OI-OPPA. European Social Fund
Prague & EU: We invest in your future.

Intersection of rectangles

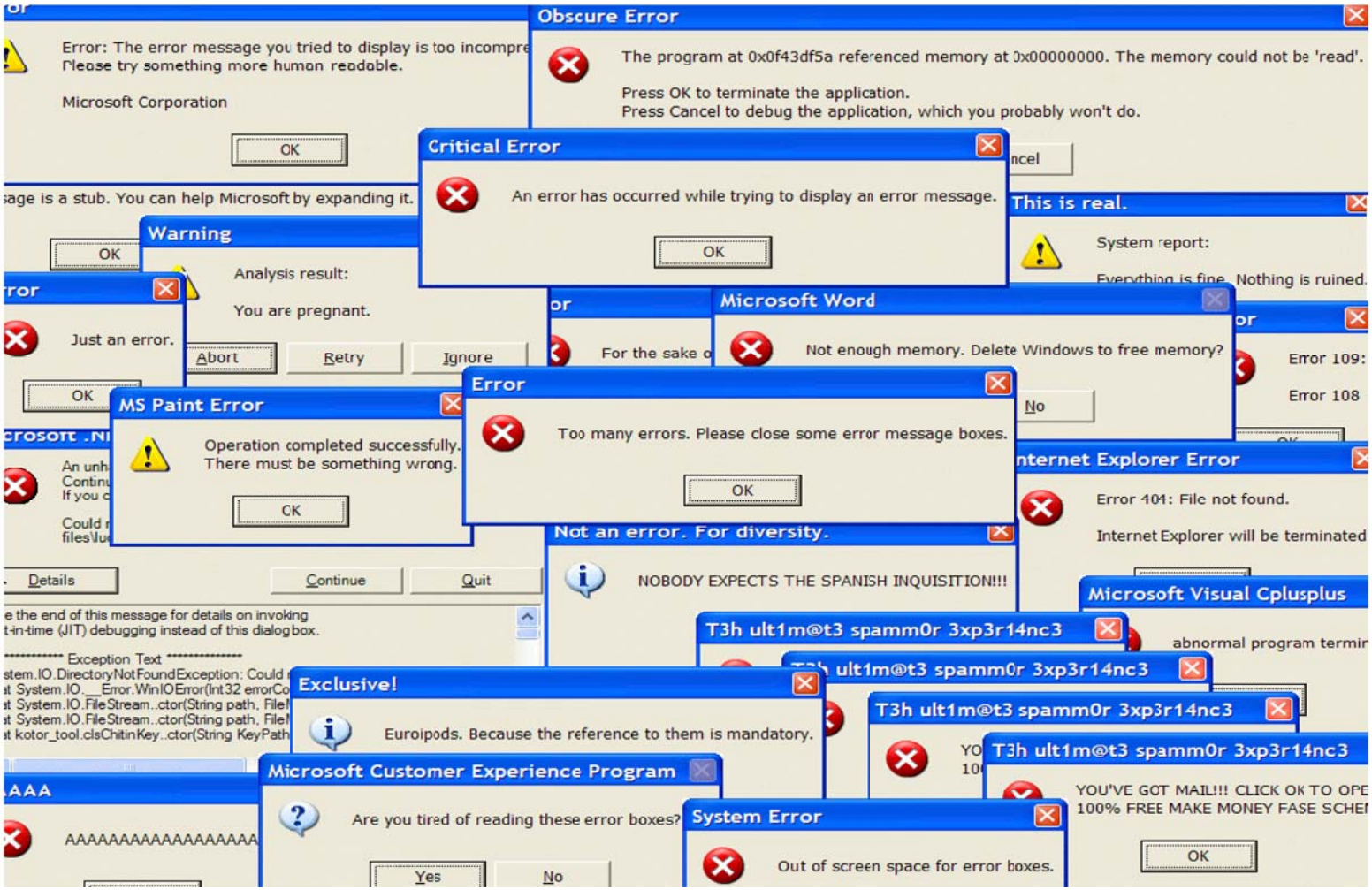
Computational Geometry presentation

22/11/2012

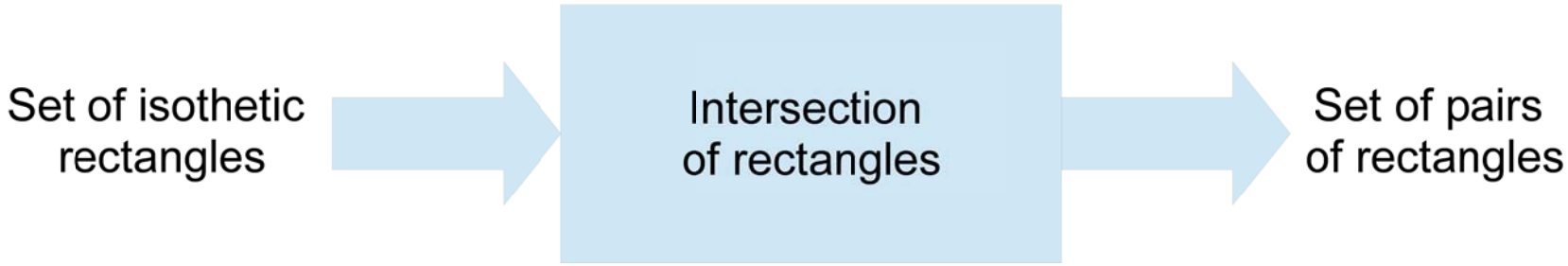
Plan

- Problem definition
- Sweep line
- Interval tree
- Summary

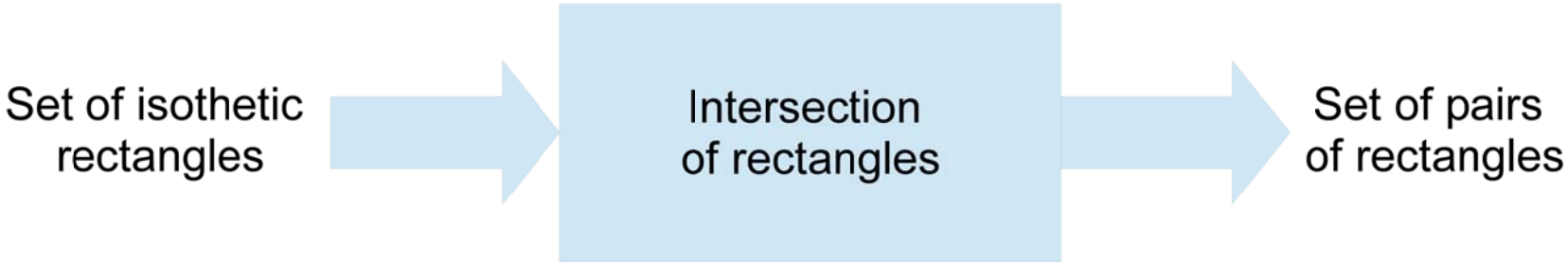
Windowing application



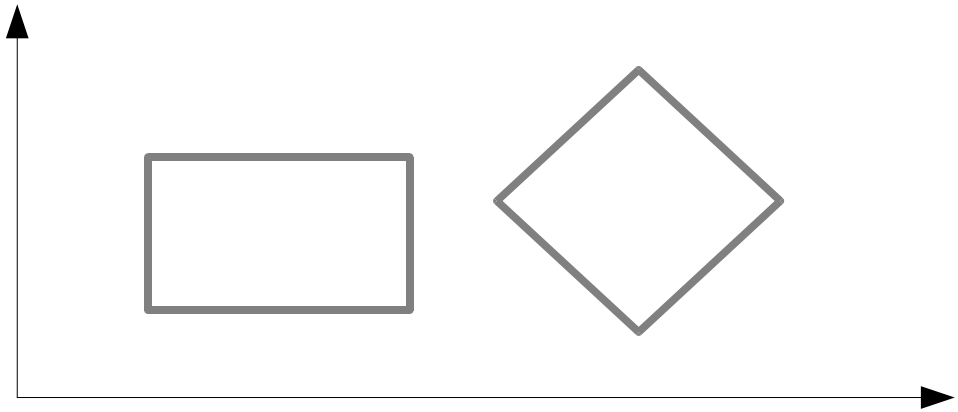
Problem definition



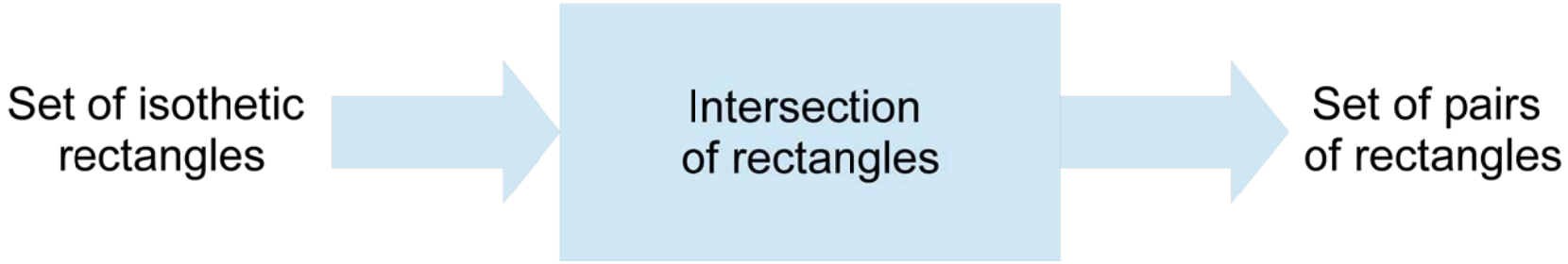
Problem definition

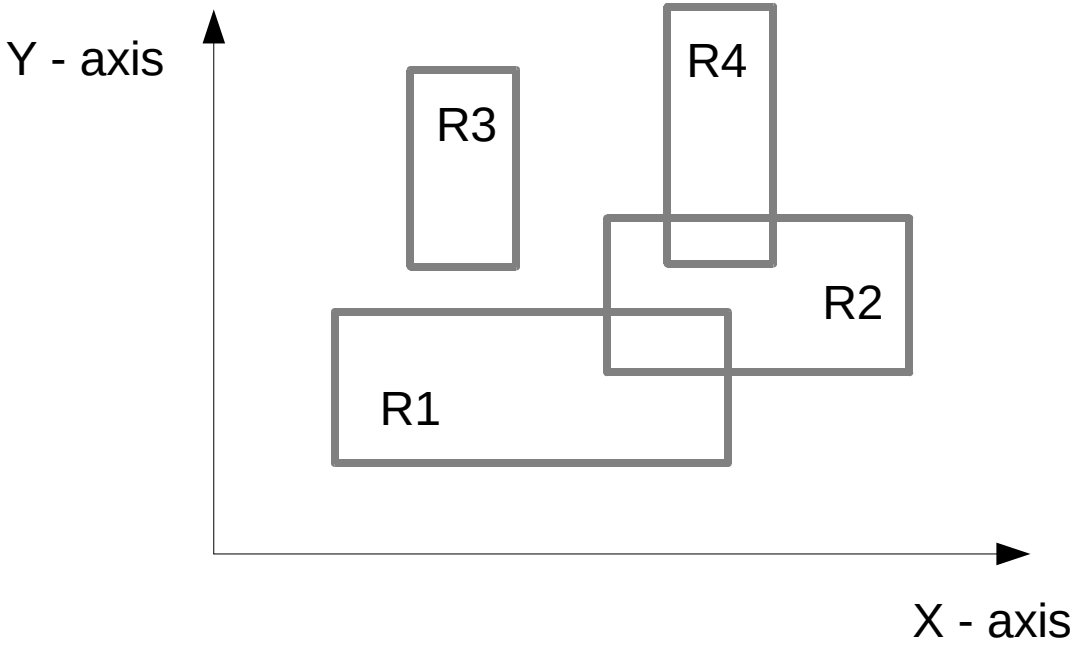


Isothetic rectangle : Each side is parallel to one axis.



Problem definition





{ {R1, R2}, {R2, R4} }

Sweep line

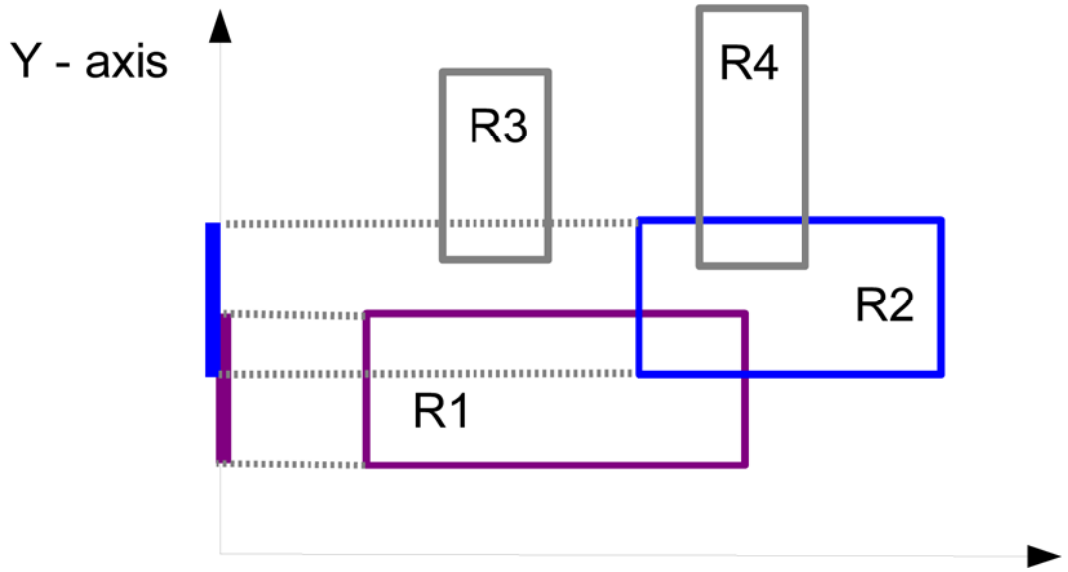
- First step of the algorithm
- We assume that we can use a function :

OverlappingOnY(Q, R)

Input
- Q : a query rectangle
- R : a list of rectangles

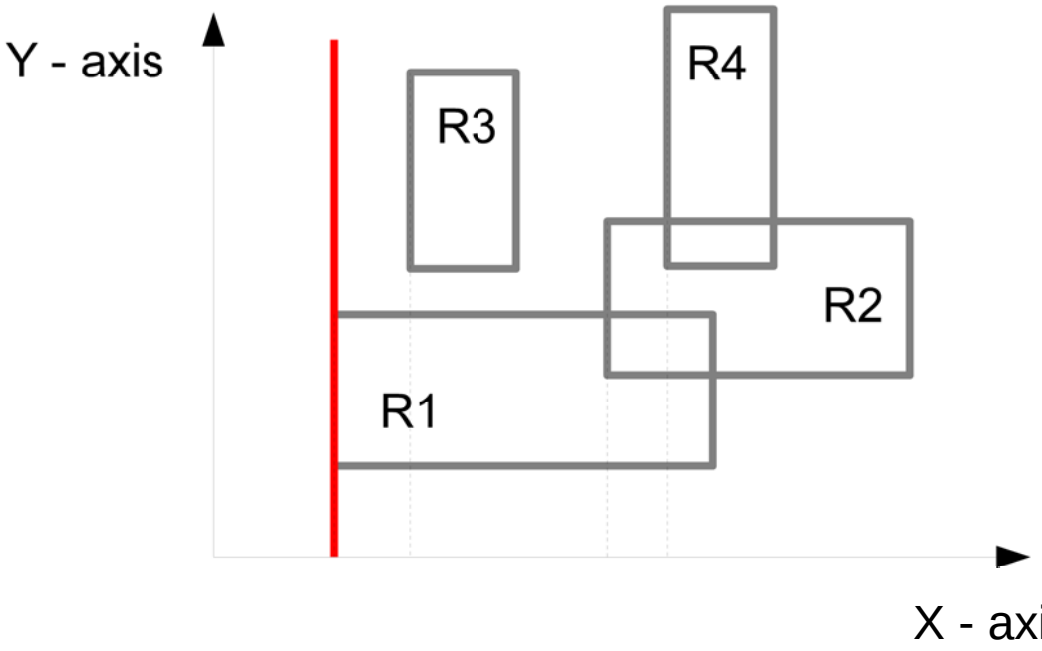
Output :
All the intersection pairs of the projection of Q and R on Y axis.

Sweep line



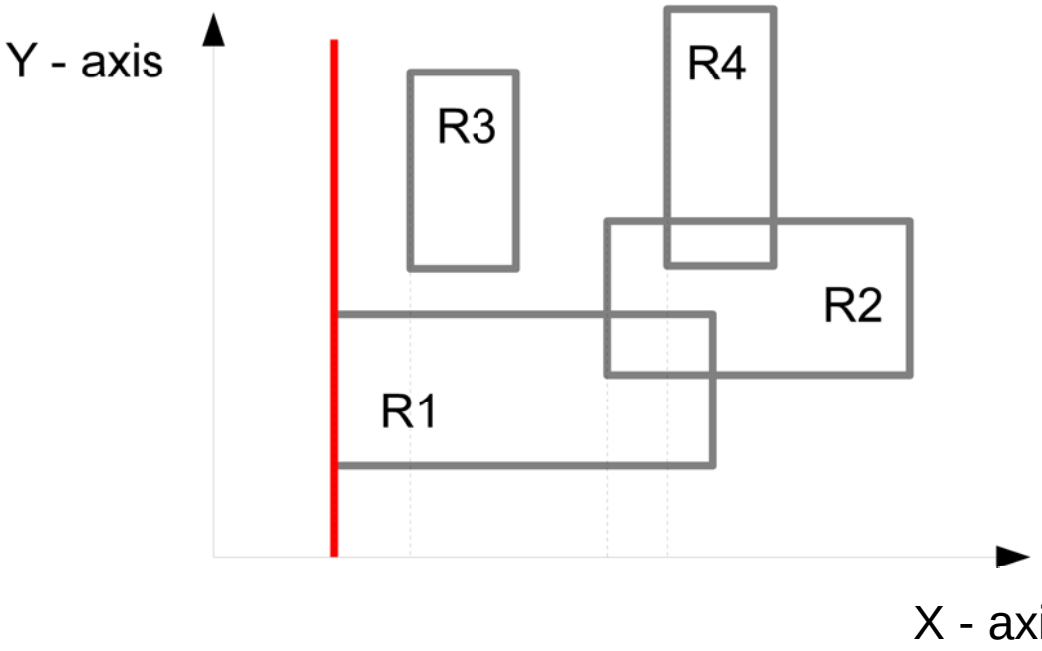
OverlappingOnY(R1, {R1, R2, R3, R4}) = {R1, R2}

Sweep line




```
Res = {}
AR = {}
For each R reached by the sweep line
    Update AR
    Res = Res U OverlappingOnY(R, AR)
```

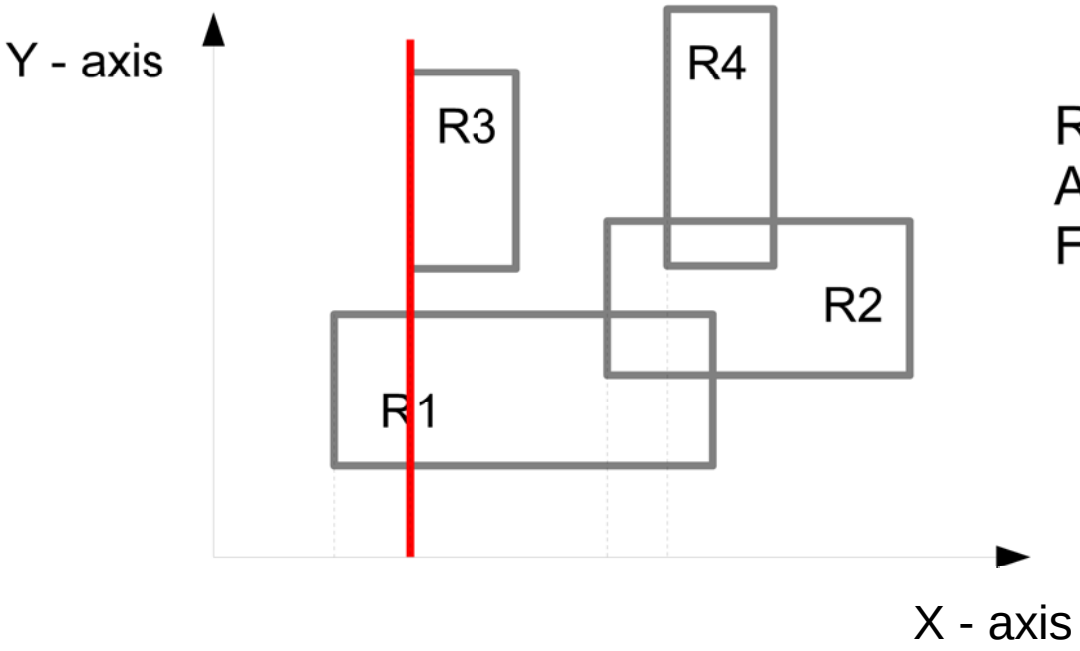
Sweep line



Res = {}
AR = {}
For each R reached by the sweep line
    Update AR
    Res = Res U OverlappingOnY(R, AR)

AR = {R1}
Res = {}

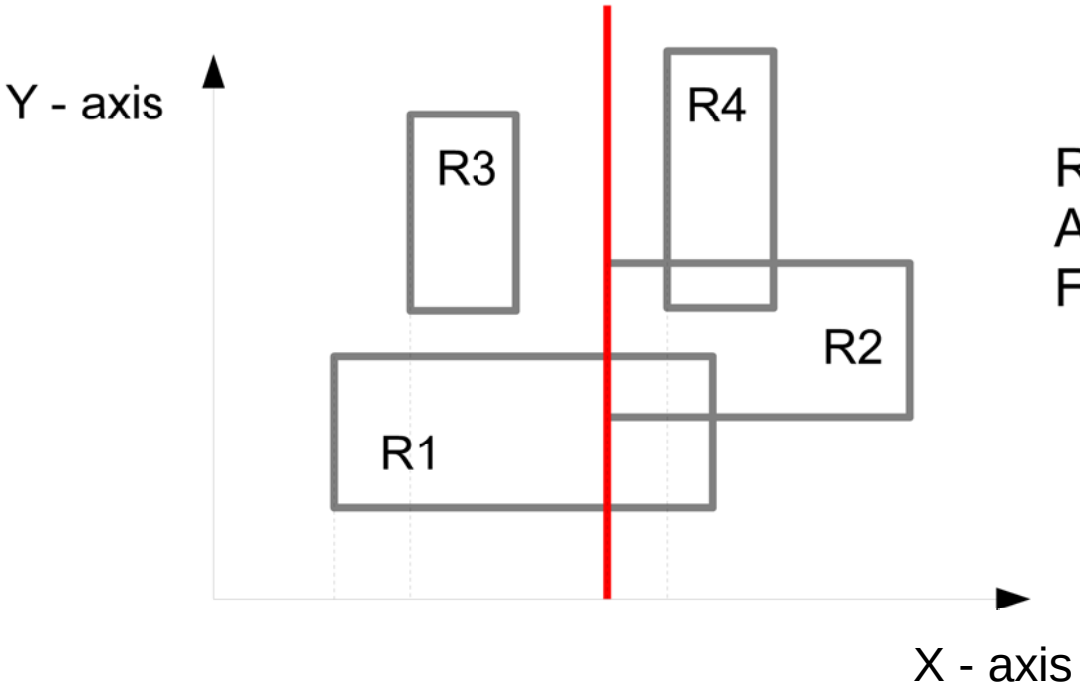
Sweep line



Res = {}
AR = {}
For each R reached by the sweep line
    Update AR
    Res = Res U OverlappingOnY(R, AR)

AR = {R1, R3}
Res = {}

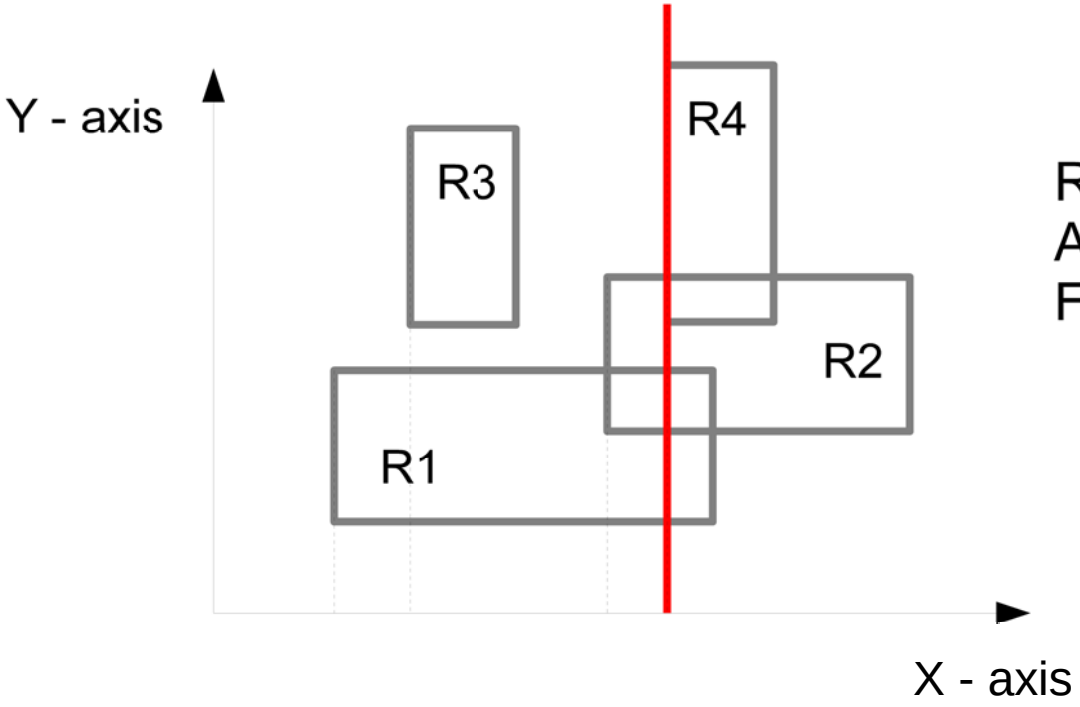
Sweep line



Res = {}
AR = {}
For each R reached by the sweep line
    Update AR
    Res = Res U OverlappingOnY(R, AR)

AR = {R1, R2}
Res = { {R1, R2} }

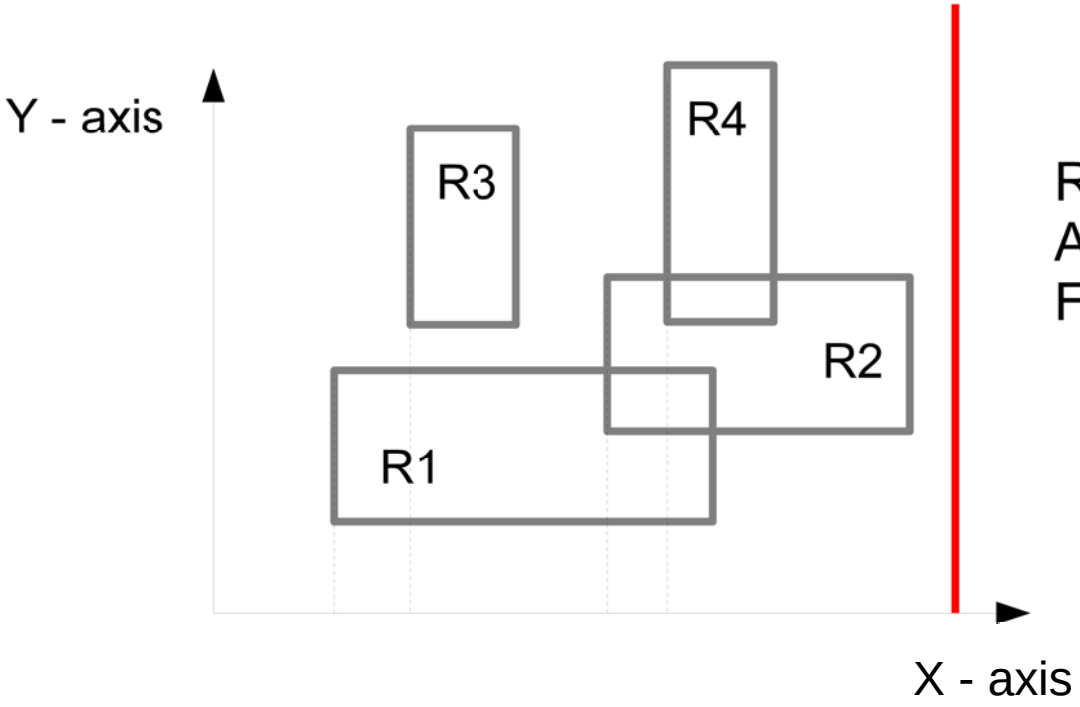
Sweep line



Res = {}
AR = {}
For each R reached by the sweep line
    Update AR
    Res = Res U OverlappingOnY(R, AR)

AR = {R1, R2, R4}
Res = { {R1, R2}, {R2, R4} }

Sweep line



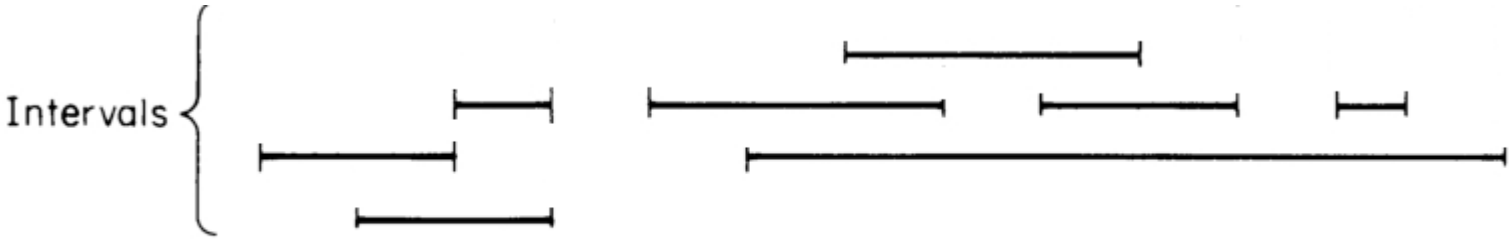
Res = {}
AR = {}
For each R reached by the sweep line
    Update AR
    Res = Res U OverlappingOnY(R, AR)

AR = {}
Res = { {R1, R2}, {R2, R4} }

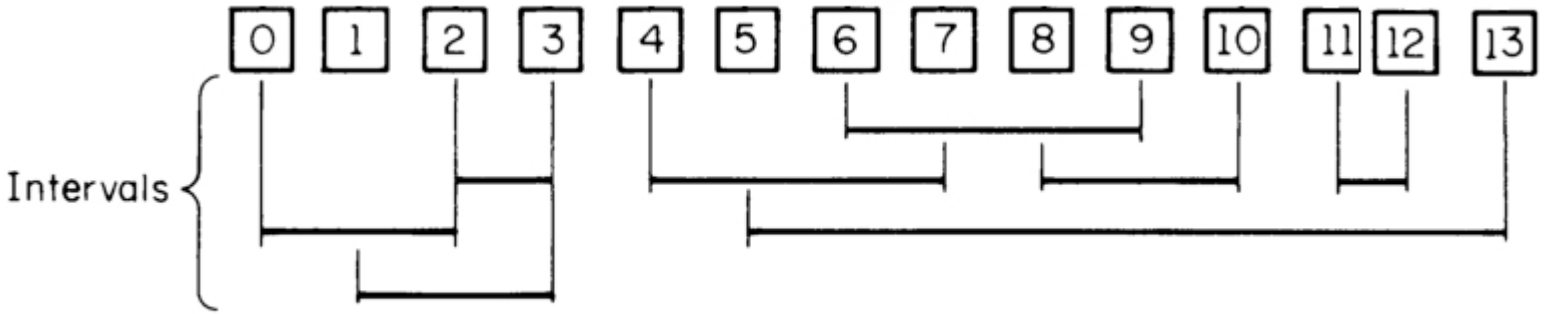
Plan

- Problem definition
- Sweep line
- Interval tree
- Summary

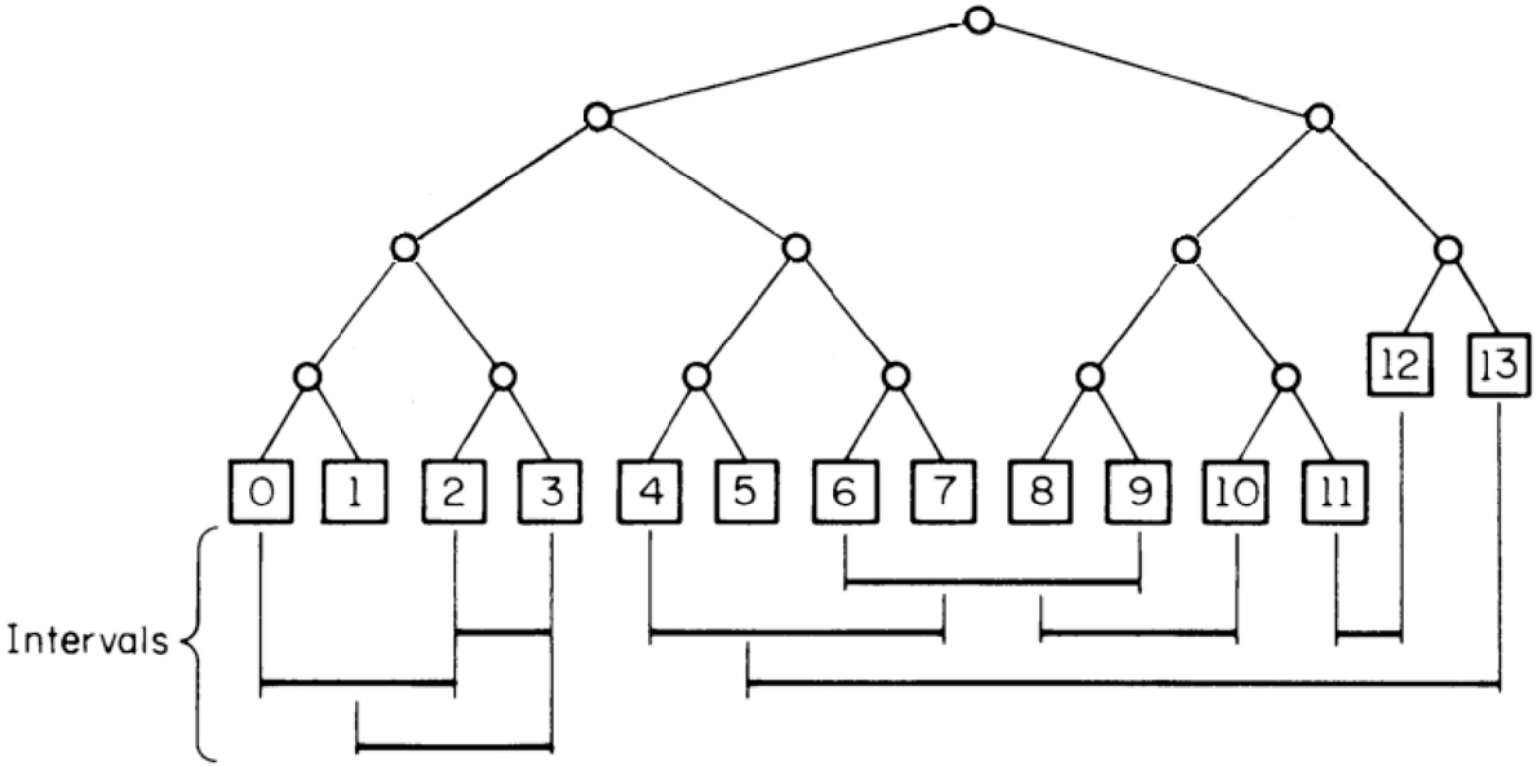
Interval tree construction



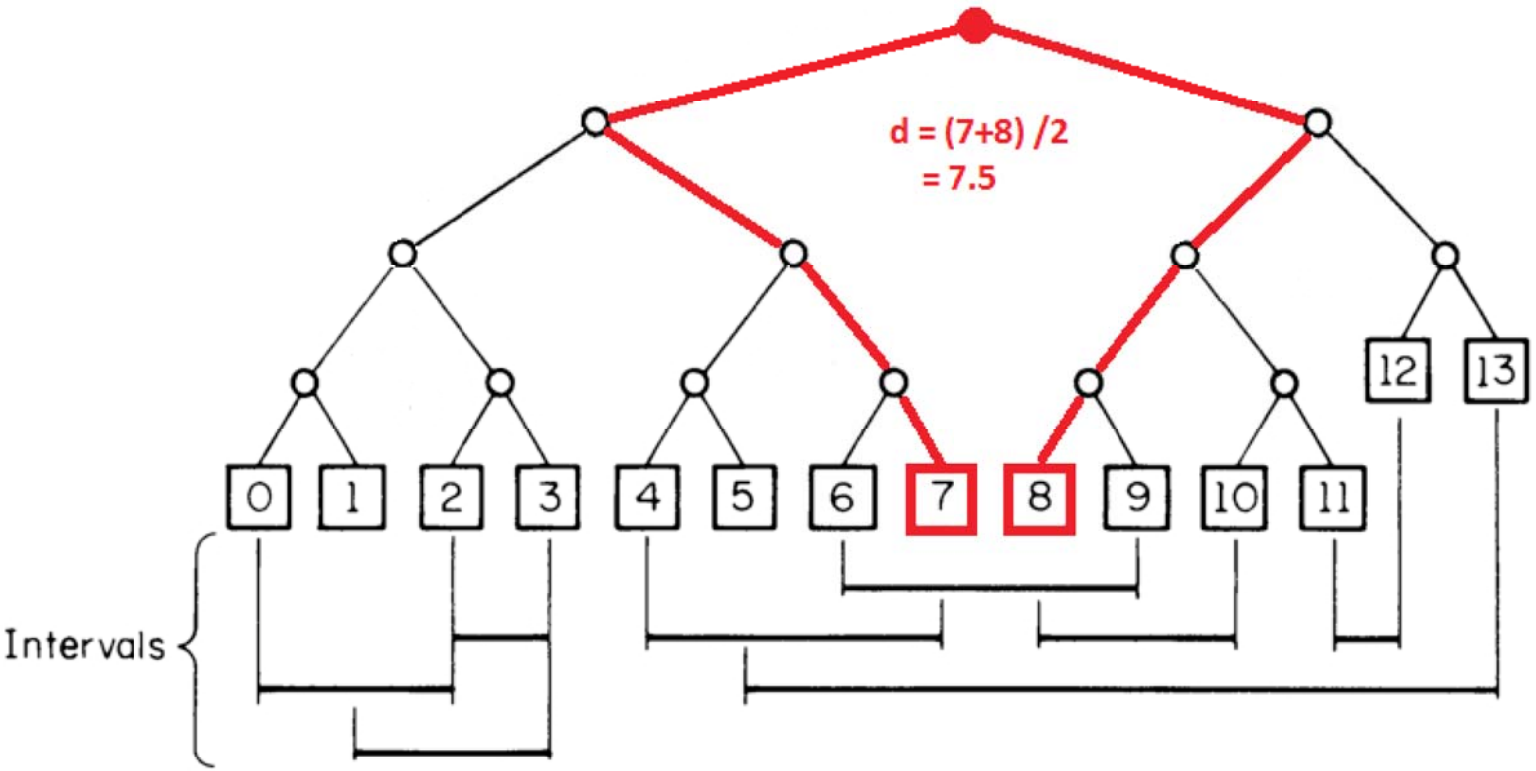
Interval tree construction



Interval tree construction



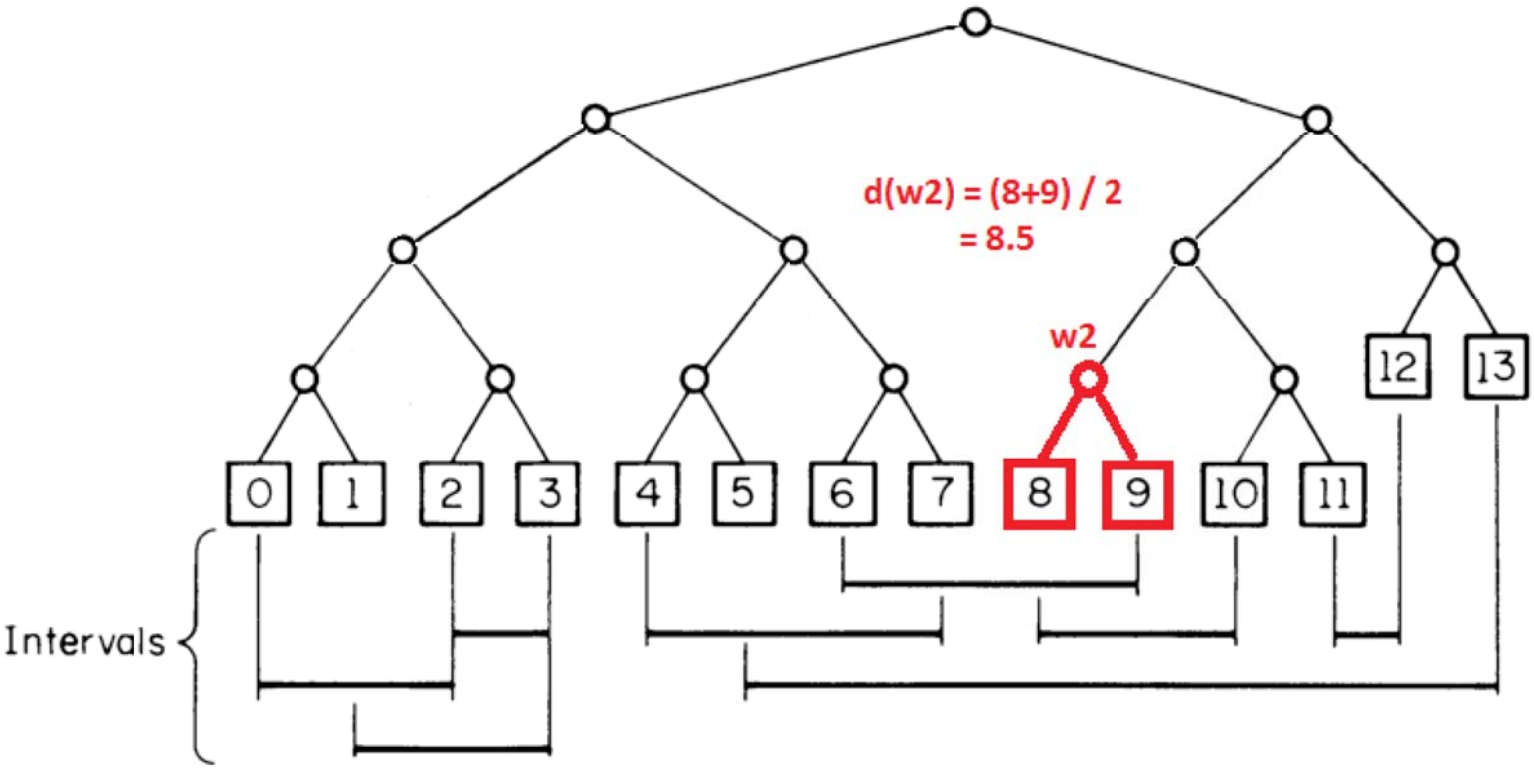
Interval tree construction



Node attribute :

- discriminant

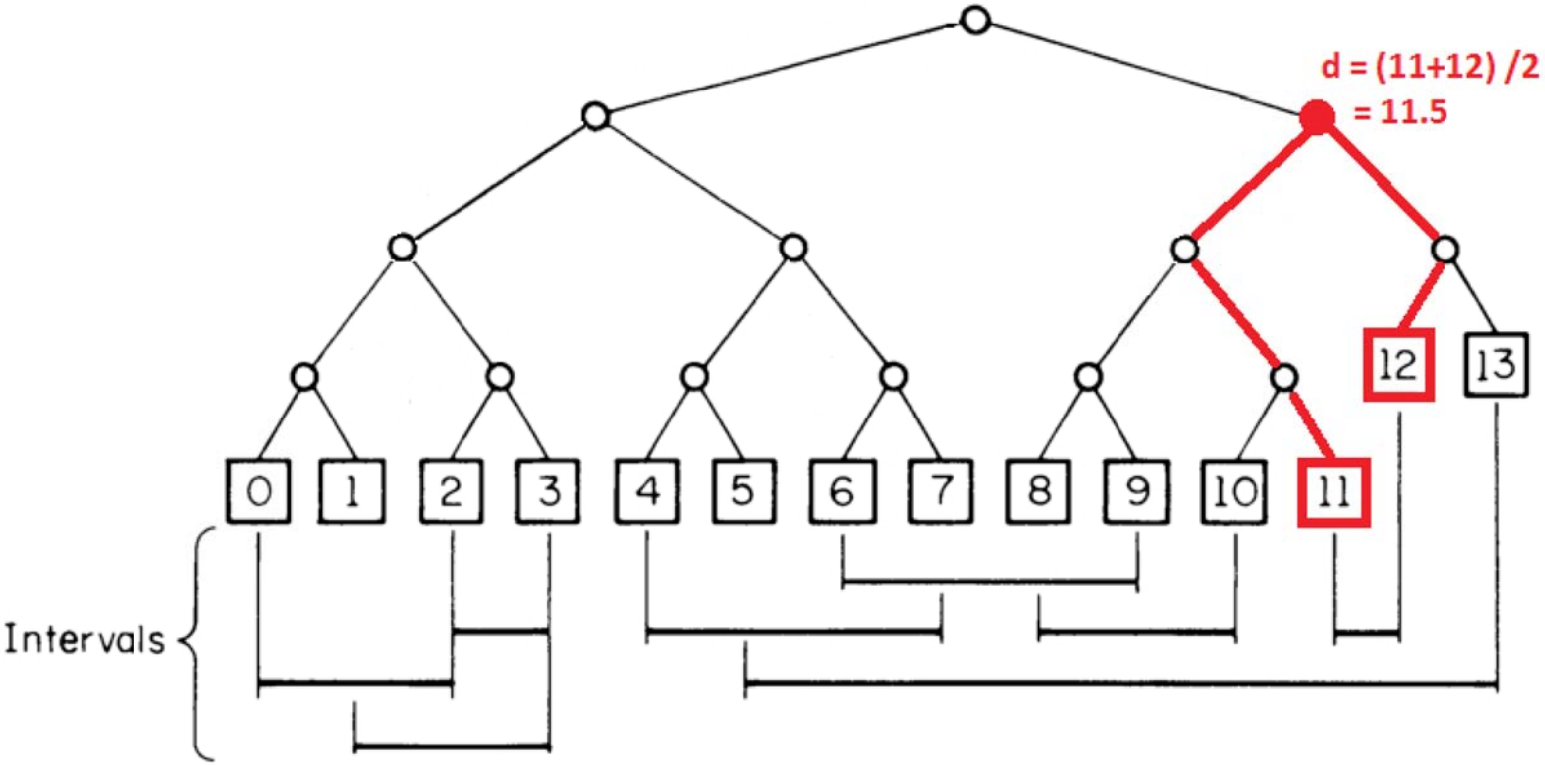
Interval tree construction



Node attribute :

- discriminant

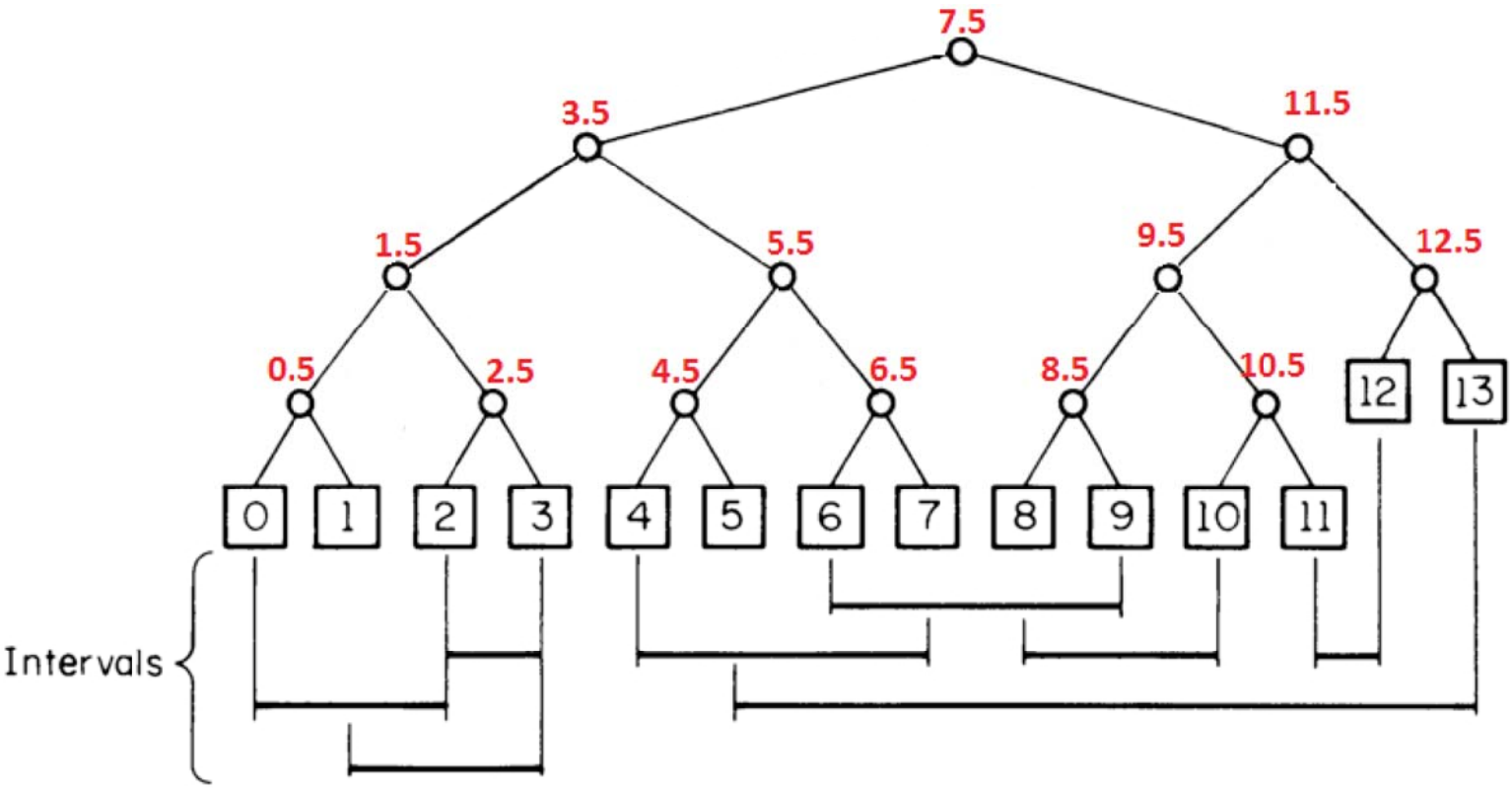
Interval tree construction



Node attribute :

- discriminant

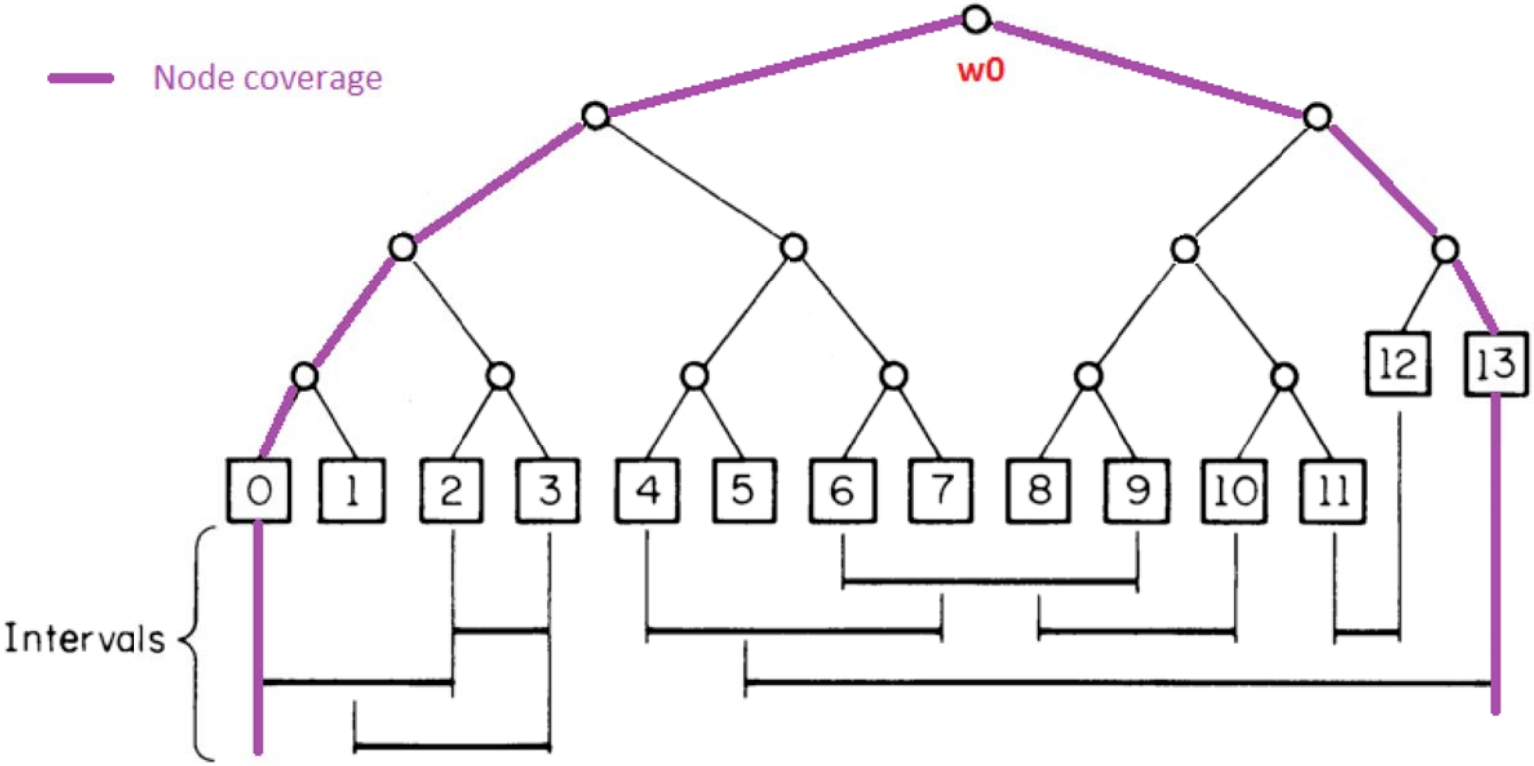
Interval tree construction



Node attribute :

- discriminant

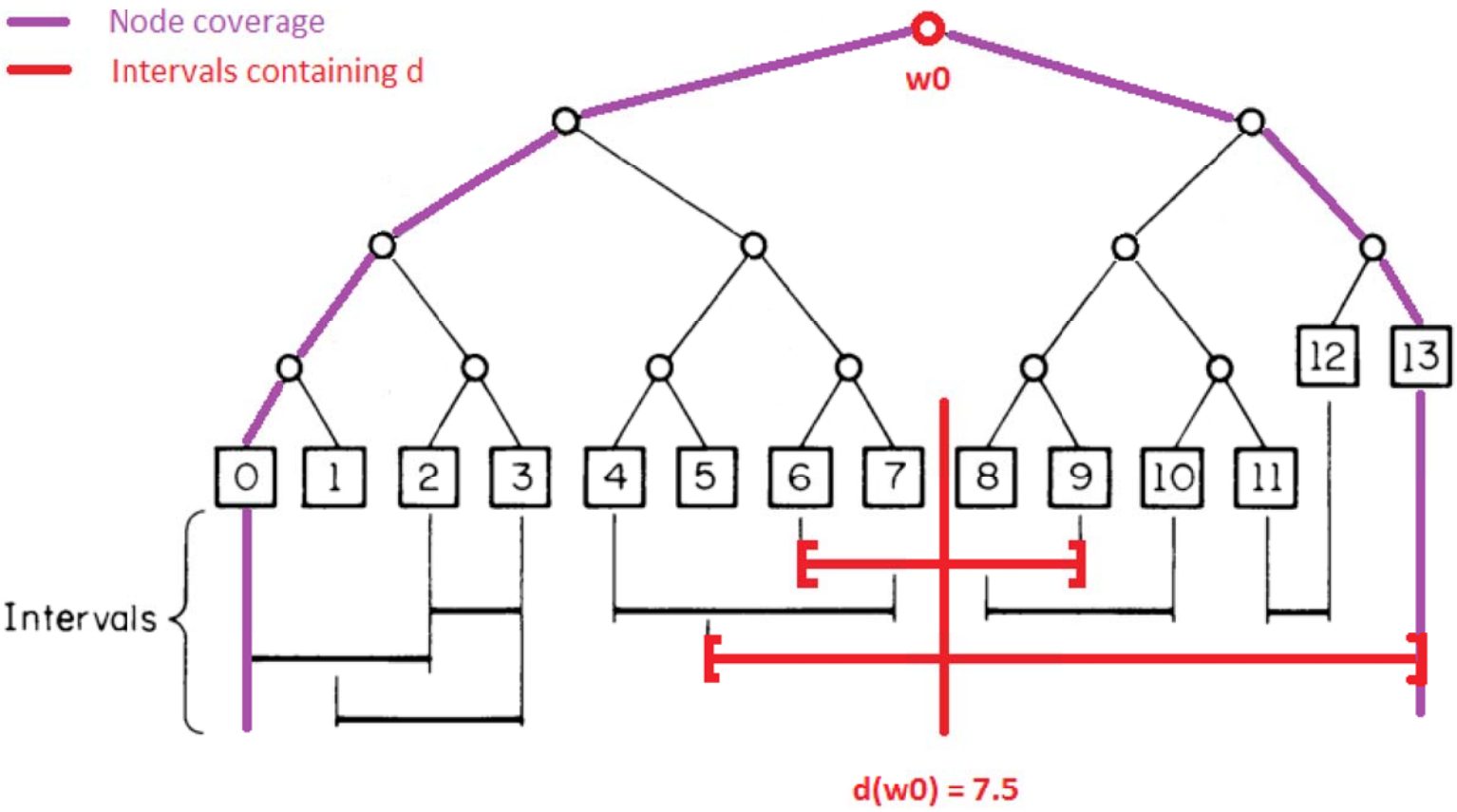
Interval tree construction



Node attributes :

- discriminant
- list of left endpoints
- list of right endpoints

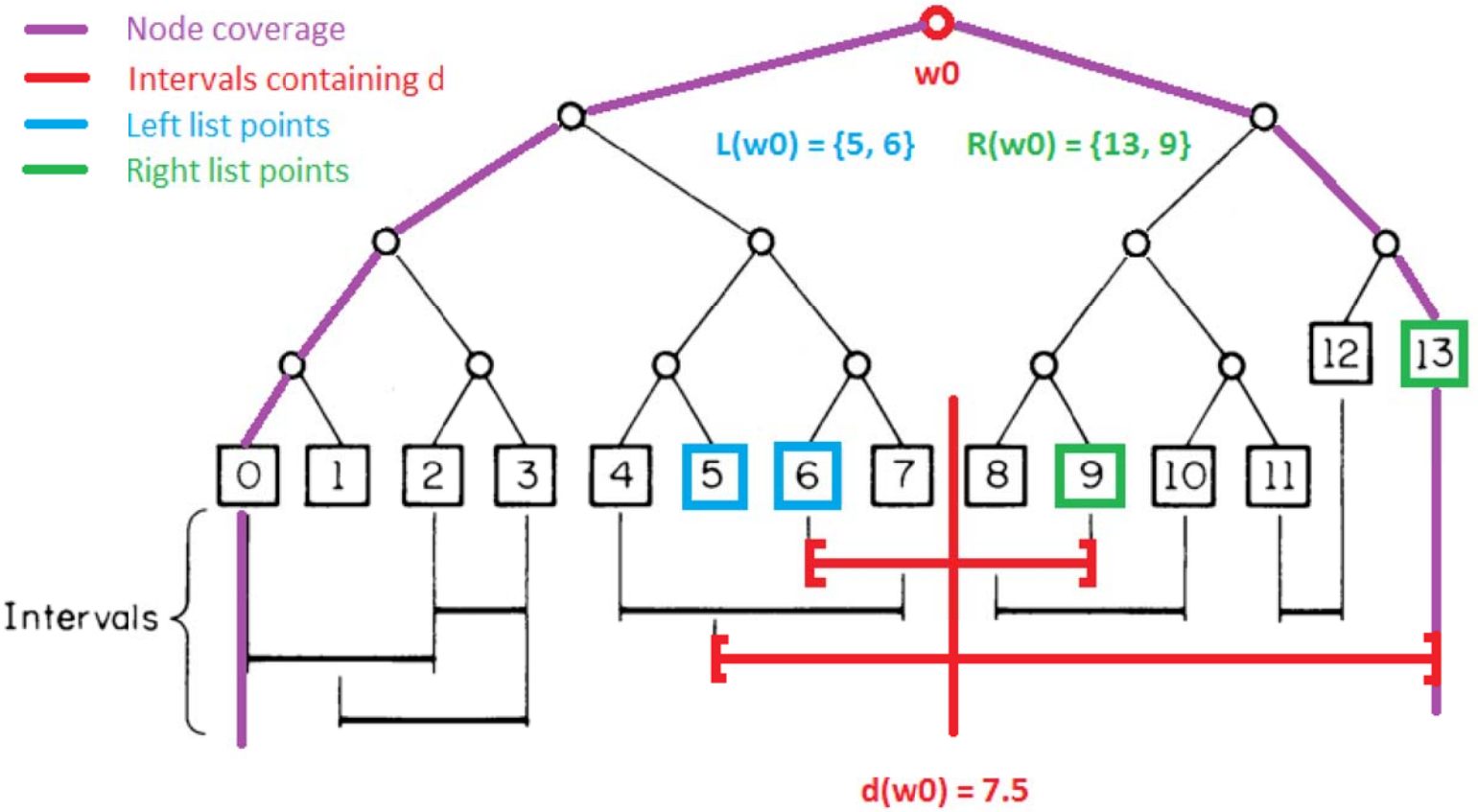
Interval tree construction



Node attributes :

- discriminant
- list of left endpoints
- list of right endpoints

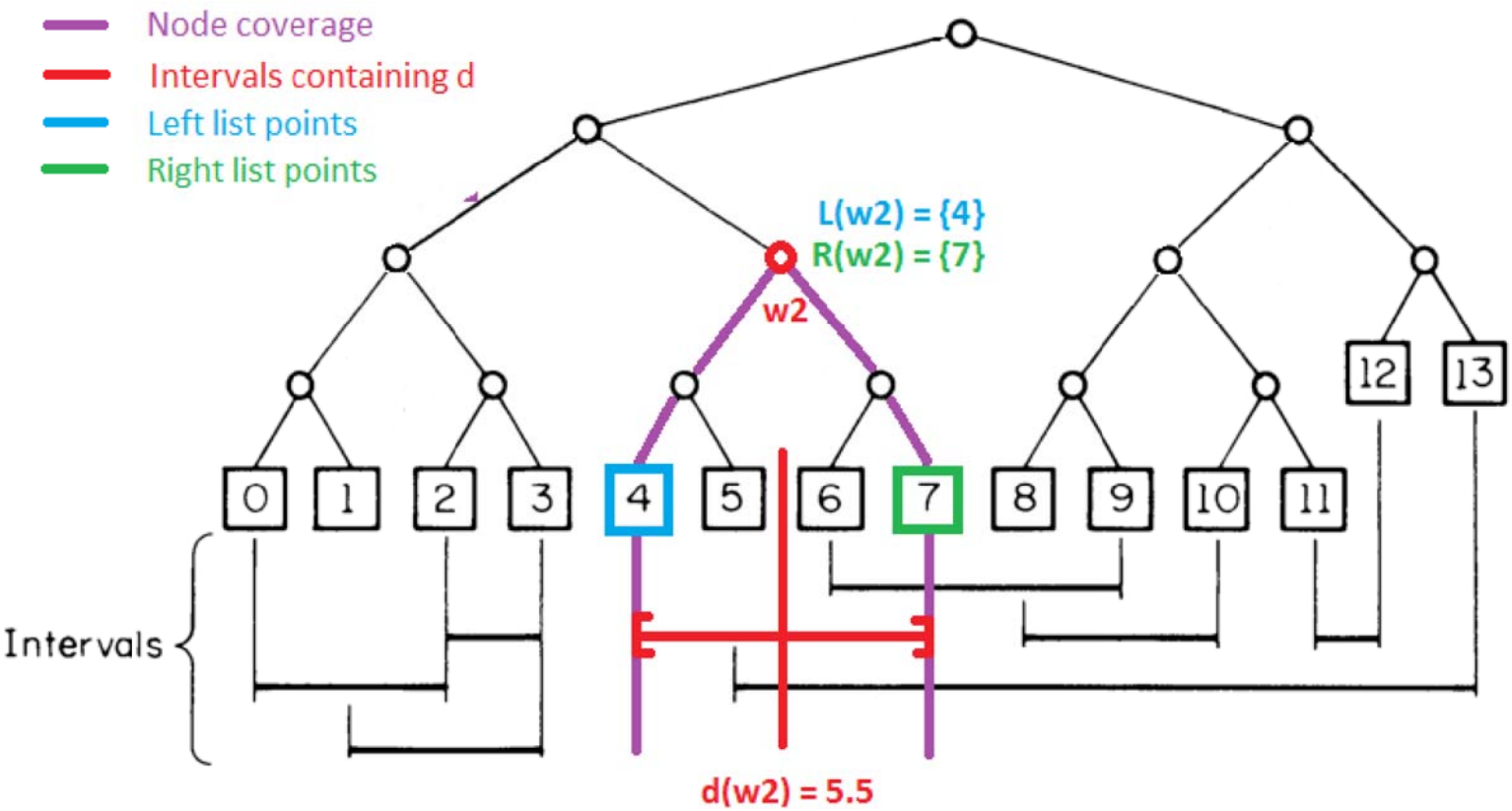
Interval tree construction



Node attributes :

- discriminant
- list of left endpoints
- list of right endpoints

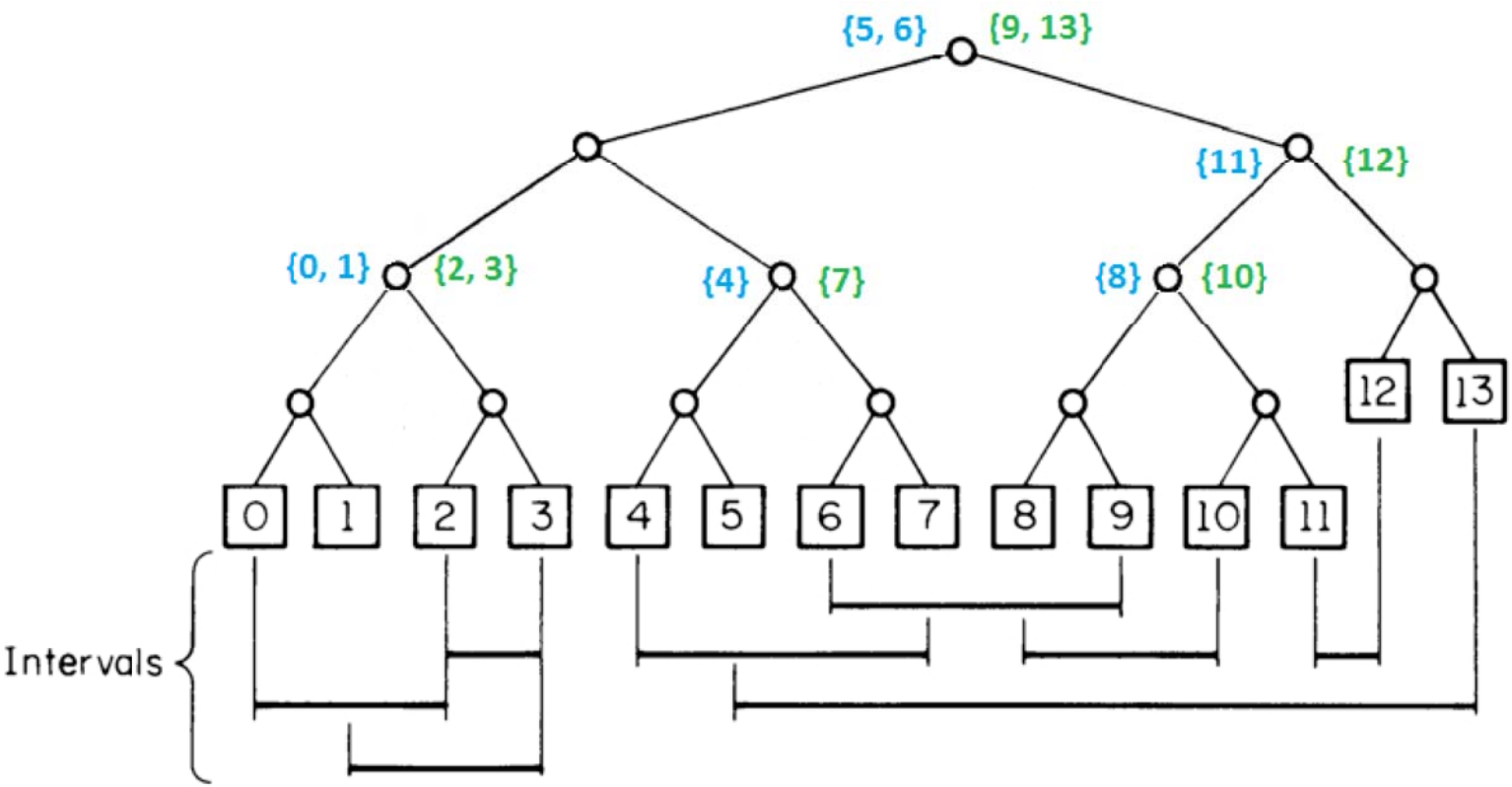
Interval tree construction



Node attributes :

- discriminant
- list of left endpoints
- list of right endpoints

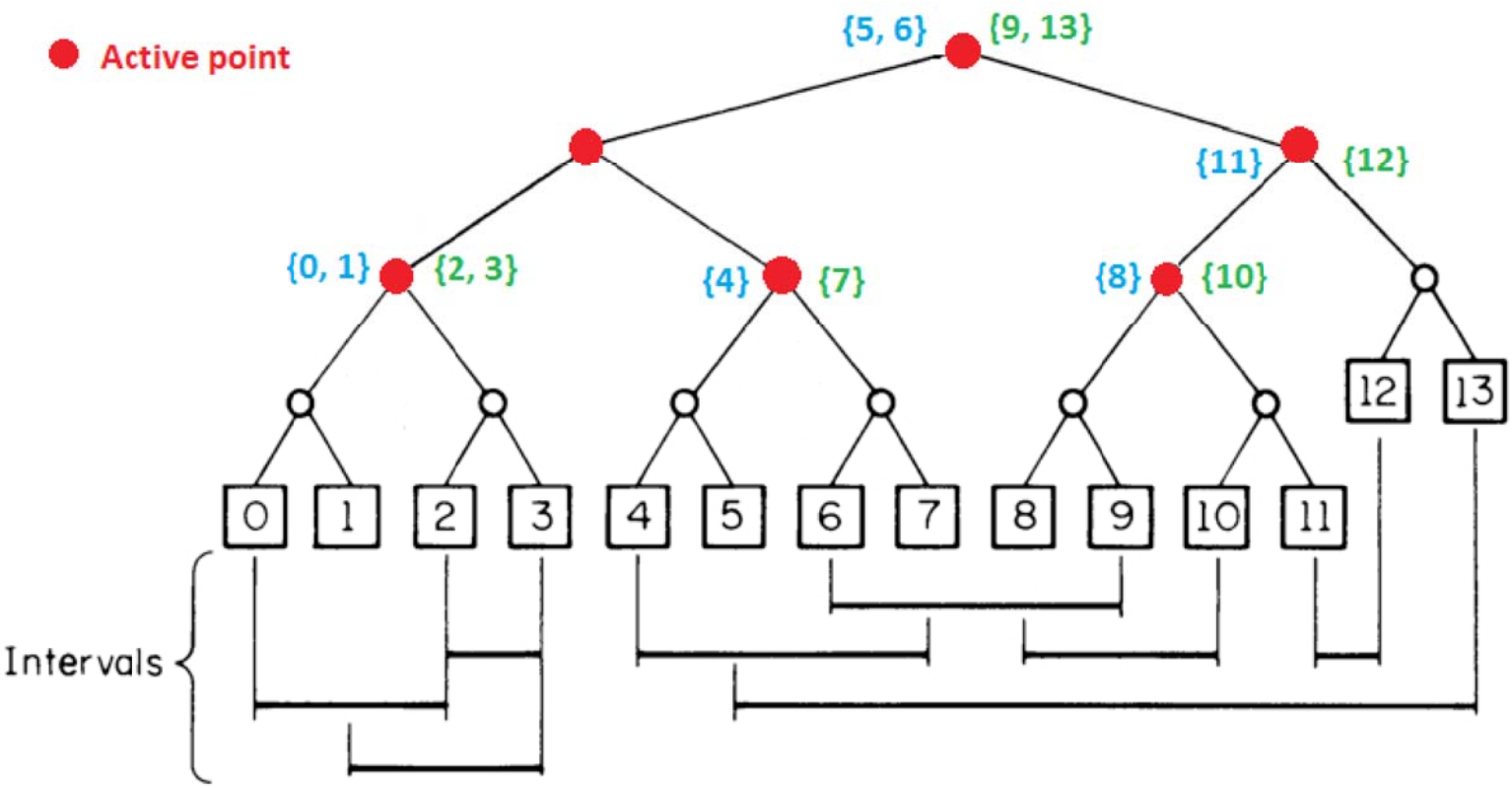
Interval tree construction



Node attributes :

- discriminant
- list of left endpoints
- list of right endpoints

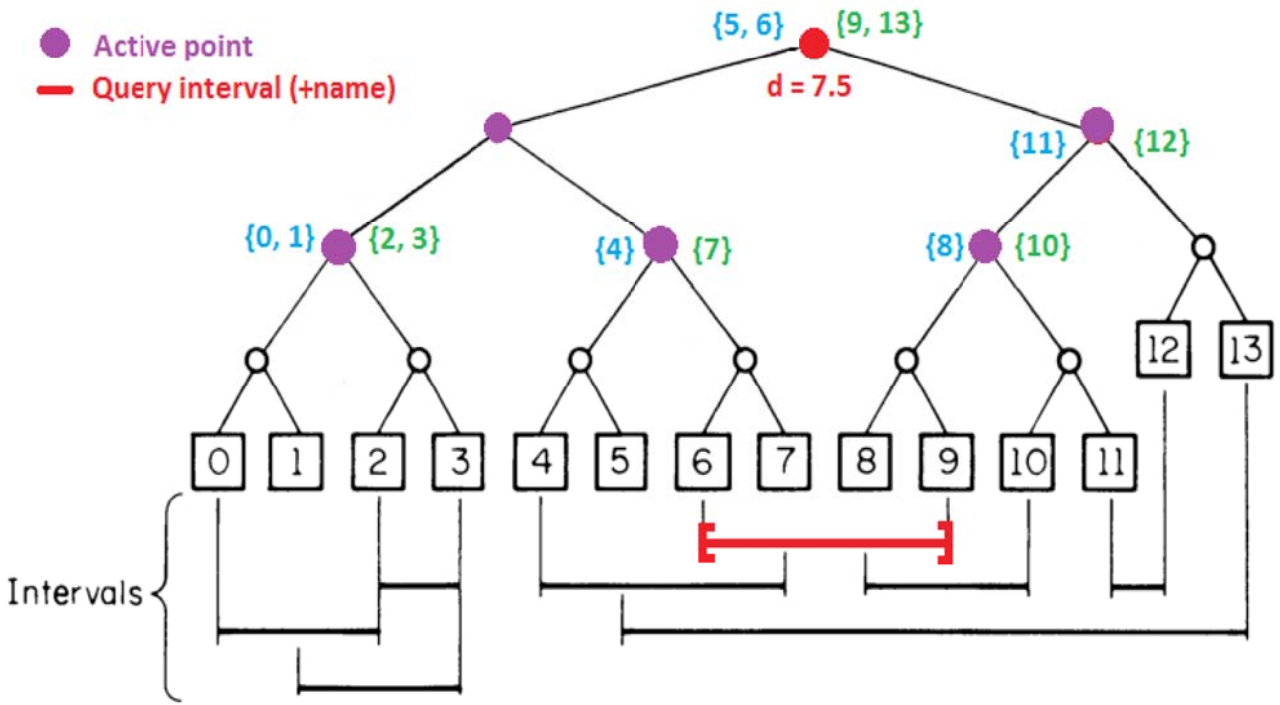
Interval tree construction



Node attributes :

- discriminant
- list of left endpoints
- list of right endpoints
- active flag

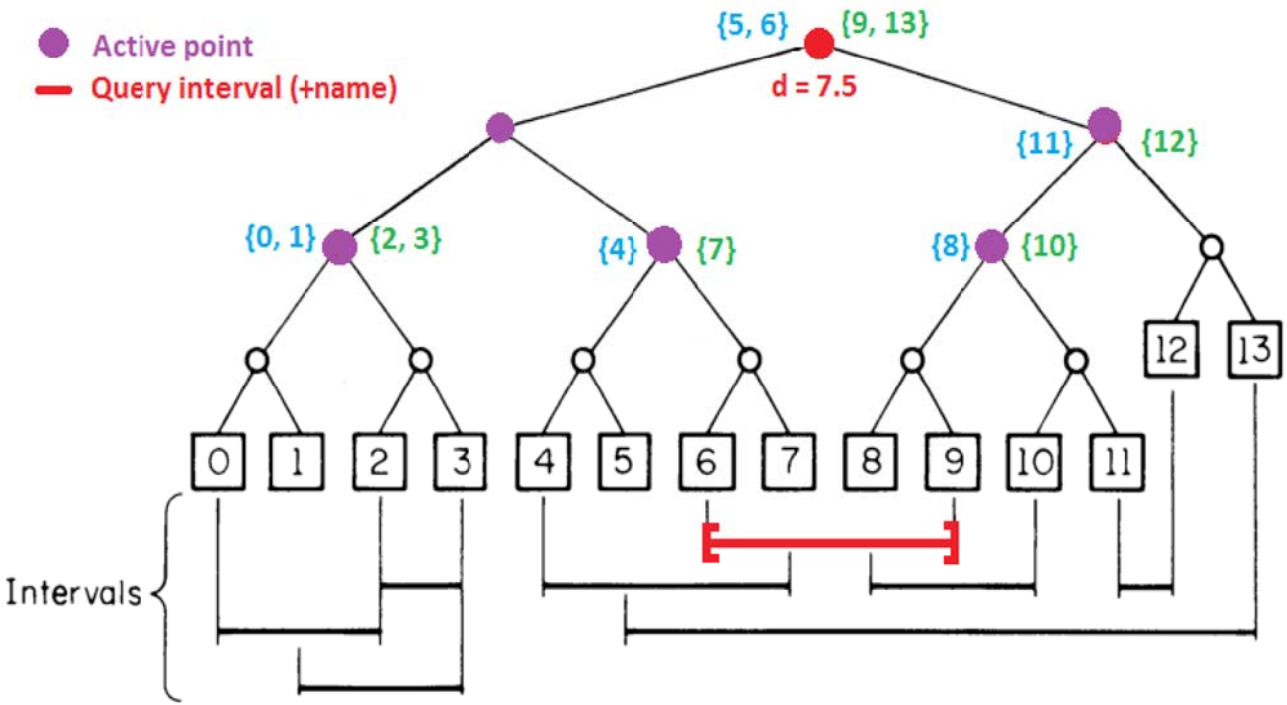
Interval tree query



Res = {}

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

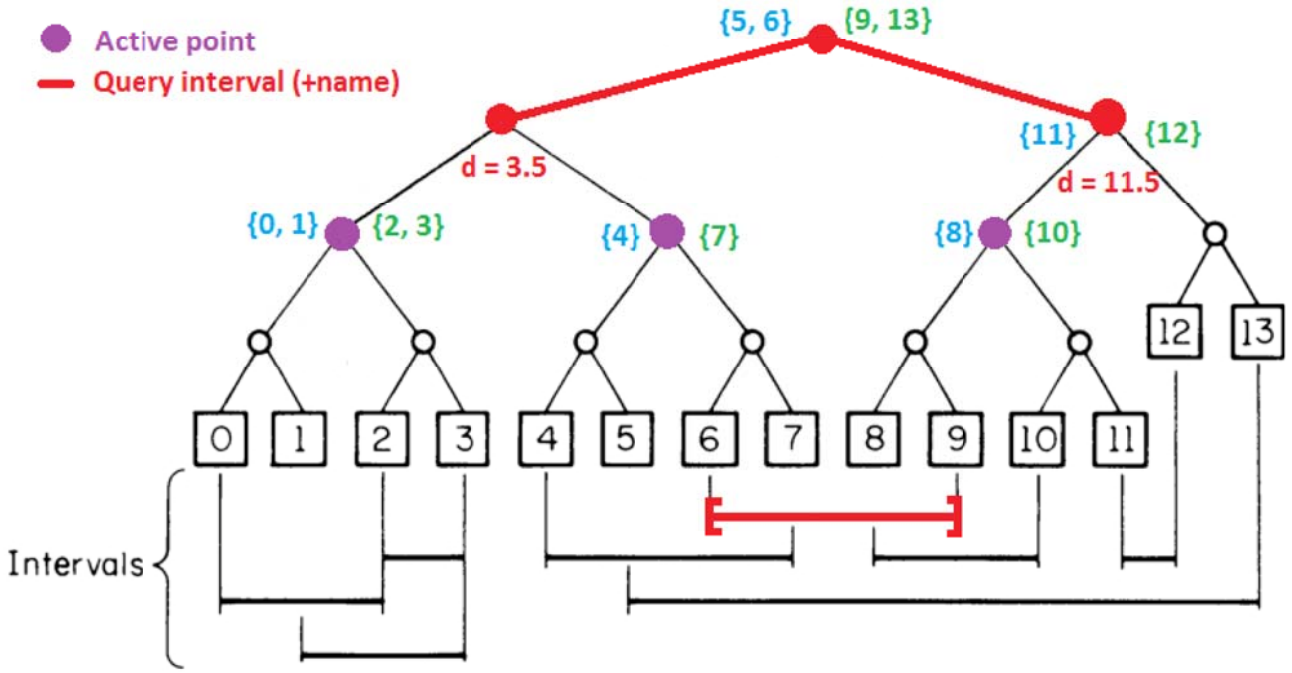
Interval tree query



Res = { [6, 9], [5, 13] }

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

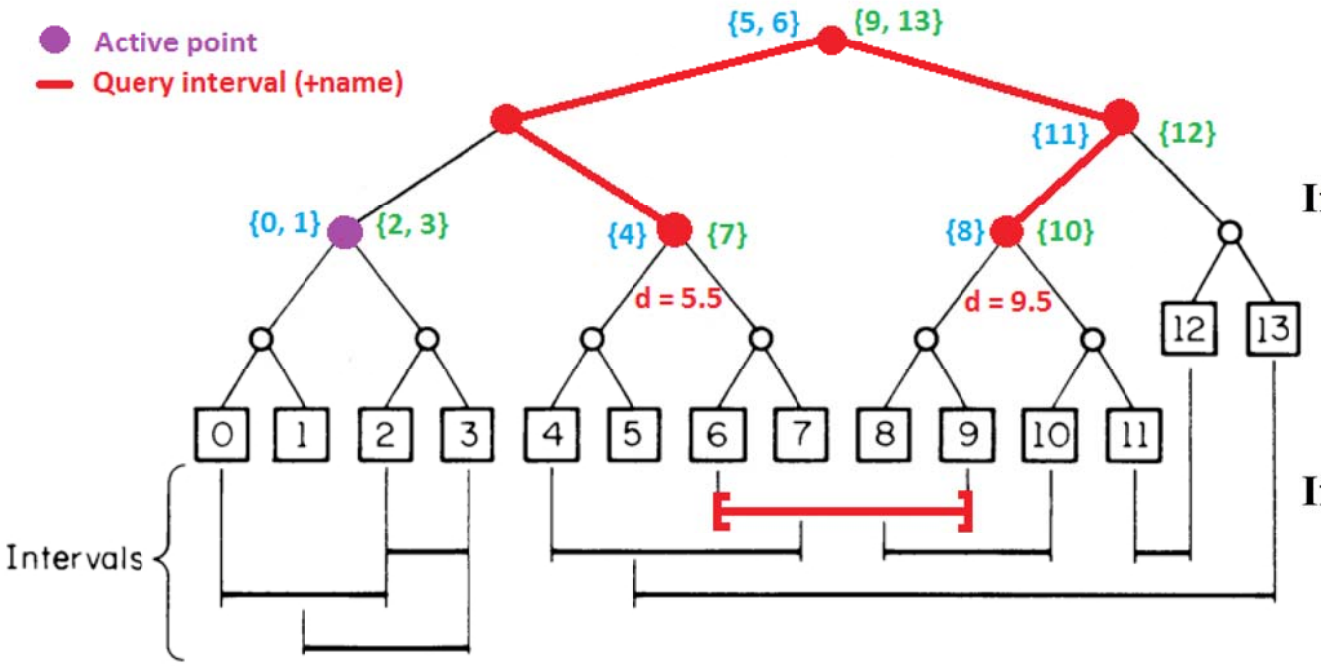
Interval tree query



Res = { [6, 9], [5, 13] }

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

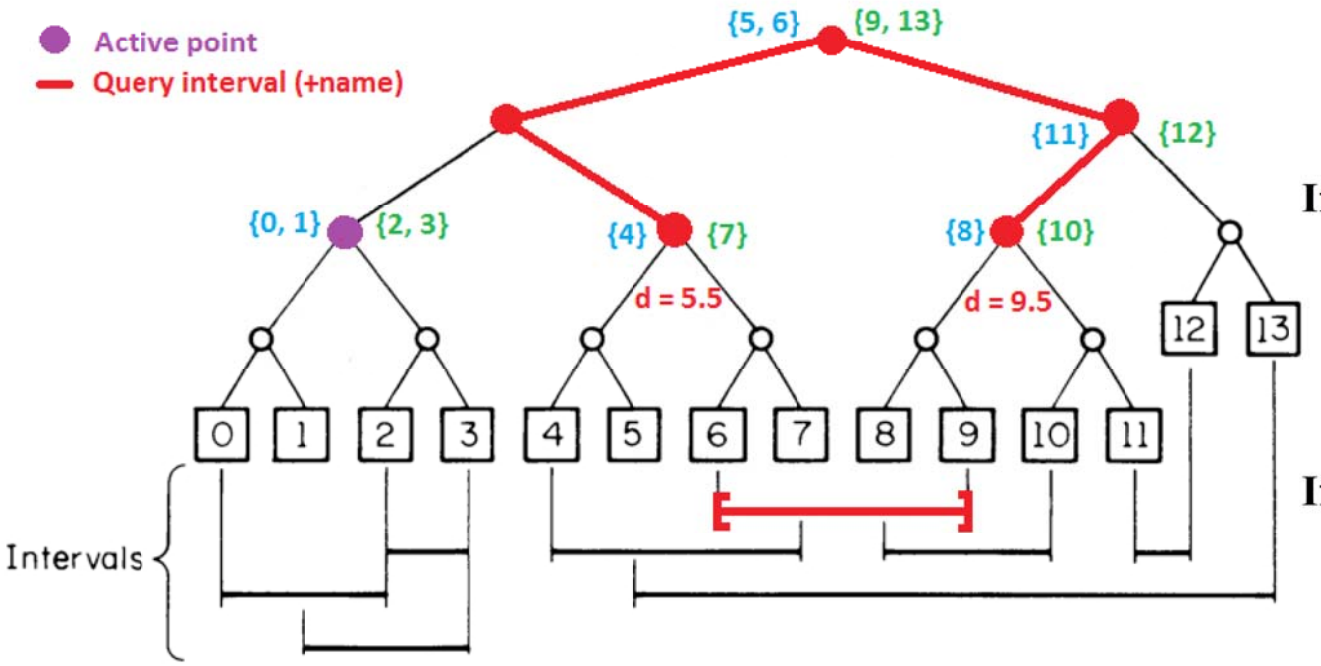
Interval tree query




```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

Res = { [6, 9], [5, 13] }

Interval tree query




```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

Res = { [6, 9], [5, 13], [4, 7], [8, 10] }

Summary

```
Res = {}
AR = {}
Tree = Build_Interval_Tree()
For each R reached by the sweep lane
        Update_AR(Tree)
        Res = Overlapping_Search(Tree, R, Res)
```

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

Complexity

Preprocessing time (interval tree construction) :

$O(N*\log(N))$

Optimal processing time :

$O(N*\log(N) + k)$

Space complexity :

$O(N)$

with N number of input rectangles and k number of output pairs.

References

[1] PREPARATA « Computational geometry: an introduction » pages 351-357

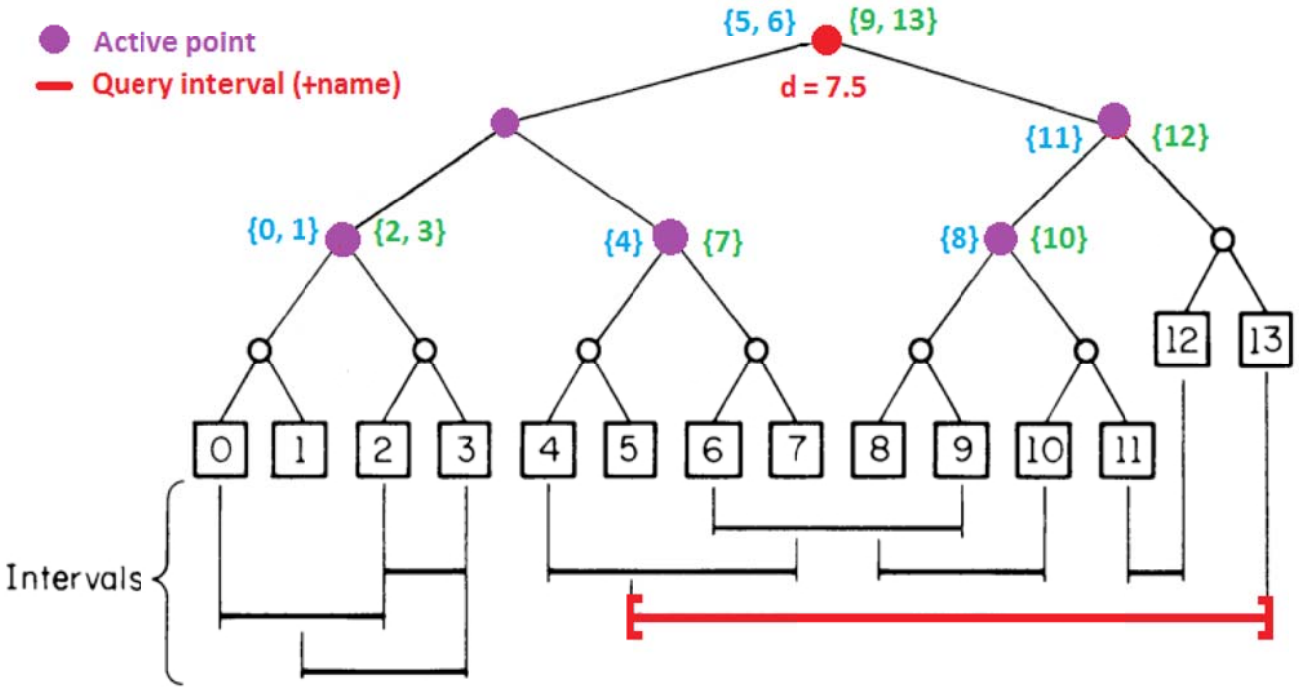
[2] D.T LEE, Academia Sinica, «Interval, segment, range and priority search trees »

[3] Antoine Vigneron, INRIA, « Computational Geometry, Lecture 6 »

Questions

Thank you !

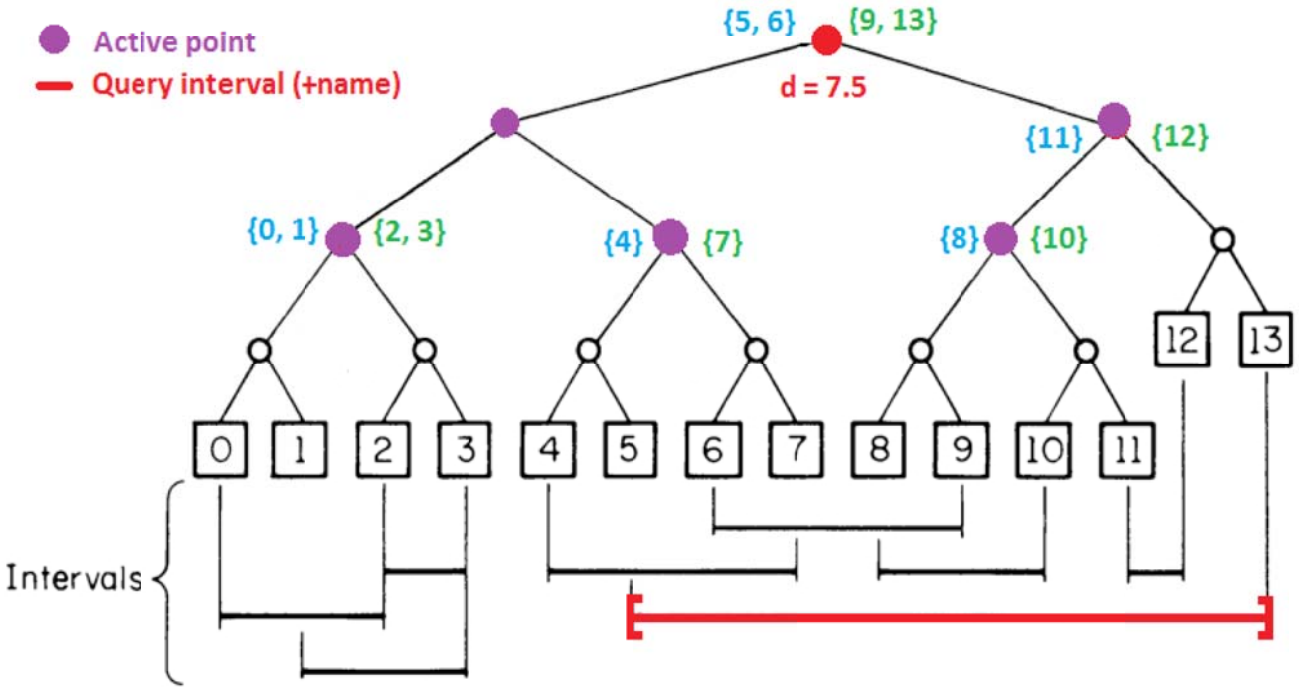
Interval tree query



Res = {}

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

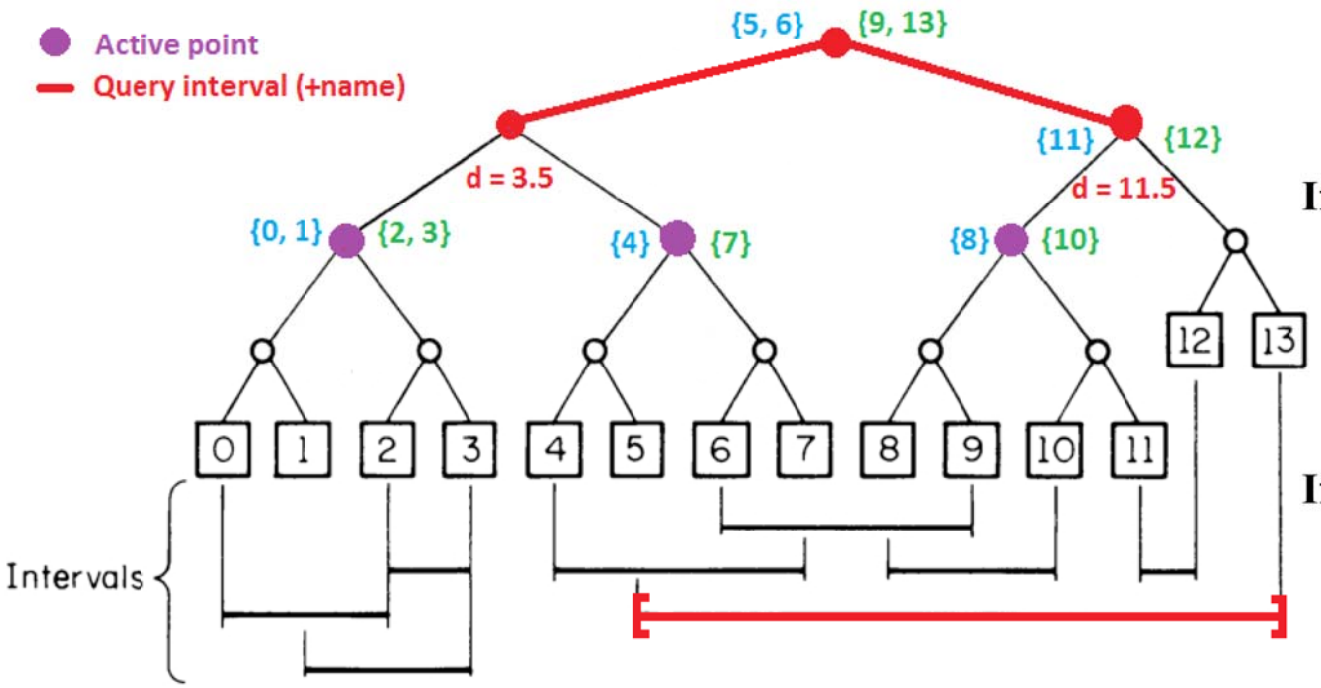
Interval tree query



Res = { [5, 13], [6, 9] }

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

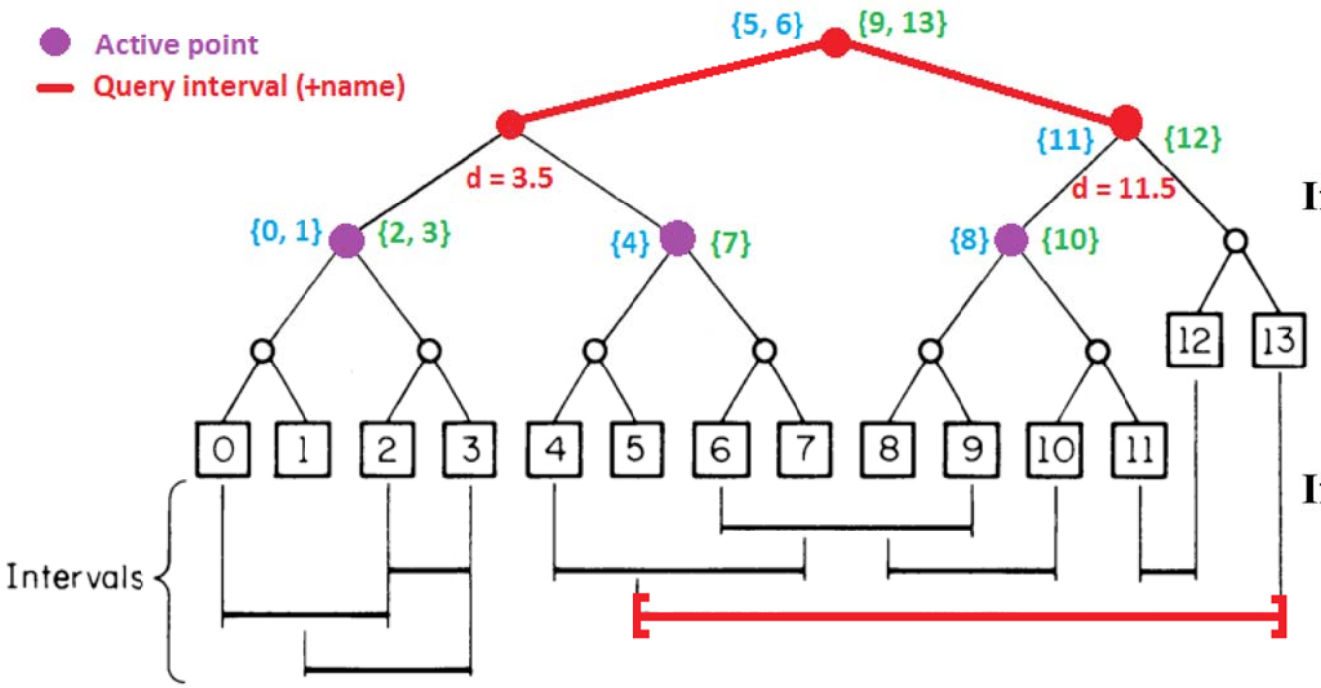
Interval tree query



Res = { [5, 13], [6, 9] }

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

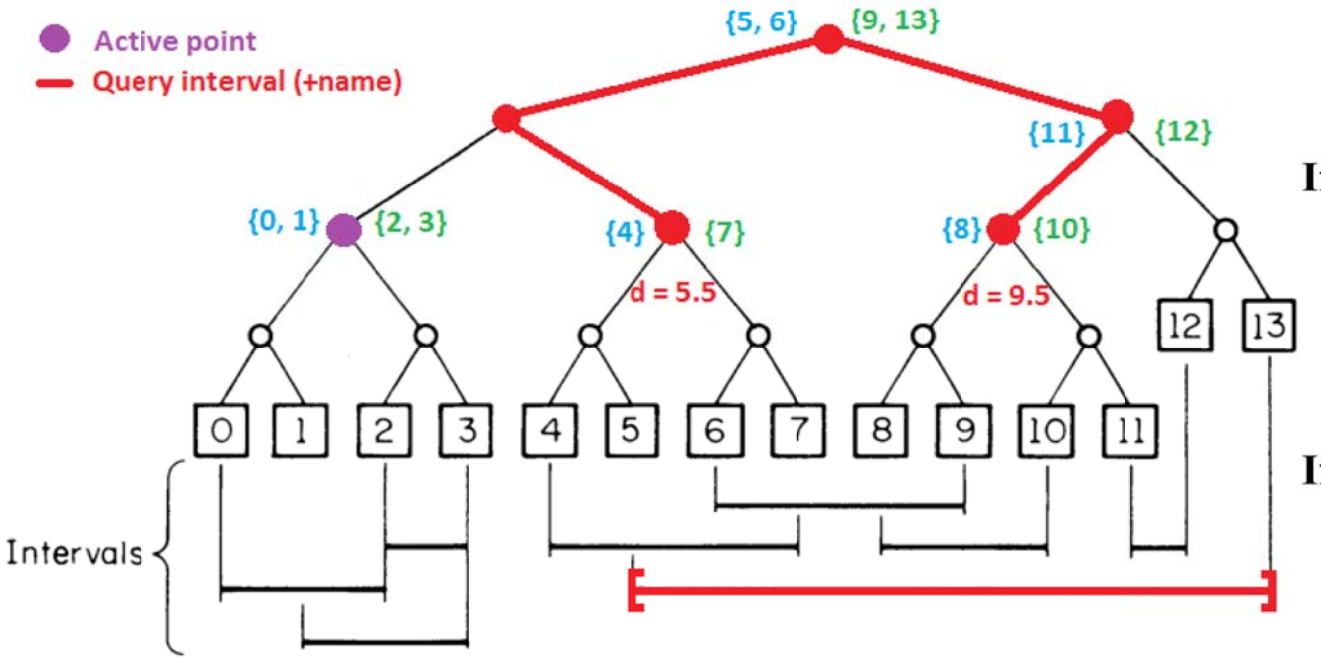
Interval tree query



Res = { [5, 13], [6, 9], [11, 12] }

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

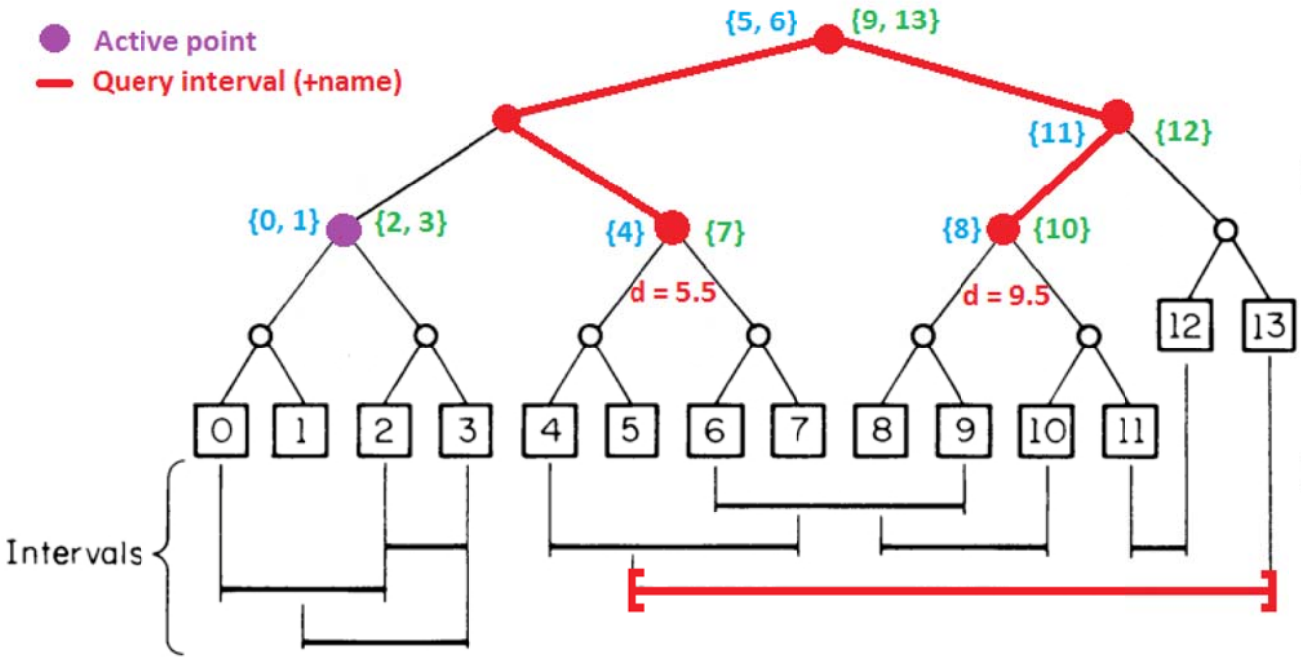
Interval tree query



Res = { [5, 13], [6, 9], [11, 12] }

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

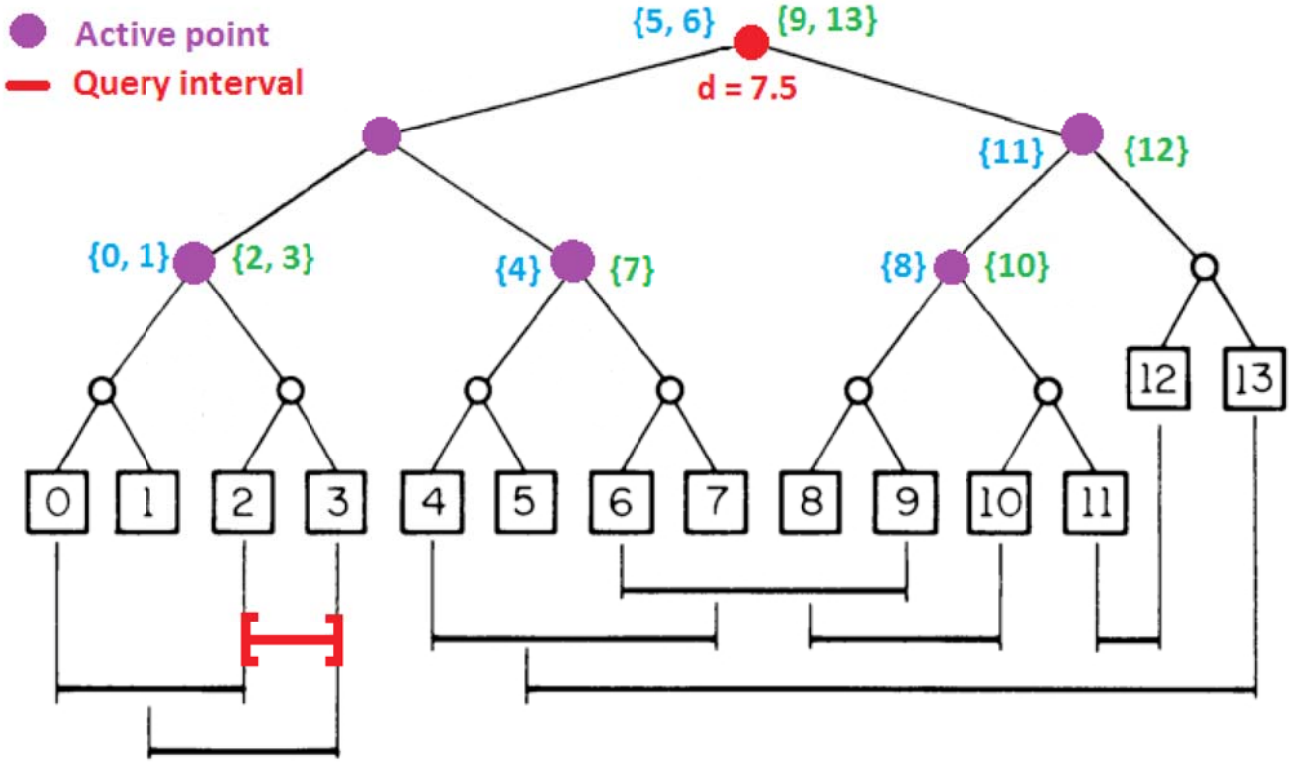
Interval tree query



Res = { [5, 13], [6, 9], [11, 12], [4, 7], [8, 10] }

```
Overlapping_Search (w, Q, Res)
    If not w.active, Return

    If Q.left < w.d < Q.right
        Res += interval(w.left_list+w.right_list)
        Res += Overlapping_Search(w.left, Q)
        Res += Overlapping_Search(w.right, Q)

    If Q.right < w.d
        For each left_endpt in w.left_list
            If left_endpt <= Q.right
                Res += interval(left_endpt)
        Res += Overlapping_Search(w.left, Q)

    If w.d < Q.left
        For each right_endpt in w.right_list
            If right_endpt >= Q.left
                Res += interval(right_endpt)
        Res += Overlapping_Search(w.right, Q)

    Return Res
```

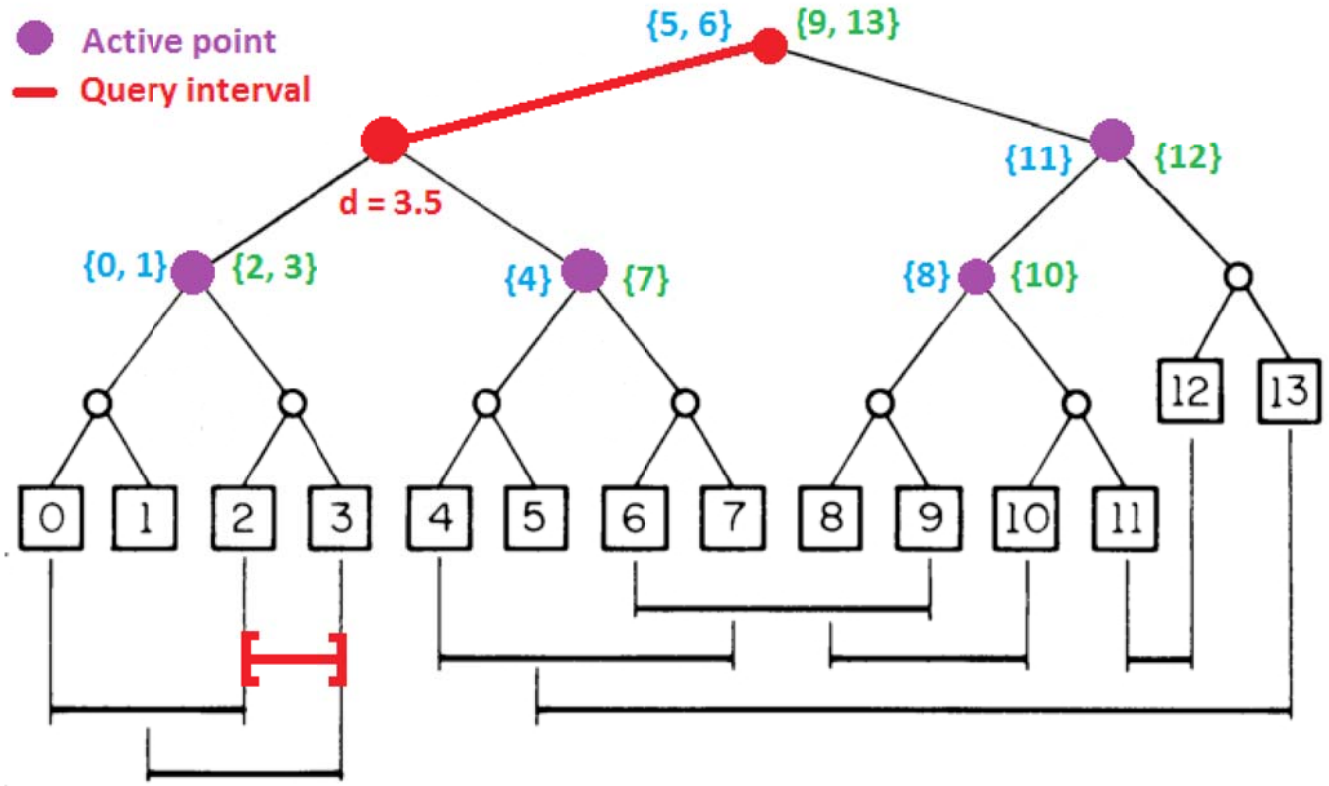
Interval tree query



Res = {}

```
Overlapping_Search (w, Q, Res)

If Q.left < w.d < Q.right
        Res = Res + w.left_list + w.right_list
        Overlapping_Search(w.left, Q, F)
        Overlapping_Search(w.right, Q, F)

If Q.right < w.d
        For each i in w.left_list
                If i >= Q.right
                        F = F + i
        Overlapping_Search(w.left, Q, F)

If w.d < Q.left
        For each i in w.right_list
                If i >= Q.left
                        F = F + i
        Overlapping_Search(w.left, Q, F)
```

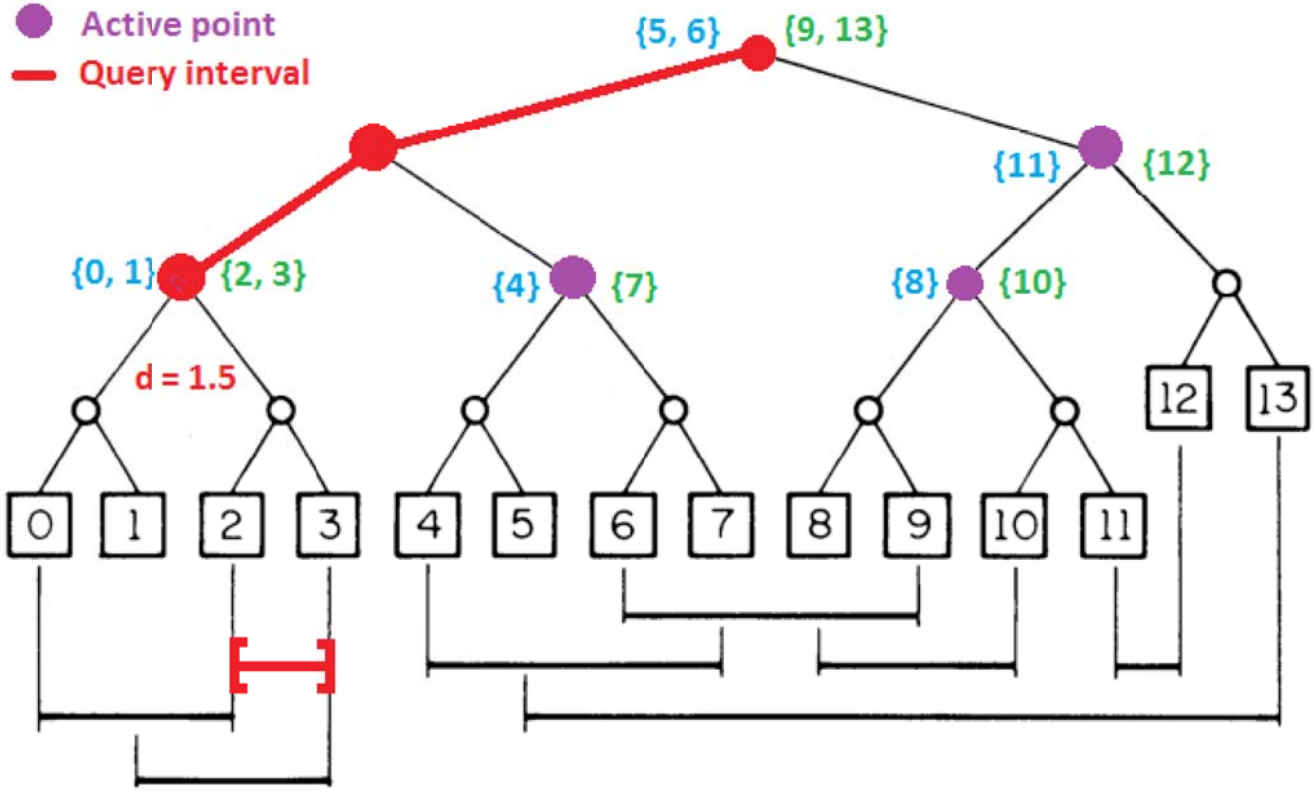
Interval tree query




```
Overlapping_Search (w, Q, Res)

If Q.left < w.d < Q.right
        Res = Res + w.left_list + w.right_list
        Overlapping_Search(w.left, Q, F)
        Overlapping_Search(w.right, Q, F)

If Q.right < w.d
        For each i in w.left_list
                If i >= Q.right
                        F = F + i
        Overlapping_Search(w.left, Q, F)

If w.d < Q.left
        For each i in w.right_list
                If i >= Q.left
                        F = F + i
        Overlapping_Search(w.left, Q, F)
```

Res = {}

Interval tree query



Res = {2, 3}

```
Overlapping_Search (w, Q, Res)

If Q.left < w.d < Q.right
        Res = Res + w.left_list + w.right_list
        Overlapping_Search(w.left, Q, F)
        Overlapping_Search(w.right, Q, F)

If Q.right < w.d
        For each i in w.left_list
                If i >= Q.right
                        F = F + i
        Overlapping_Search(w.left, Q, F)

If w.d < Q.left
        For each i in w.right_list
                If i >= Q.left
                        F = F + i
        Overlapping_Search(w.left, Q, F)
```

OI-OPPA. European Social Fund
Prague & EU: We invest in your future.