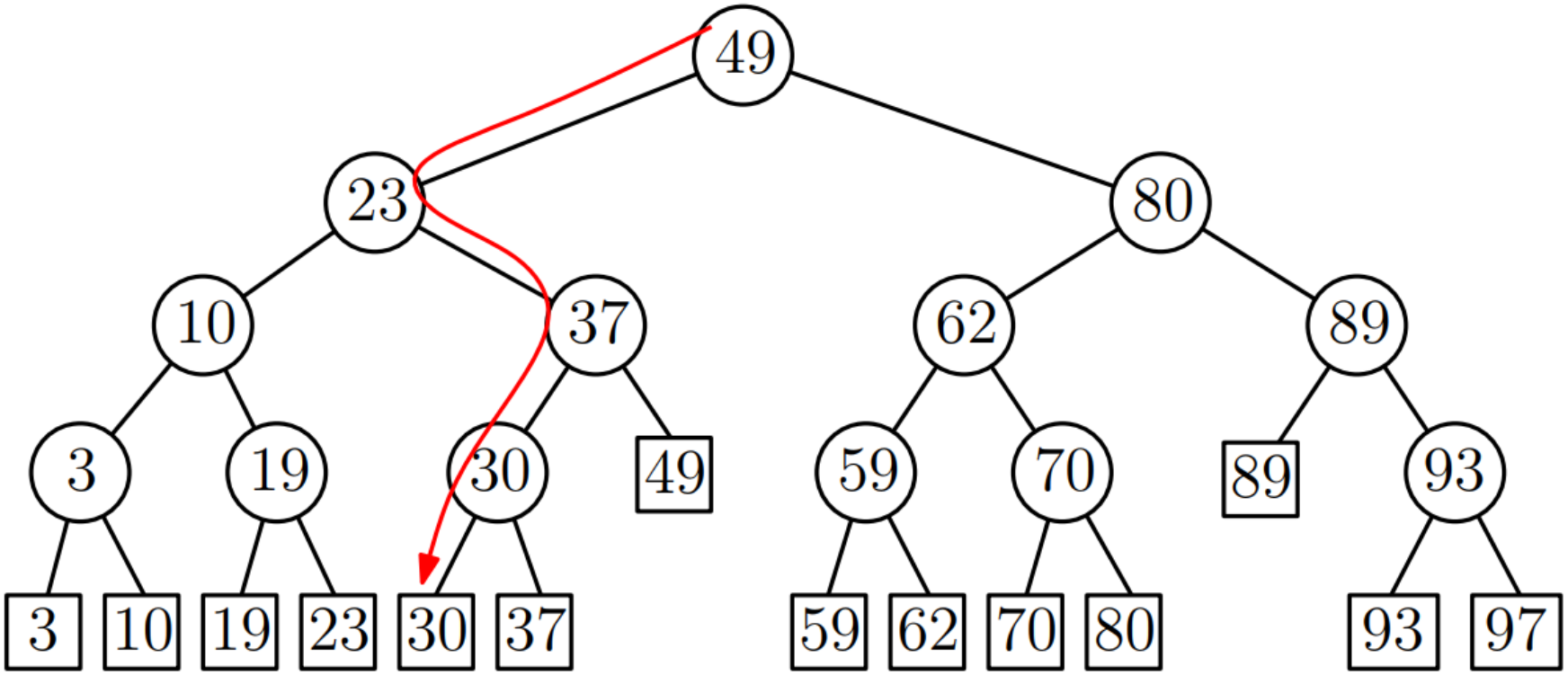
```
List<double> points = new List<double>()
{ 3, 10, 19, 23, 30, 37, 49, 59, 62, 70, 80, 89, 93, 97 };

points.Sort();
Node1D tree = Node1D.BuildRangeTree1D_TopDown(points, 0, points.Count - 1);
```



Przeszukiwanie zakresu

Przykład 1 – zakres <25, 90>

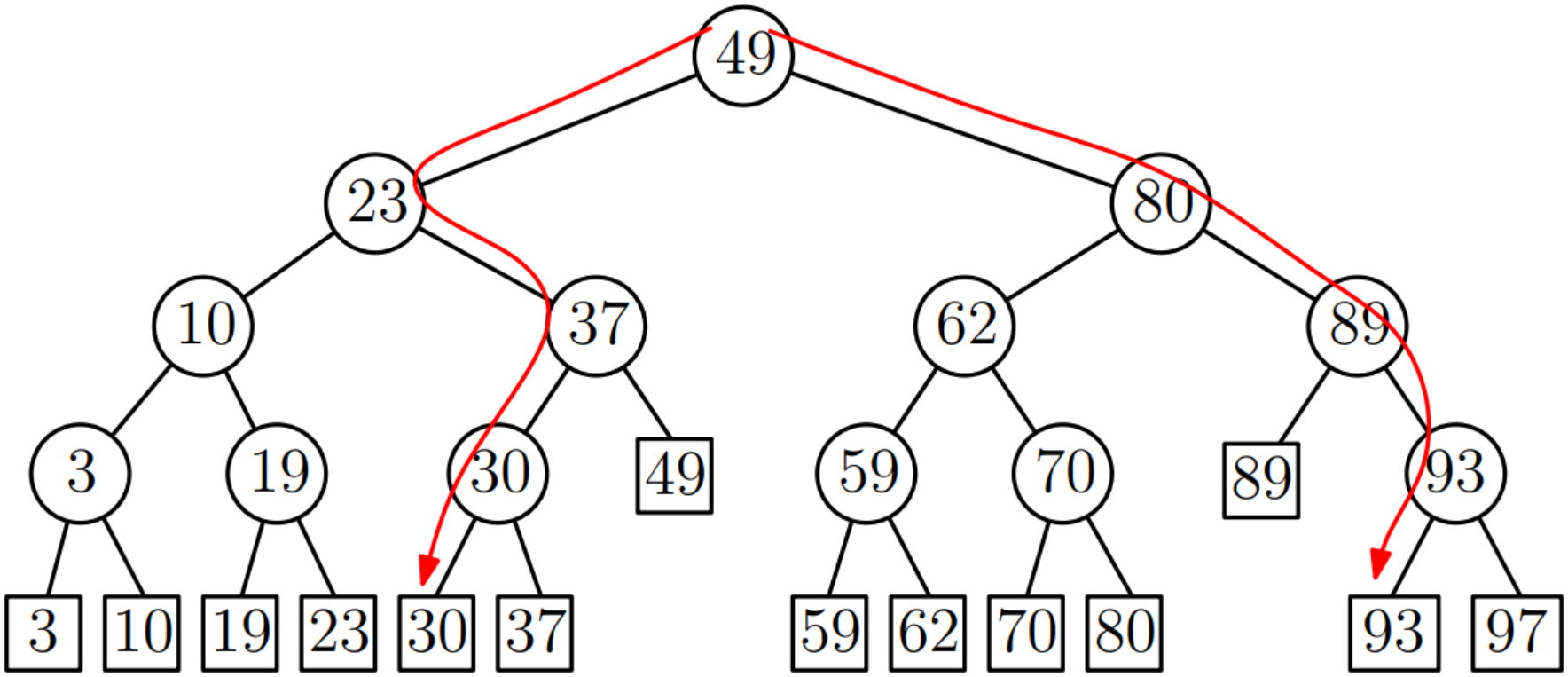
Ścieżka poszukiwania dla 25



Przeszukiwanie zakresu

Przykład 1 – zakres <25, 90>

Ścieżka poszukiwania dla 25 i 90



Range Tree 1D

Poszukiwanie wartości (jeśli nie ma to najbliższej większej wartości)

```
Node1D FindValue(Node1D tree, double val)
{
    if (tree == null) return null;

    List<Node1D> printNodes = new List<Node1D>();

    Node1D tmp = tree;
    while (true)
    {
        if(tmp.Left == null && tmp.Right == null)
            return tmp;

        if (tmp.Key >= val)          // go to left
        {
            tmp = tmp.Left;
        }

        else if (tmp.Key < val)      // go to right
        {
            tmp = tmp.Right;
        }
    }
}
```

Przeszukiwanie Zakresu

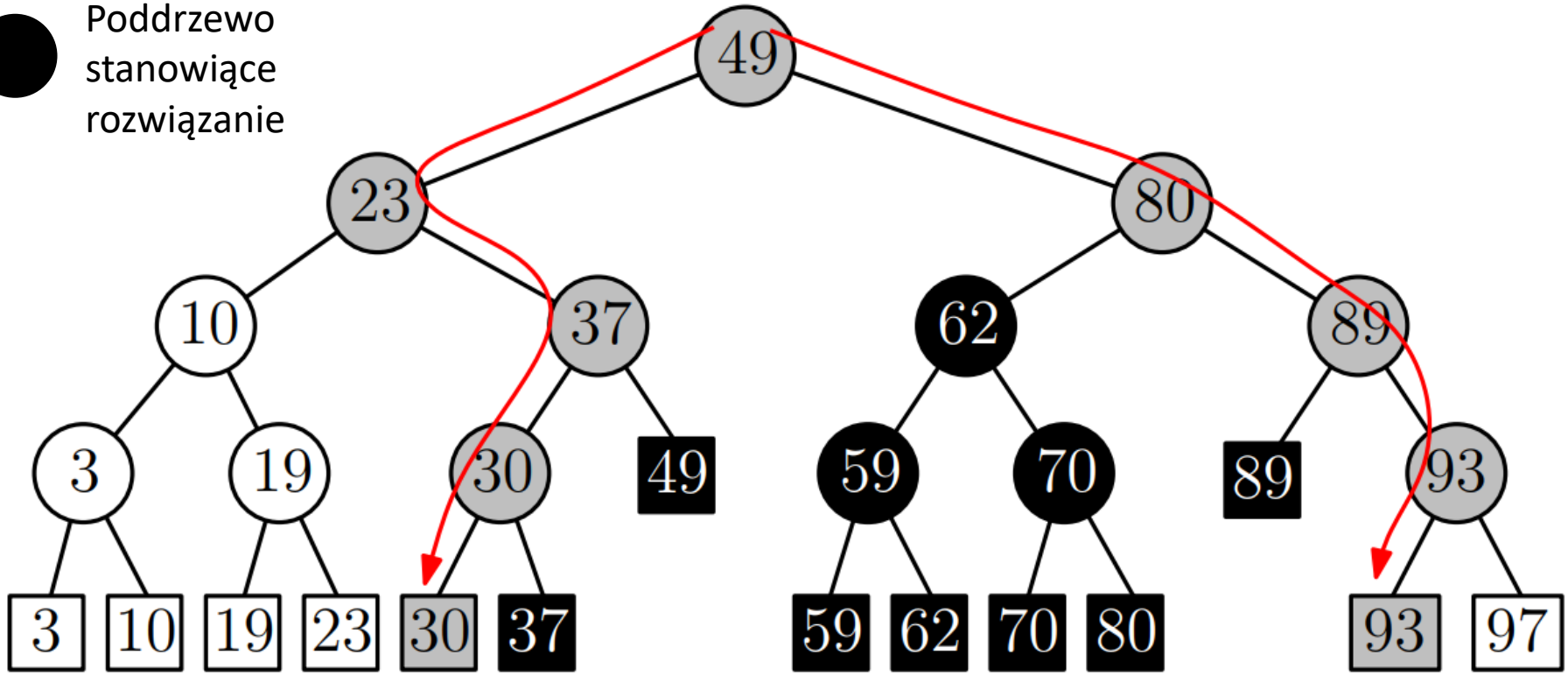
Przykład 1 – zakres <25, 90>

Nie sprawdzane

Sprawdzane **może** być elementem rozwiązania

Poddrzewo stanowiące rozwiązanie

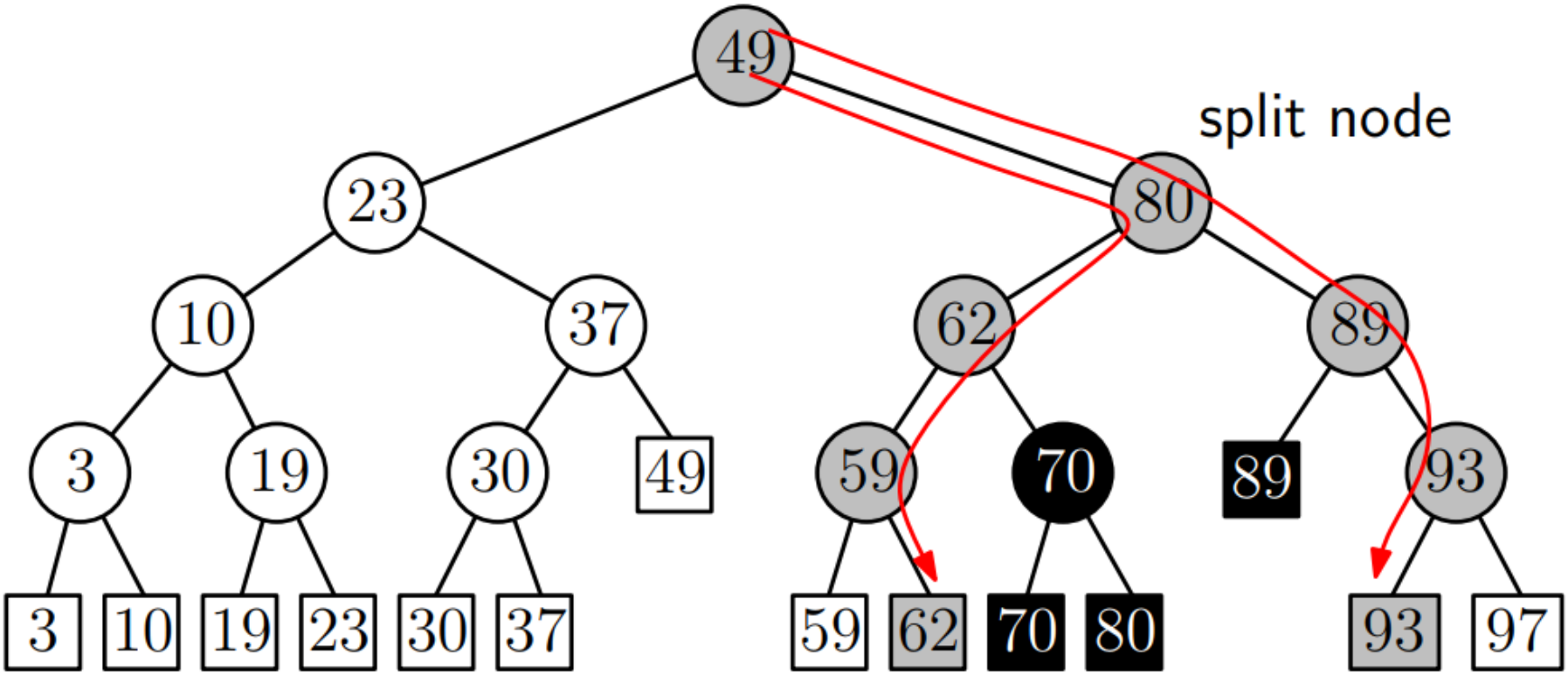
Ścieżka poszukiwania dla 25 i 90 oraz zaznaczone typy węzłów



Przeszukiwanie Zakresu

Przykład 2 – zakres <61, 90>

Ścieżka poszukiwania dla 61 i 90



Range Tree 1D

Poszukiwanie wartości - zaznaczanie poddrzew należących do zakresu (1)

Wywołujemy tą metodę 2 razy: val = min oraz val = max

```
List<Node1D> GetSubTreesInRange(Node1D tree, double val, double min, double max)
{
    List<Node1D> nodesInRange = new List<Node1D>();

    if (tree == null) return null;

    Node1D node = tree, prevRight = null, prevLeft = null;
    while (true)
    {
        if (node.Left == null && node.Right == null)    // if leaf -> finish
        { ... }

        if (node.Key >= val)                            // go to left
        { ... }

        else if (node.Key < val)                        // go to right
        { ... }
    }

    return nodesInRange;
}
```

1

2

3

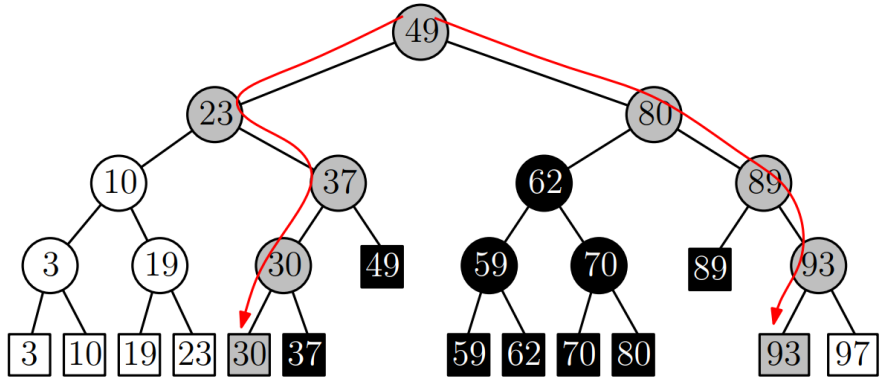
Range Tree 1D

Poszukiwanie wartości - zaznaczanie poddrzew należących do zakresu (2)

Jeśli węzeł jest liściem, sprawdzamy czy należy do zakresu (min, max)

```
List<Node1D> GetSubTreesInRange(Node1D tree, double val, double min, double max)
{
    ...

        if (node.Left == null && node.Right == null)     // if leaf -> finish
        {
            if(node.Key >= min && node.Key <= max)
                nodesInRange.Add(node);

            break;
        }

    ...

    }

    return nodesInRange;
}
```

1



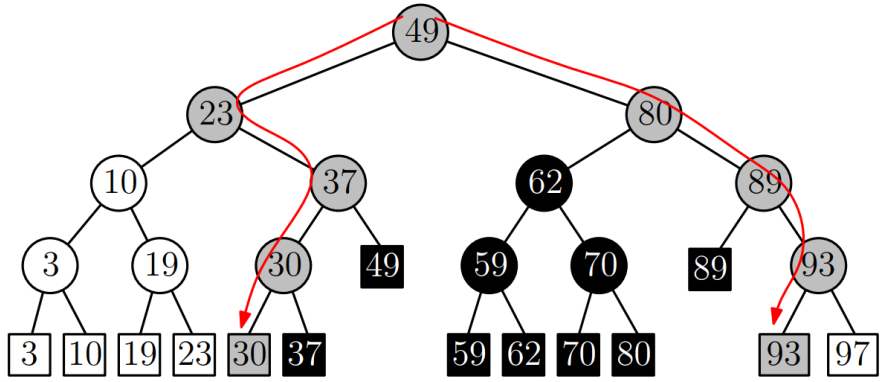
Range Tree 1D

Poszukiwanie wartości - zaznaczanie poddrzew należących do zakresu (3)

- Jeśli będziemy przechodzić w lewo to sprawdzamy czy cała prawa gałąź należy do zakresu
- Dopieo wtedy przechodzimy w lewo.

```
List<Node1D> GetSubTreesInRange(Node1D tree, double val, double min, double max)
{
    ...

        if (node.Key >= val)                                    // go to left
        {
            if (node.Key >= min && prevRight != null && prevRight.Key <= max)
                nodesInRange.Add(node.Right);

            prevRight = node;
            node = node.Left;
        }
    ...

    }

    return nodesInRange;
}
```

2



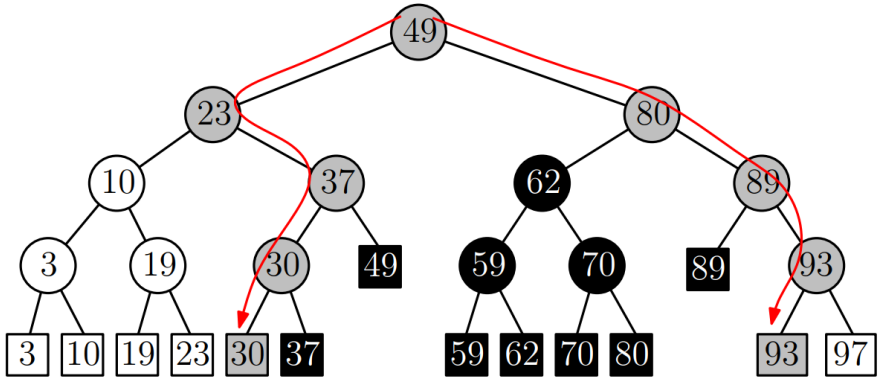
Range Tree 1D

Poszukiwanie wartości - zaznaczanie poddrzew należących do zakresu (4)

- Jeśli będziemy przechodzić w prawo to sprawdzamy czy cała lewa gałąź należy do zakresu
- Dopieo wtedy przechodzimy w prawo.

```
List<Node1D> GetSubTreesInRange(Node1D tree, double val, double min, double max)
{
    ...

        else if (node.Key < val)                          // go to right
        {
            if (node.Key <= max && prevLeft != null && prevLeft.Key >= min)
                nodesInRange.Add(node.Left);

            prevLeft = node;
            node = node.Right;
        }
    ...

    }

    return nodesInRange;
}
```

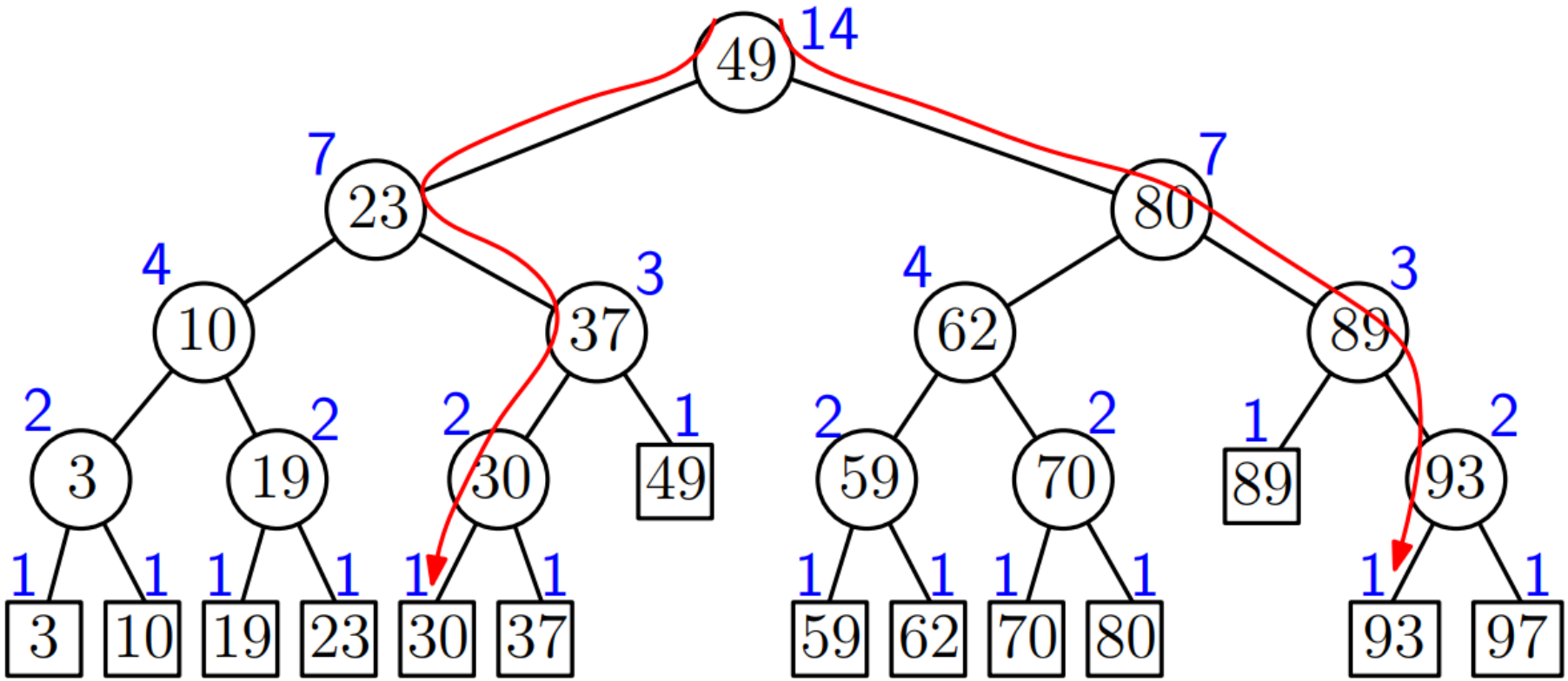
3



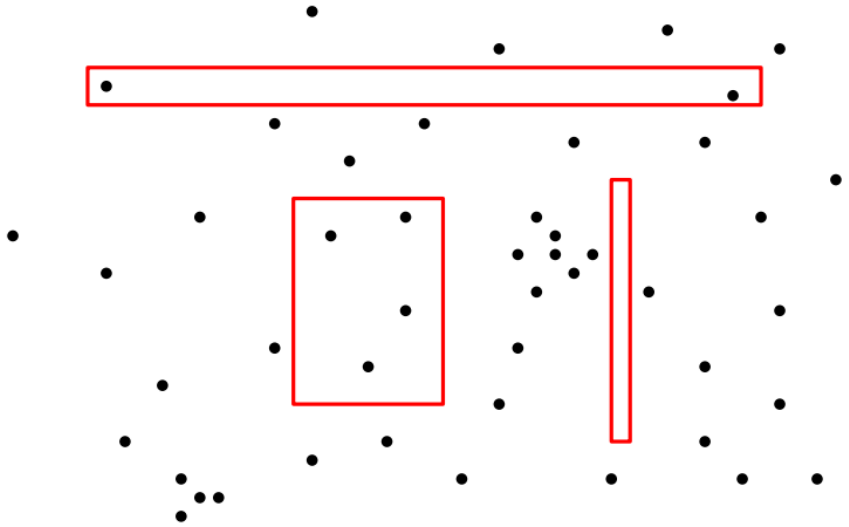
Przeszukiwanie Zakresu

Lizba odwiedzin węzłów

Liczba możliwych odwiedzin węzłów



Przeszukiwanie w przestrzeni wielowymiarowej



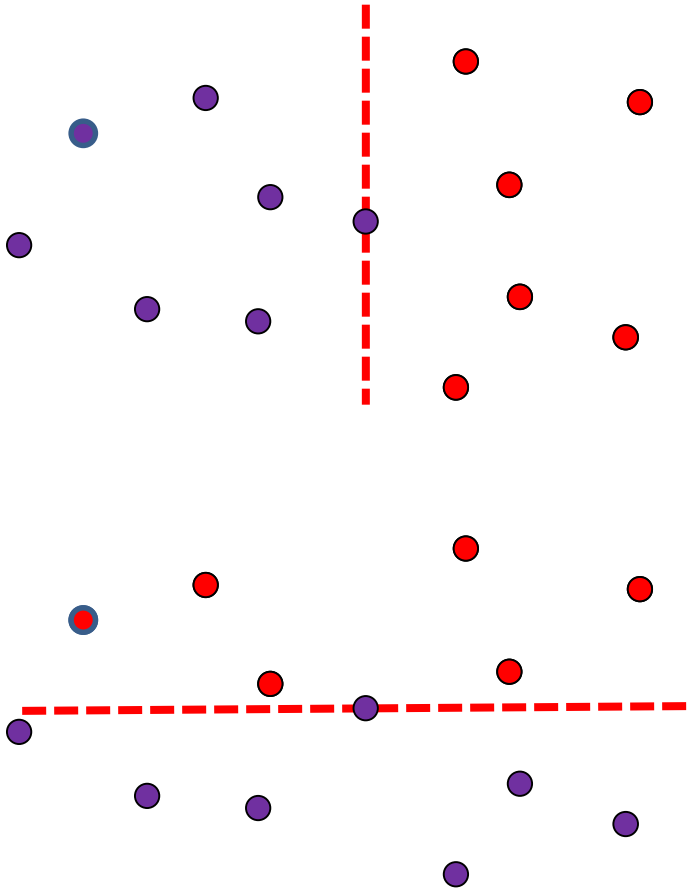
Drzewa KD – 2D

Idea, implementacja

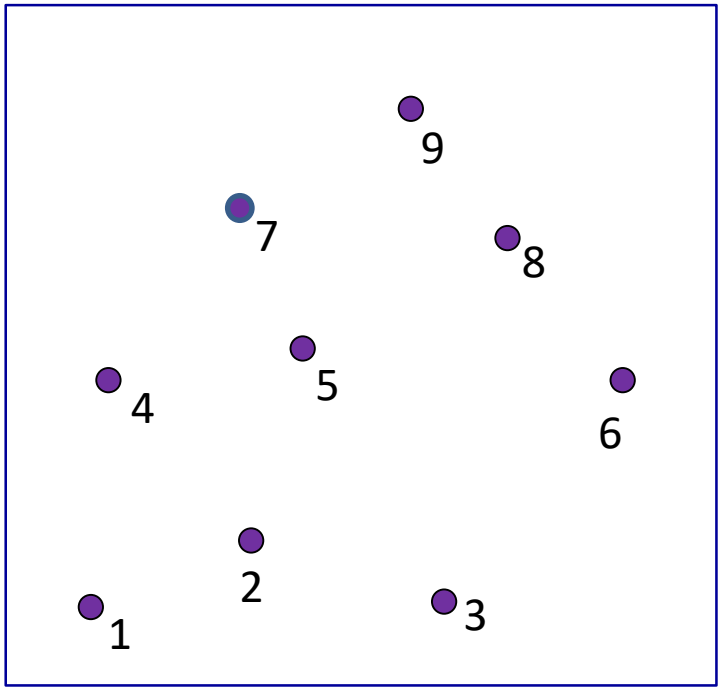
Podział zbioru punktów naprzemiennie: względem współrzędnej **x** i względem współrzędnej **y**.

Podział punktów względem współrzędnej x: pionowa linia dzieli punkty, **połowa punktów znajduje się po lewej** stronie linii (lub na lini), **połowa po prawej** stronie,

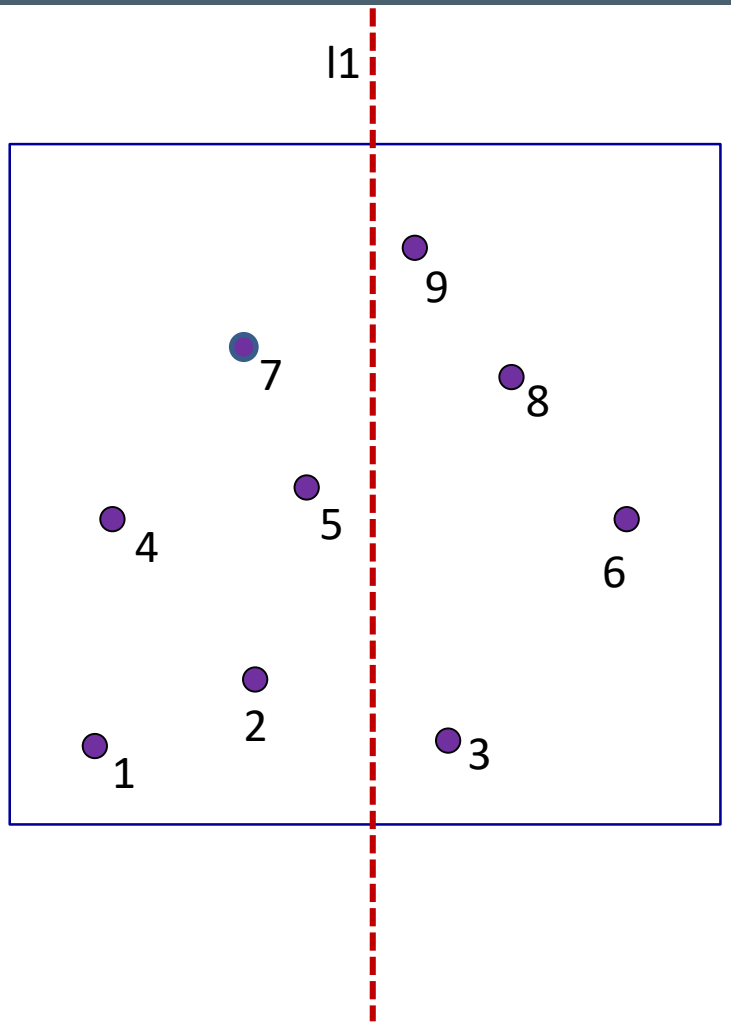
Podział punktów względem współrzędnej y: pozioma linia dzieli punkty, **połowa punktów znajduje się powyżej linii** (lub na linii), **połowa poniżej** linii.



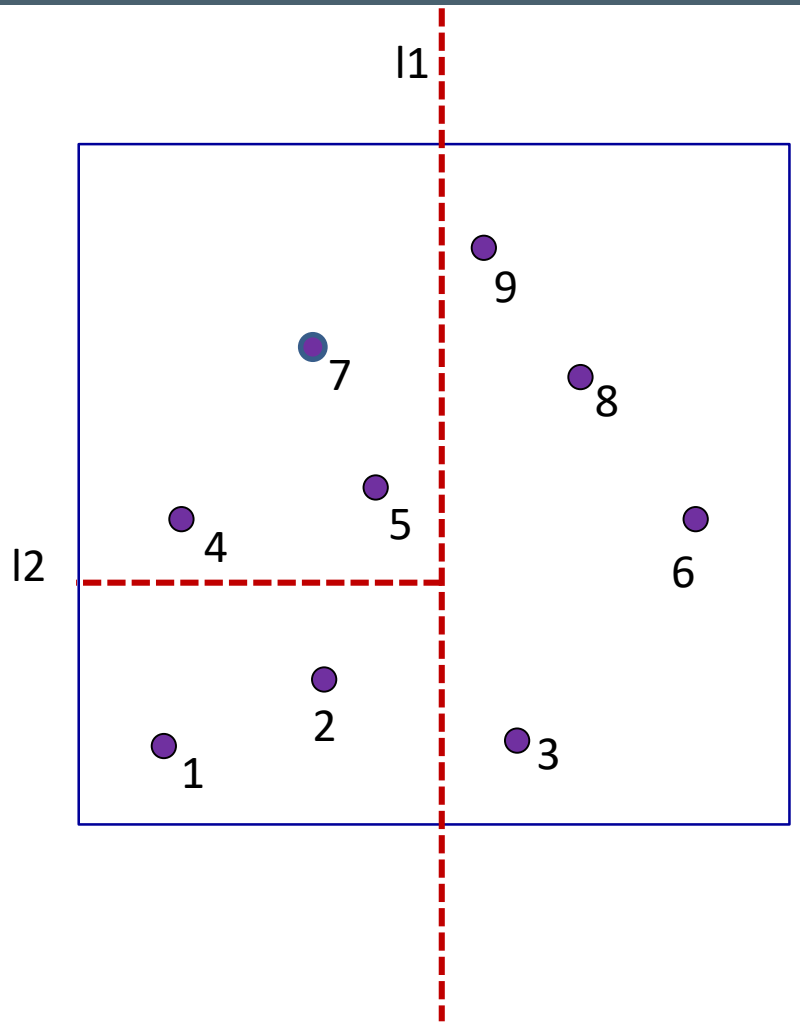
Węzły wewnętrzne drzewa – linie poziome i pionowe, liście drzewa – punkty w przestrzeni

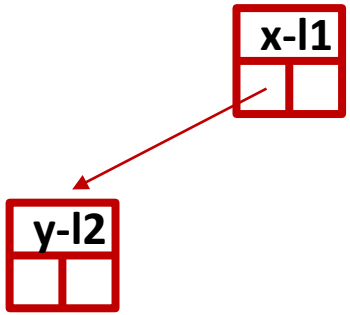


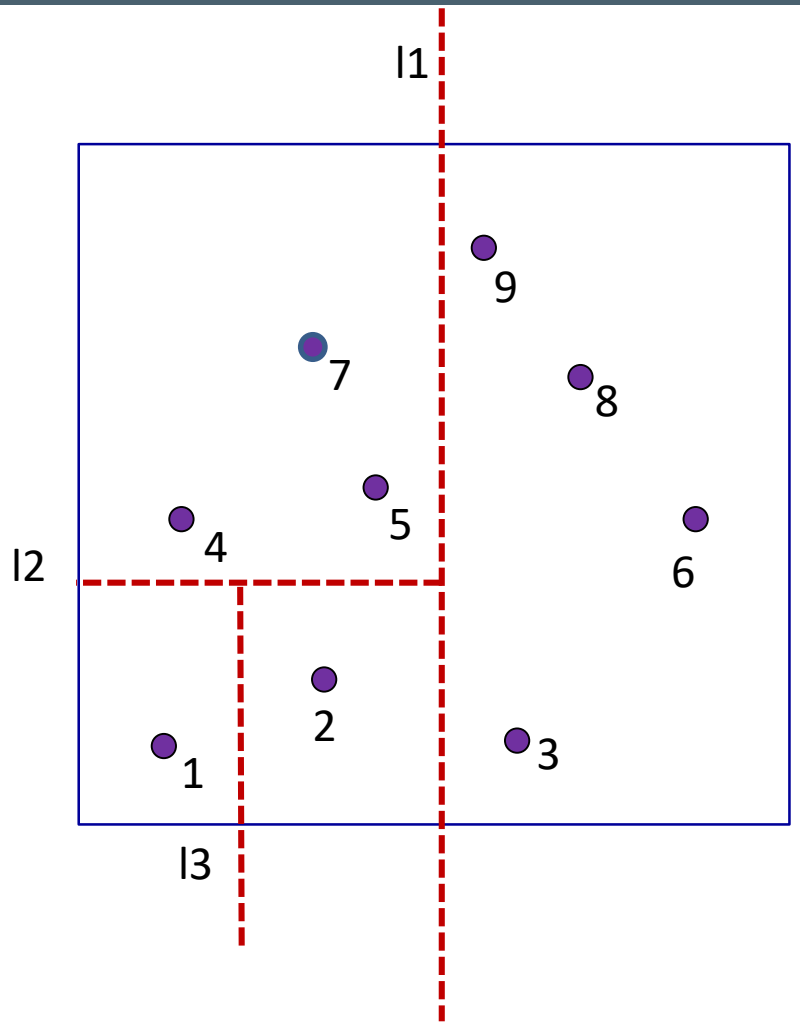
Podział punktów względem współrzędnej x

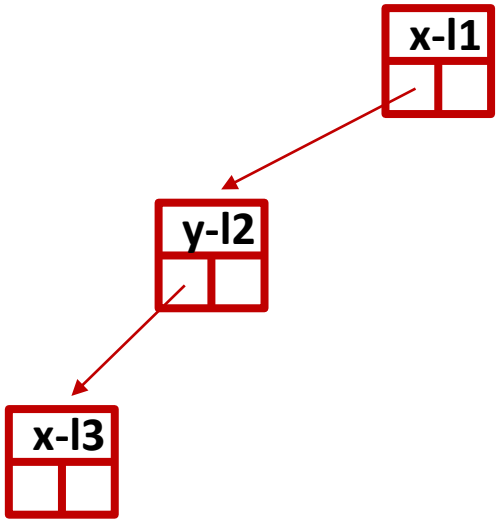

l1
9
7
8
5
4
6
2
1
3

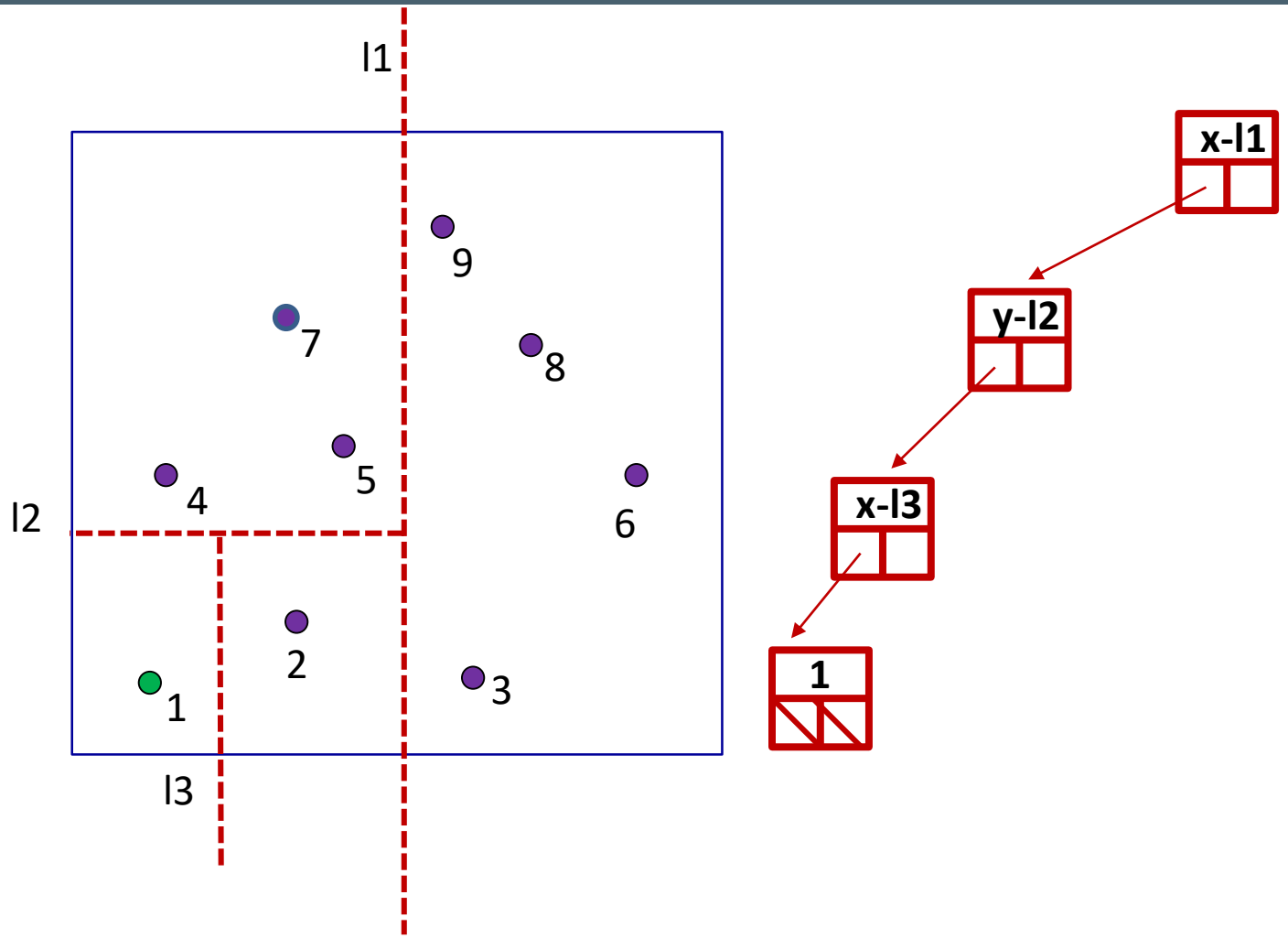

x-l1

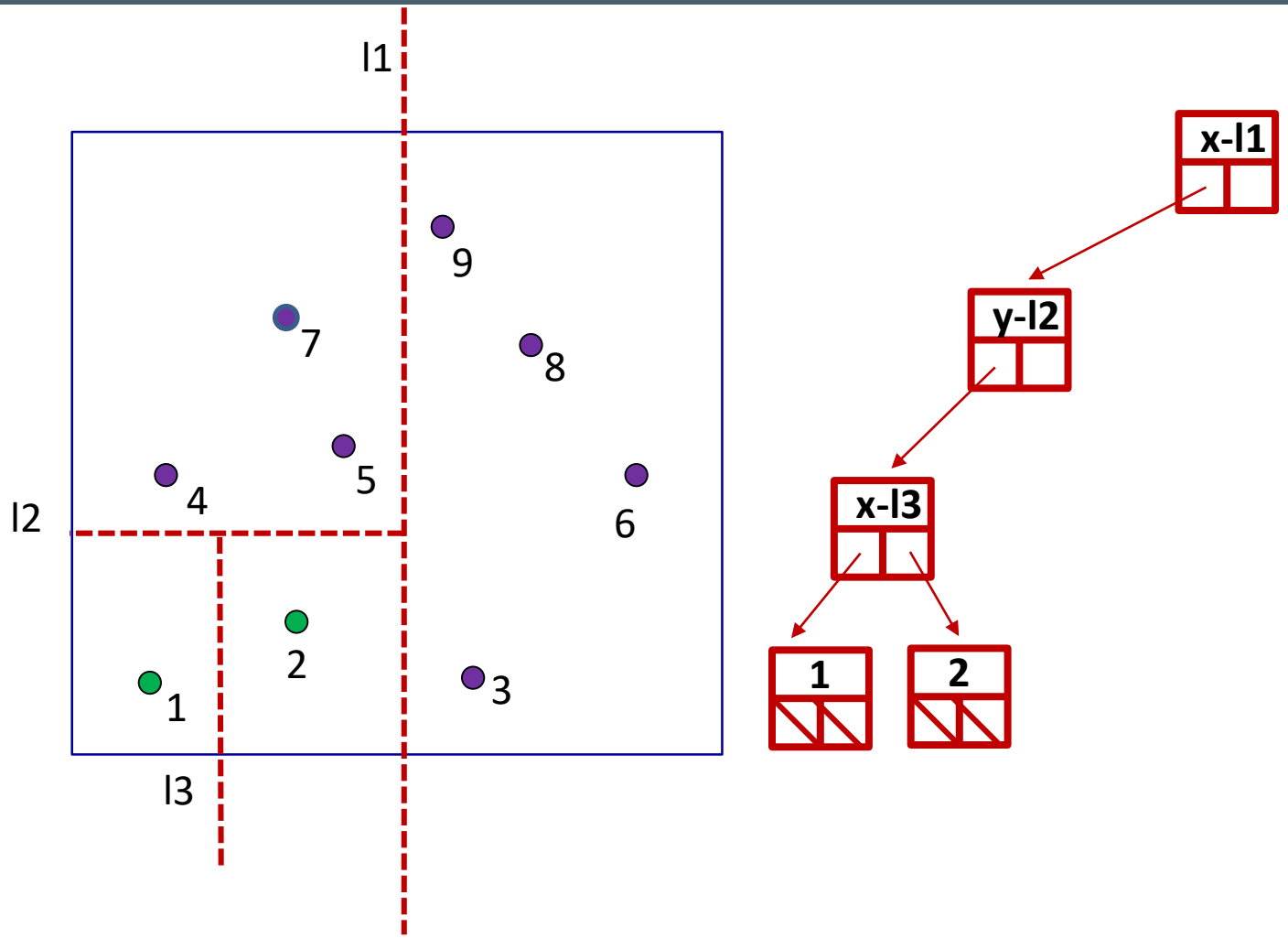

l1
l2
9
7
8
5
4
6
2
1
3

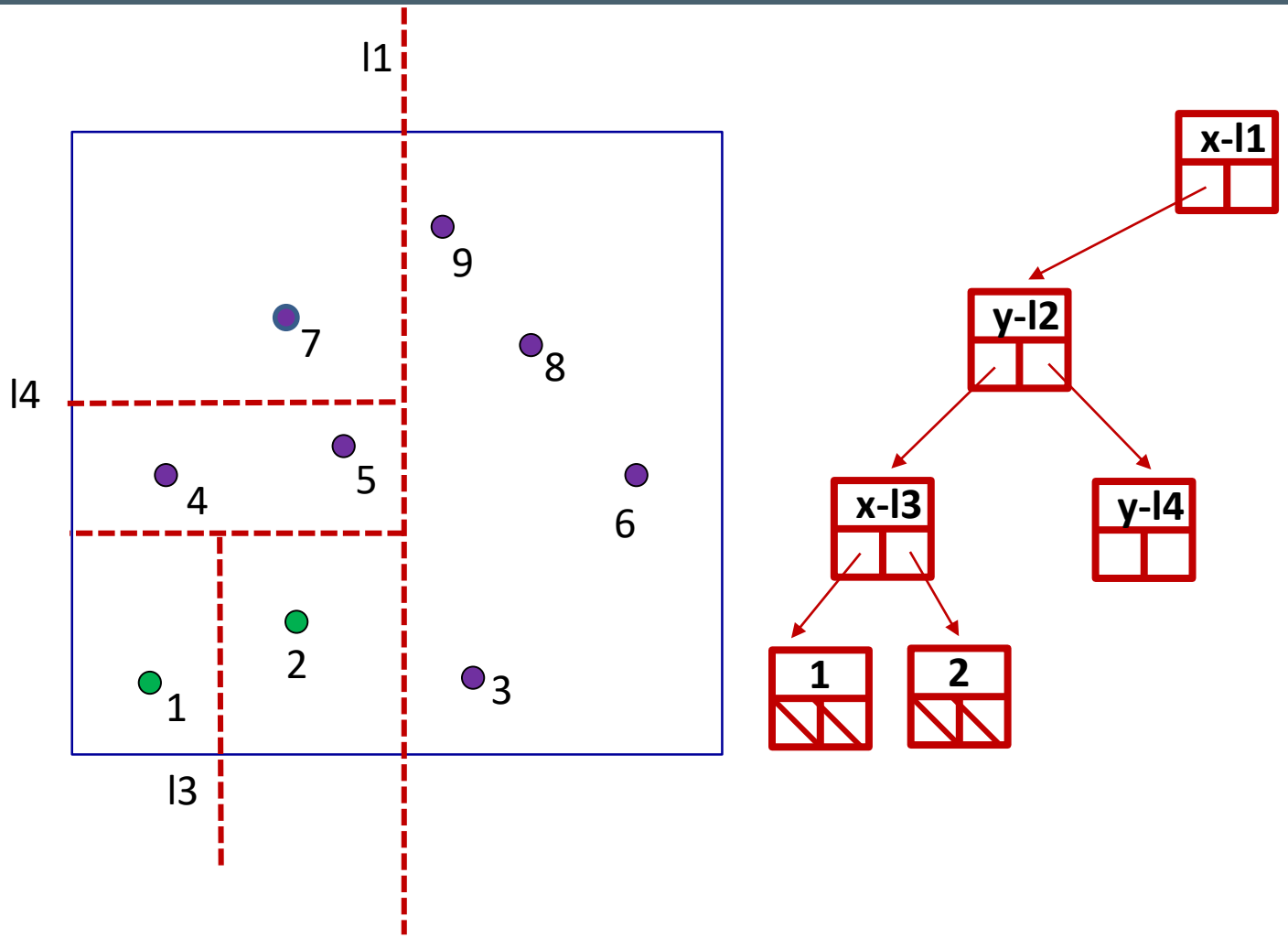

x-l1
y-l2

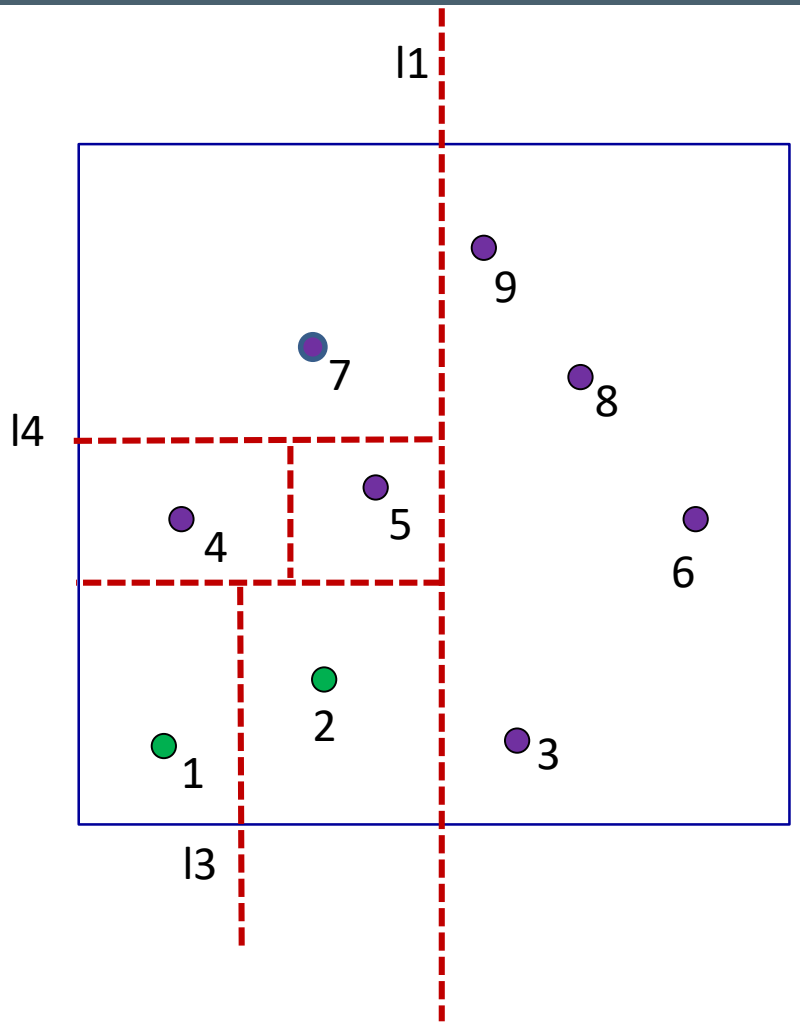

l1
l2
l3
1
2
3
4
5
6
7
8
9

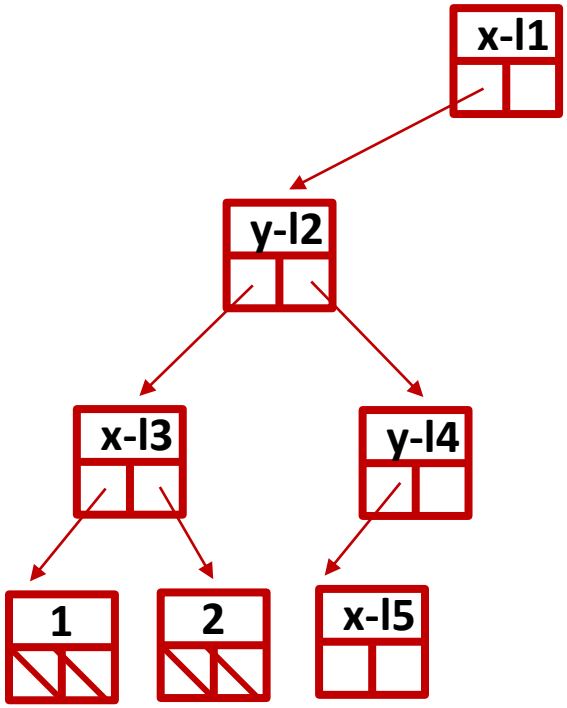

x-l1
y-l2
x-l3

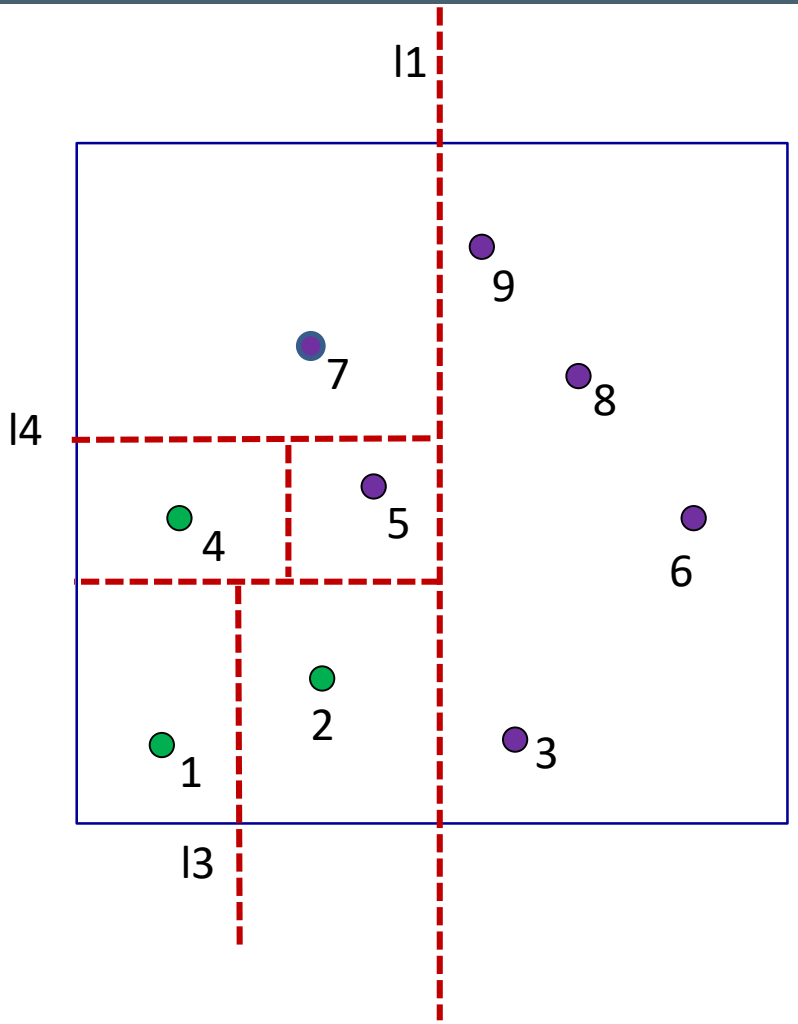

l1
l2
l3
9
7
8
5
4
6
2
1
3
x-l1
y-l2
x-l3
1

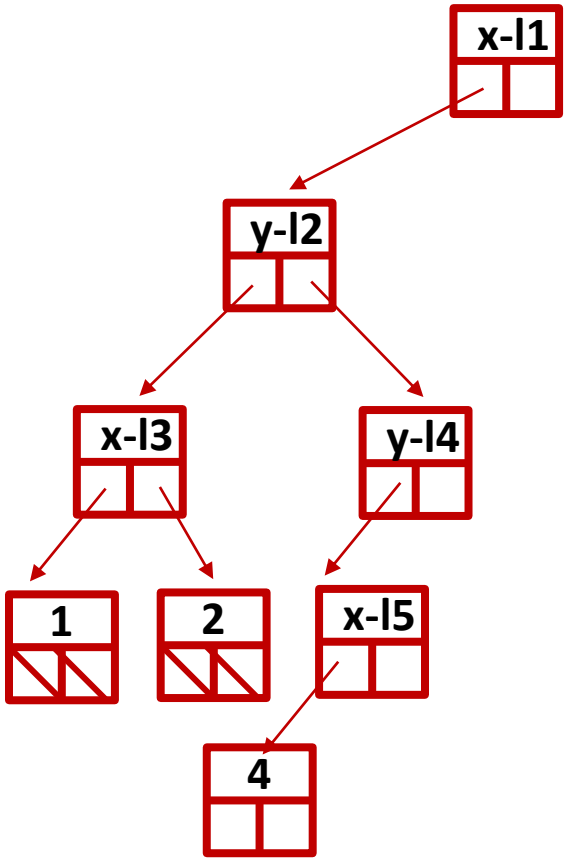

l1
l2
l3
9
7
8
5
4
6
2
1
3
x-l1
y-l2
x-l3
1
2

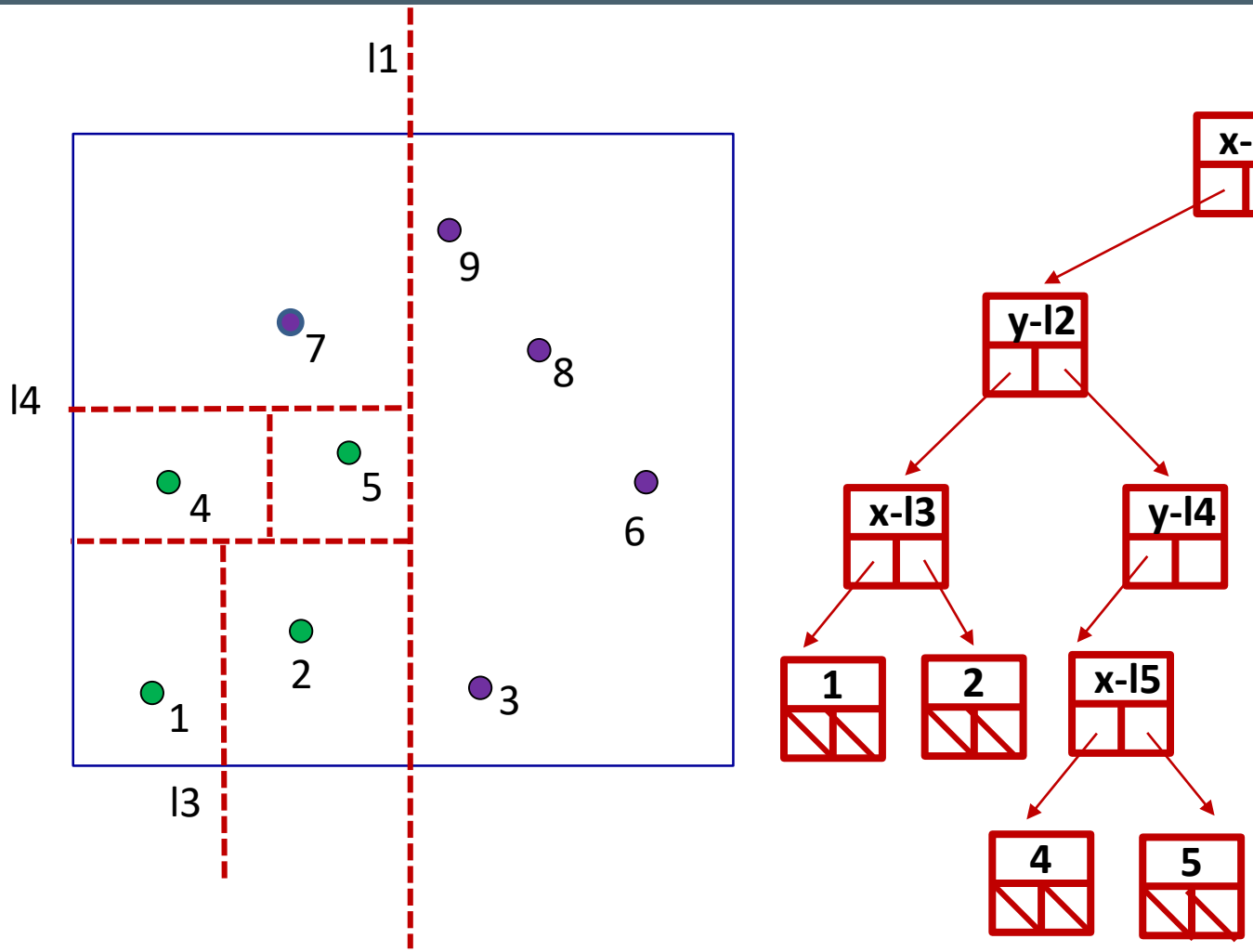

l1
l4
l3
9
7
8
4
5
6
2
1
3
x-l1
y-l2
x-l3
y-l4
1
2


l1
l4
l3
9
7
8
4
5
6
2
1
3

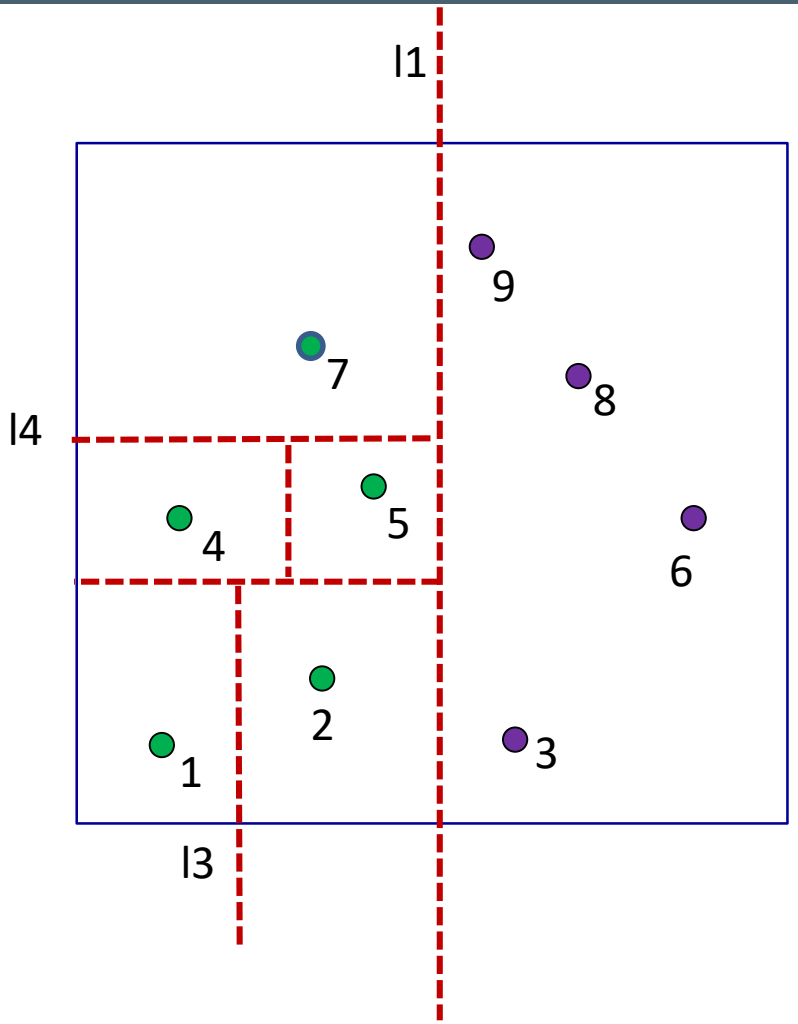

x-l1
y-l2
x-l3
y-l4
1
2
x-l5

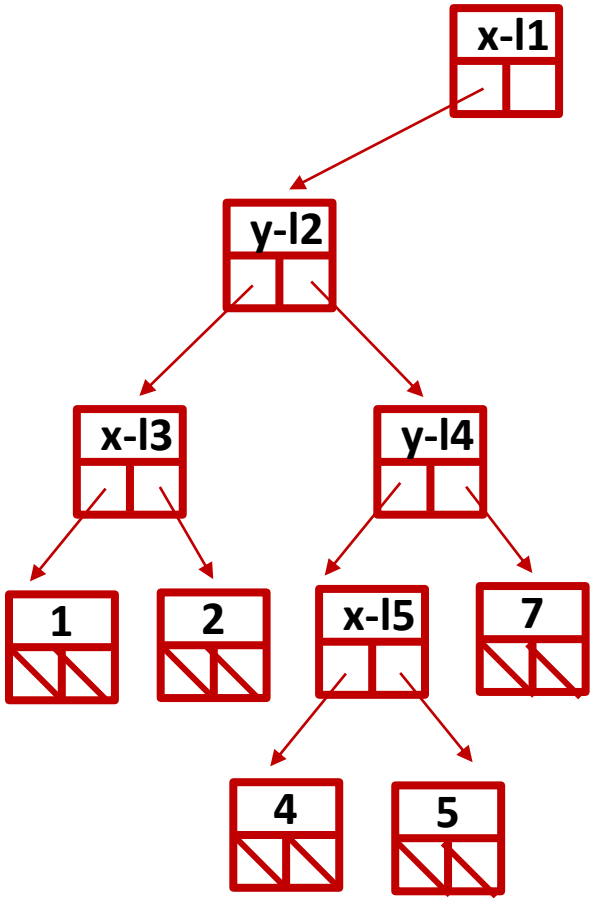

l1
l4
l3
9
7
8
5
4
6
2
1
3

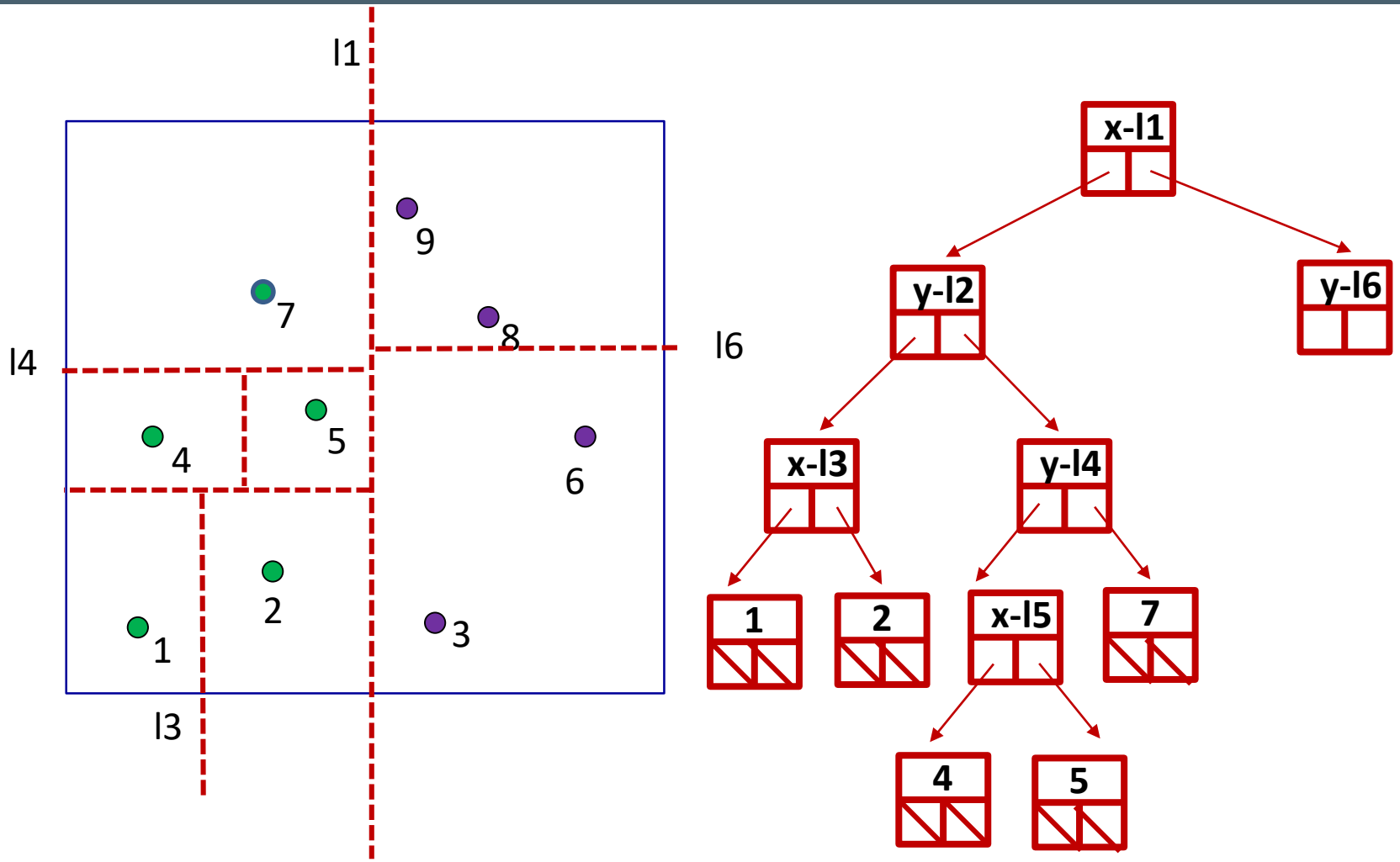

x-l1
y-l2
x-l3
y-l4
1
2
x-l5
4

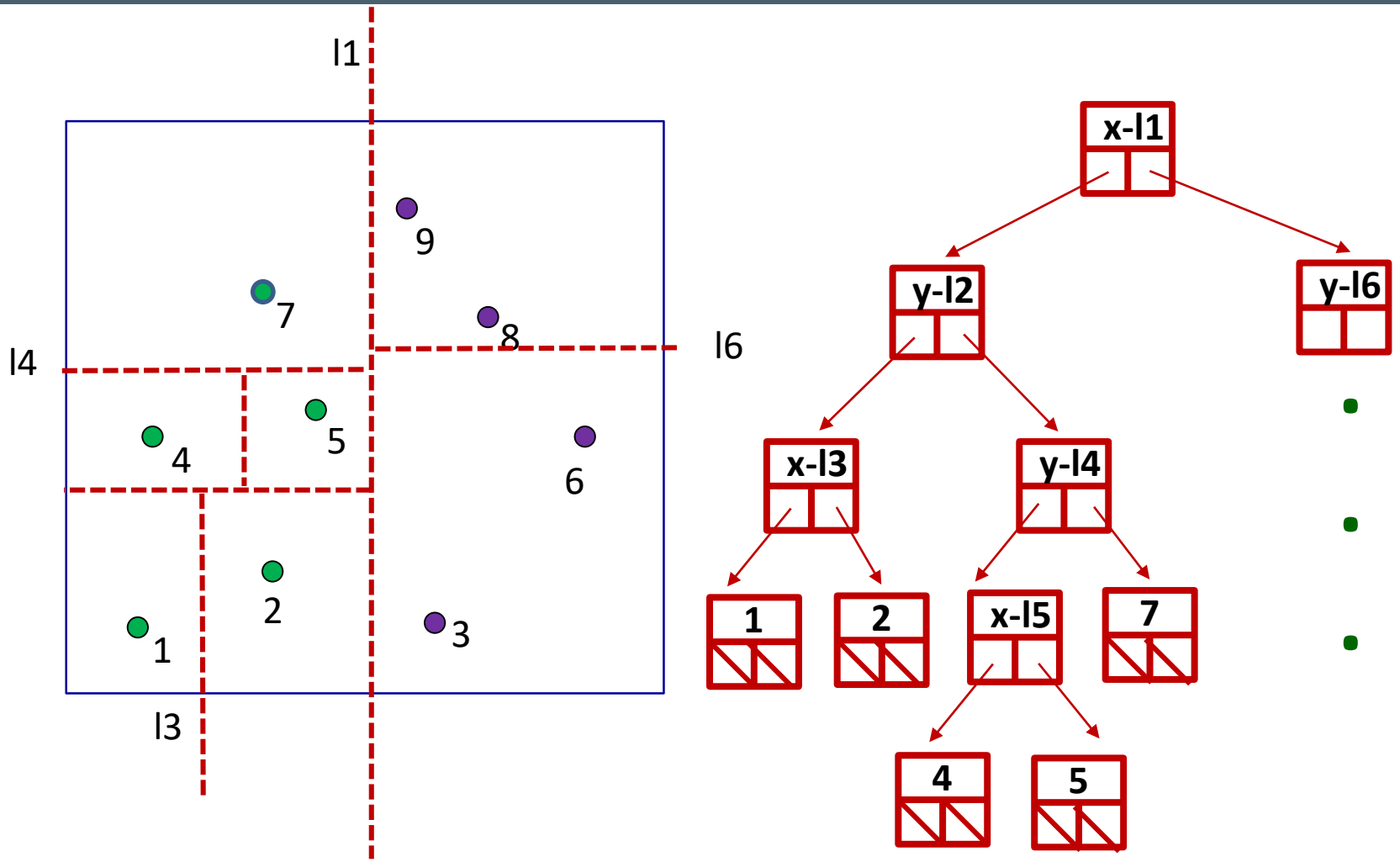

l1
l4
l3
9
7
8
4
5
6
2
1
3

x-l1
y-l2
x-l3
y-l4
1
2
x-l5
4
5


l1
l4
l3
9
7
8
4
5
6
2
1
3

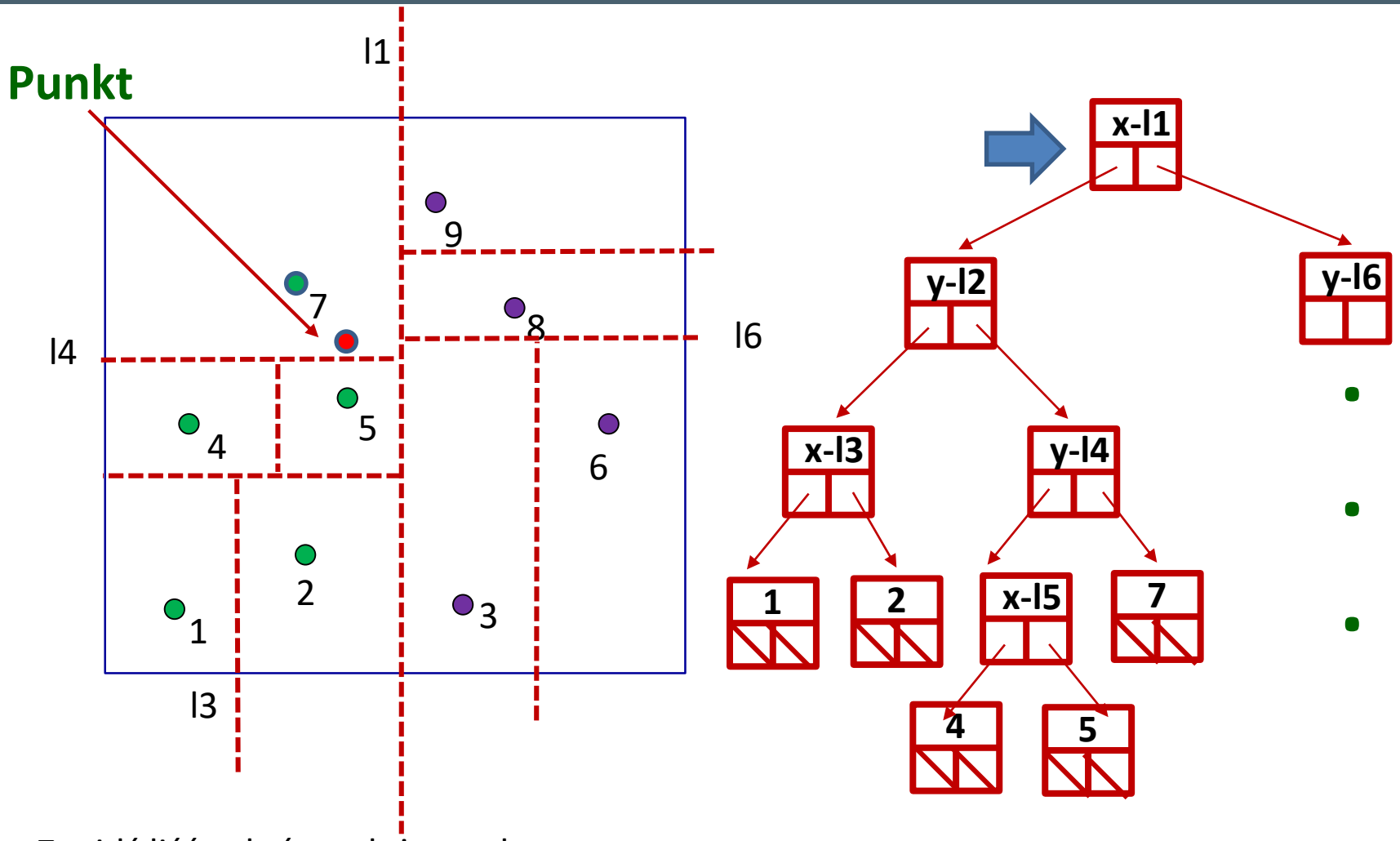

x-l1
y-l2
x-l3
y-l4
1
2
x-l5
7
4
5


l1
9
7
8
l6
l4
4
5
6
2
1
3
l3
x-l1
y-l2
y-l6
x-l3
y-l4
1
2
x-l5
7
4
5


l1
l4
l6
l3
9
7
8
4
5
6
2
1
3
x-l1
y-l2
y-l6
x-l3
y-l4
1
2
x-l5
7
4
5


```
public class Node2D {

        public double SplittingAxis;
        public double SplittingValue;
        public double Key;

        public Node2D Left;
        public Node2D Right;

}
```

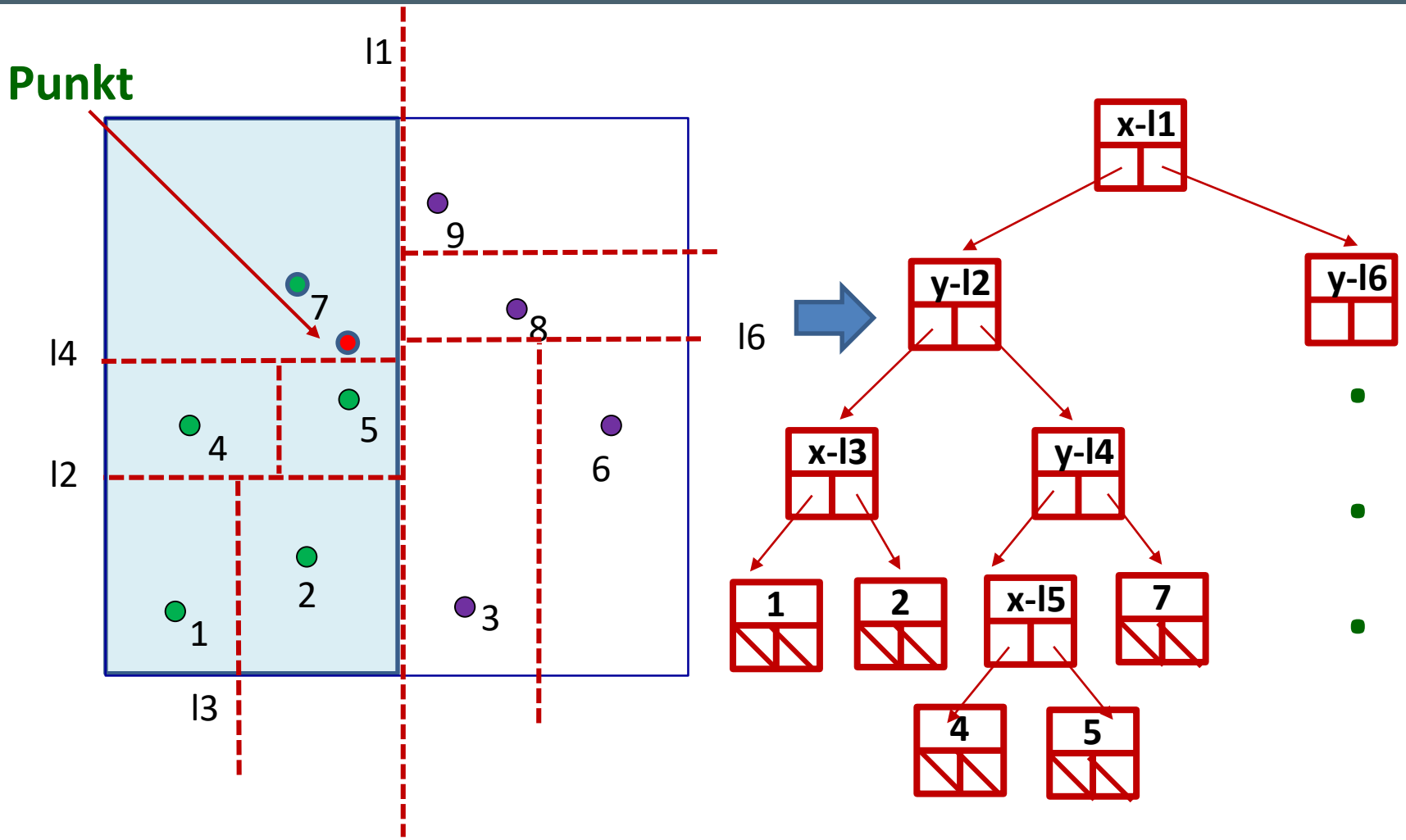
Poszukiwanie najbliższego sąsiada danego punktu



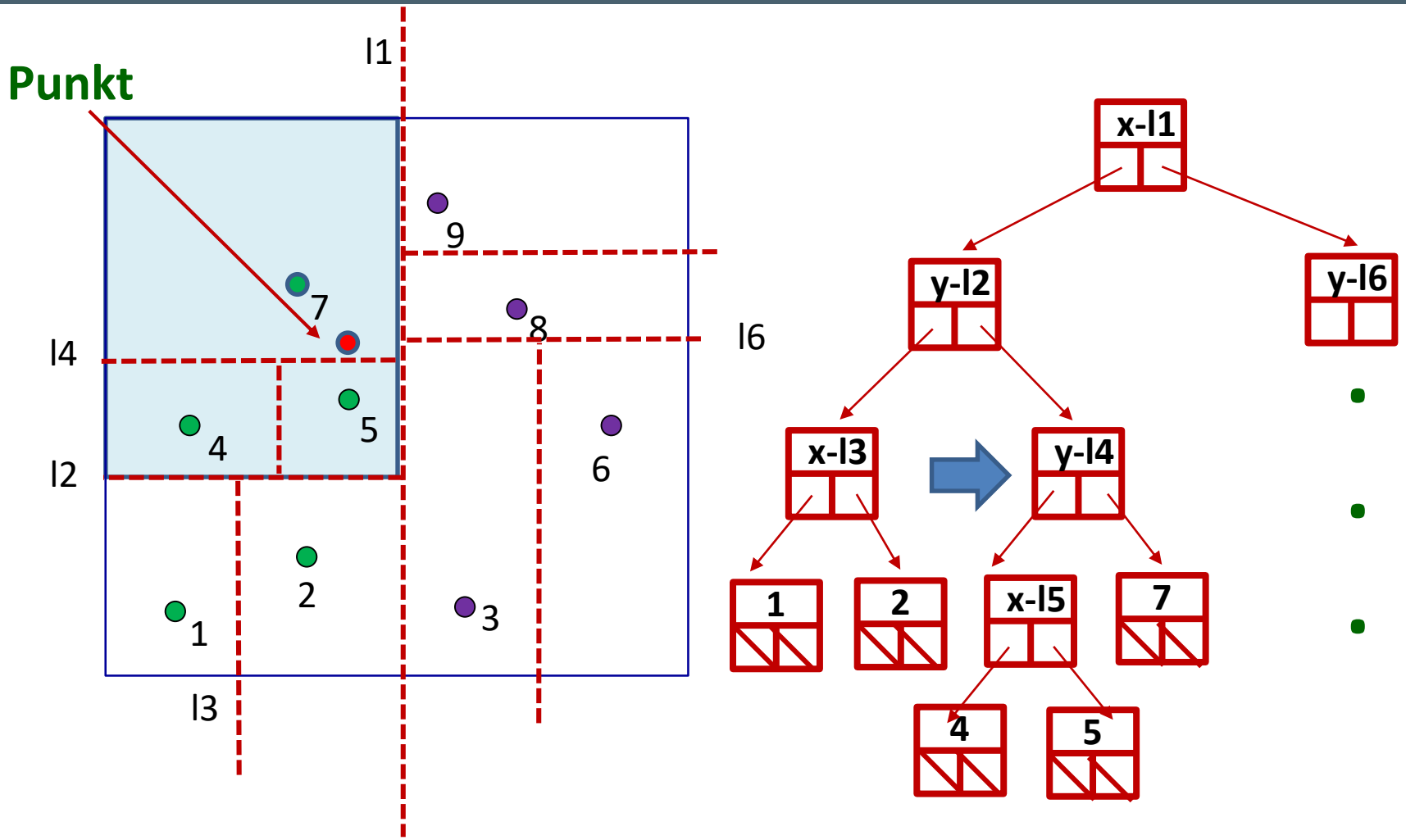
Znajdź liść w którym leży punkt

Sprawdź pozostałe poddrzewa czy nie bliższego puntku

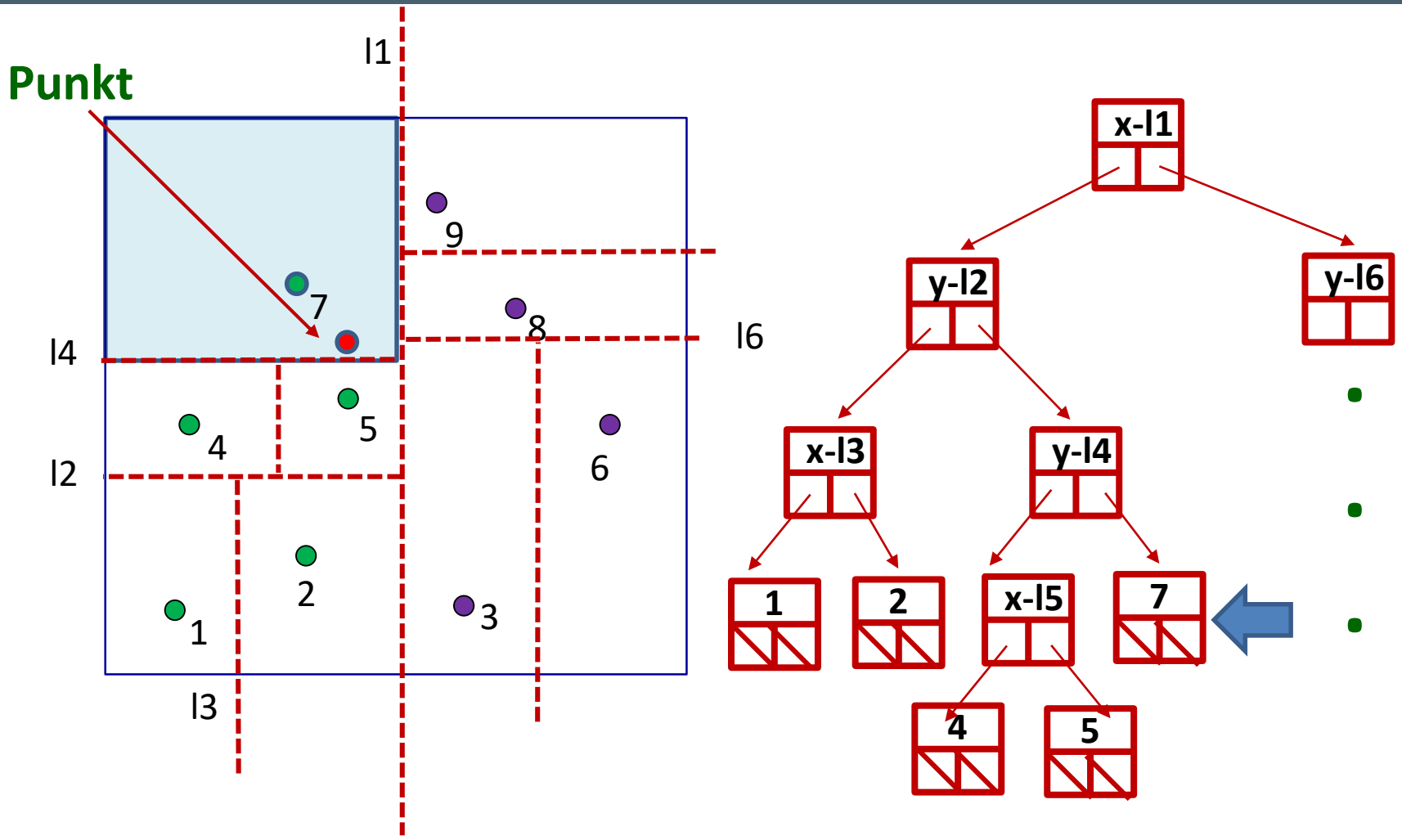
Poszukiwanie najbliższego sąsiada danego punktu



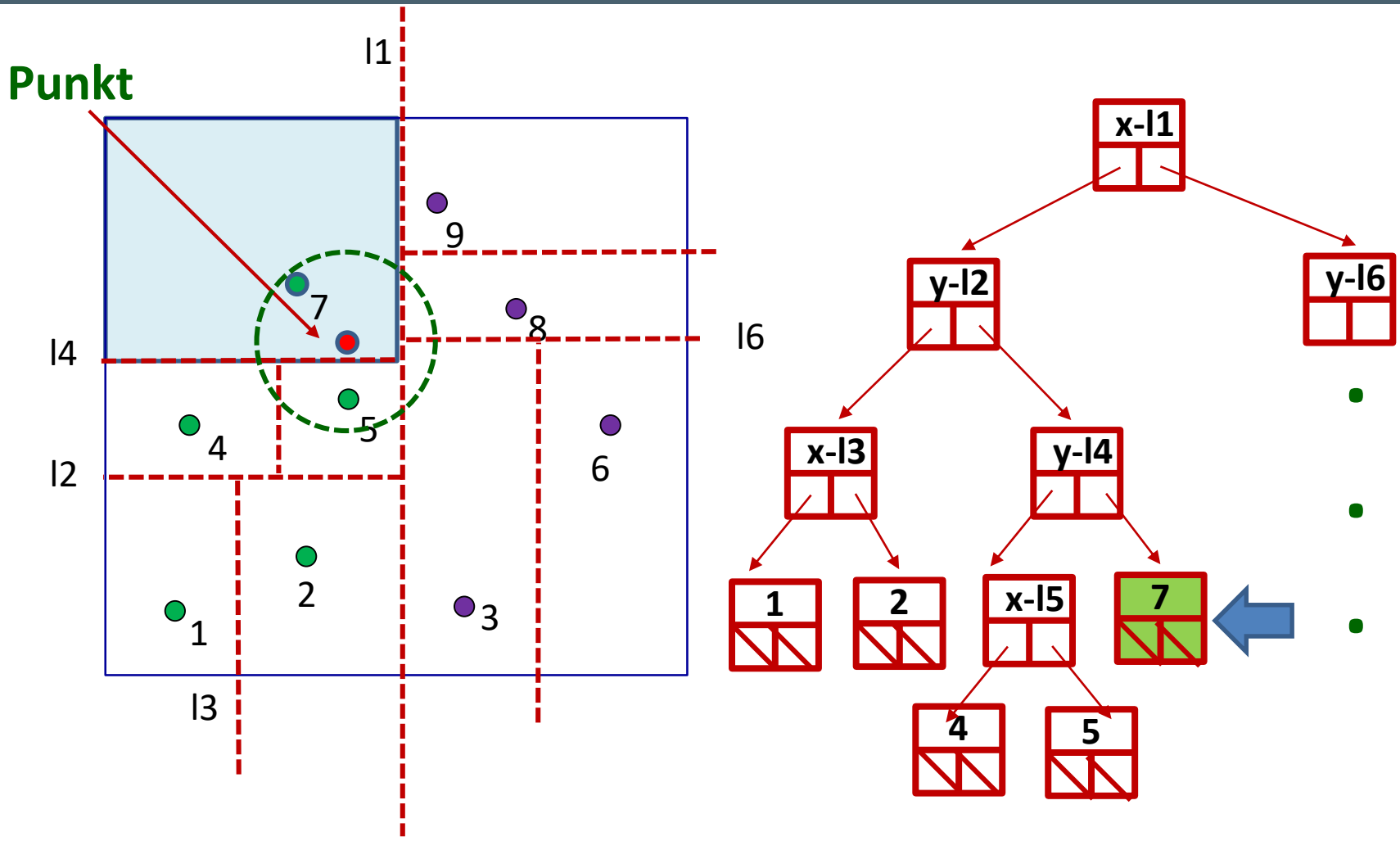
Poszukiwanie najbliższego sąsiada danego punktu



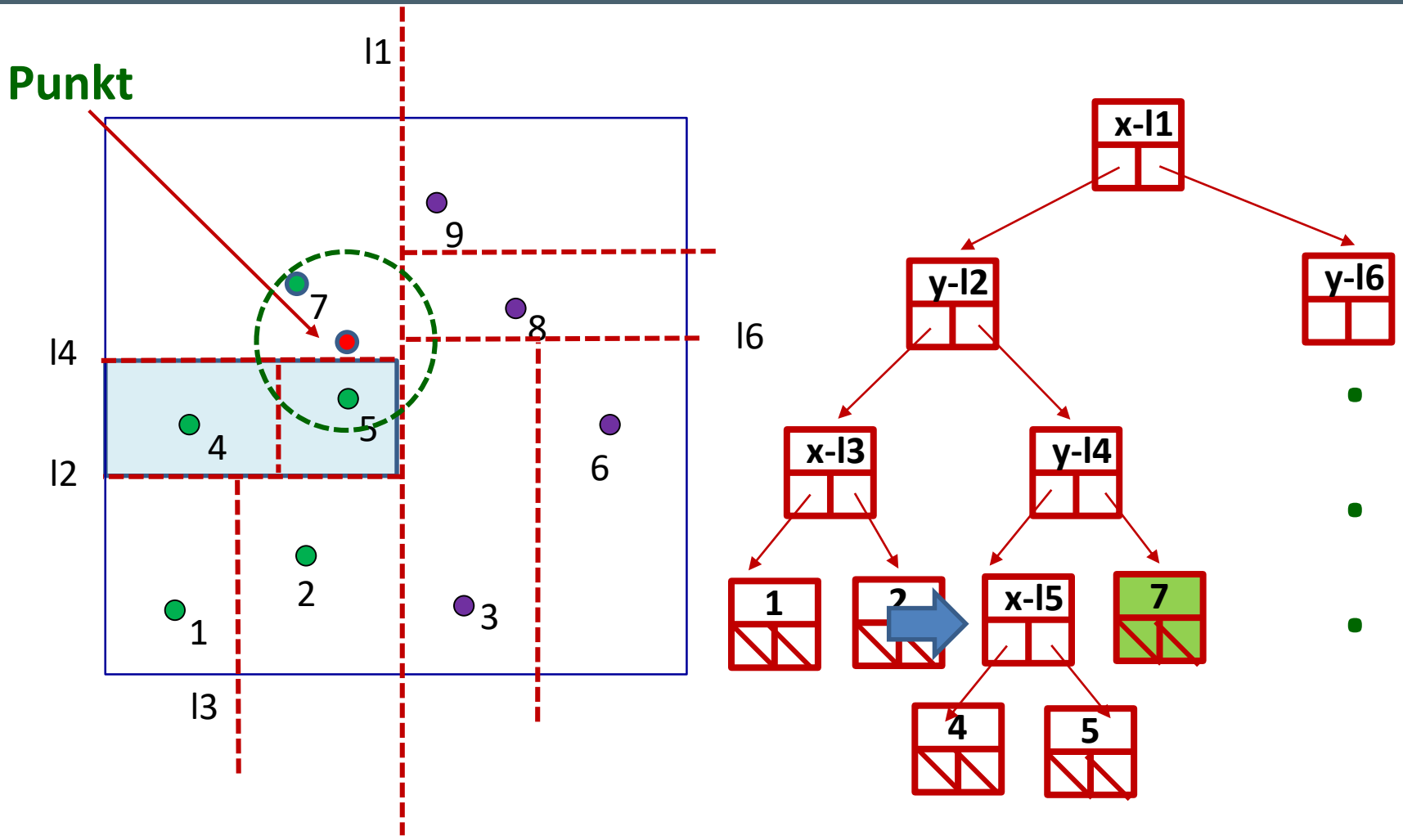
Poszukiwanie najbliższego sąsiada danego punktu



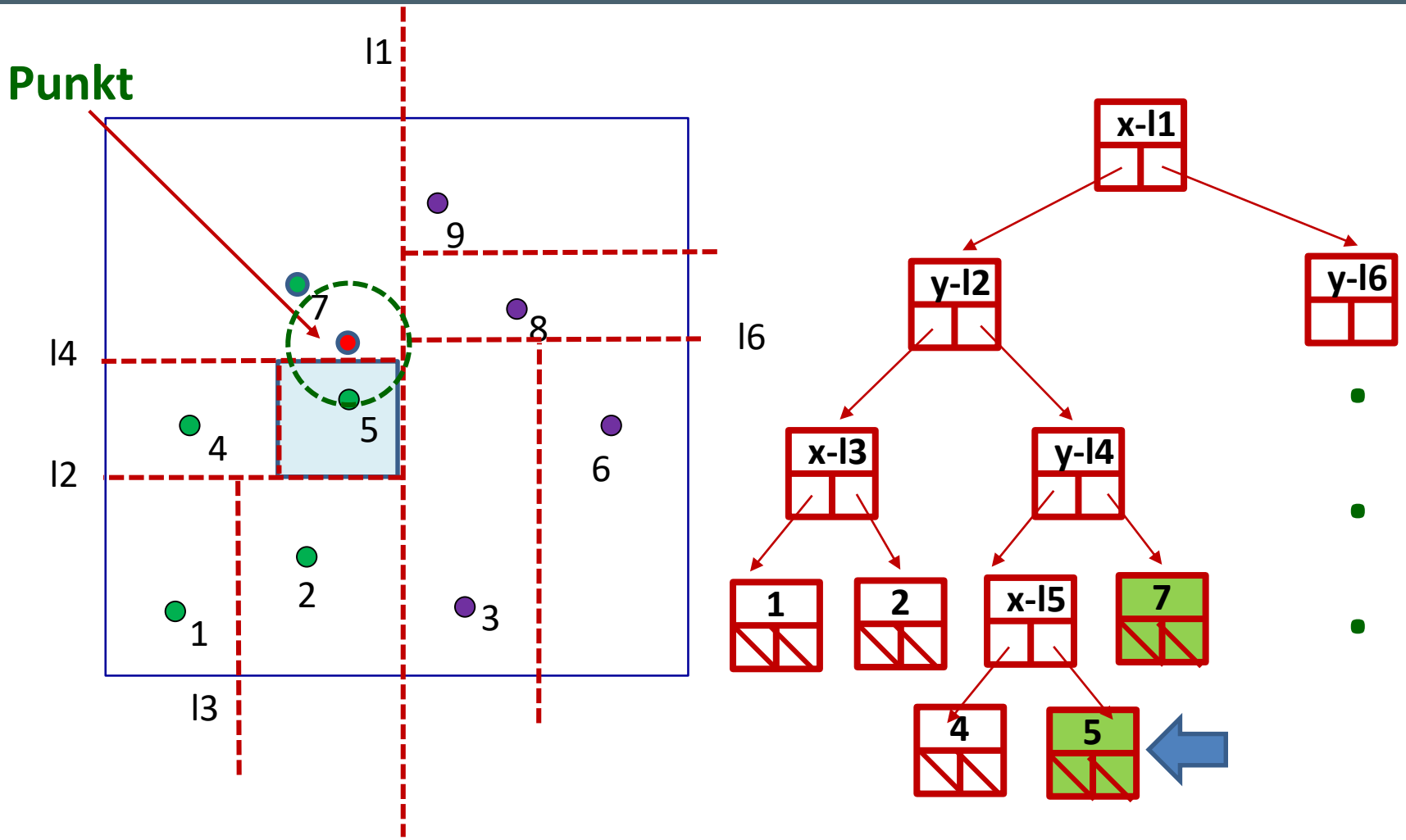
Poszukiwanie najbliższego sąsiada danego punktu



Poszukiwanie najbliższego sąsiada danego punktu



Poszukiwanie najbliższego sąsiada danego punktu



Drzewa KD – 3D