Lecture 5.2: Priority Search Trees

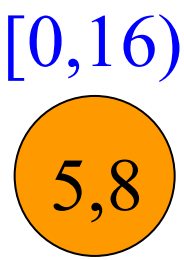
- Keys là cặp có thứ tự (x_i, y_i).
- Cây được xây dựng min theo giá trị y. (ở gốc giá trị y là nhỏ nhất)
- Các thao tác tìm kiếm chủ yếu là theo giá trị x.
- Có 2 phương án:
  - Search tree is a balanced binary search tree such as a red-black tree.
    - Red-black Priority Search Tree (RBPST)
  - Search tree is a radix search tree.
    - Radix Priority Search Tree (RPST)

Radix Priority Search Tree

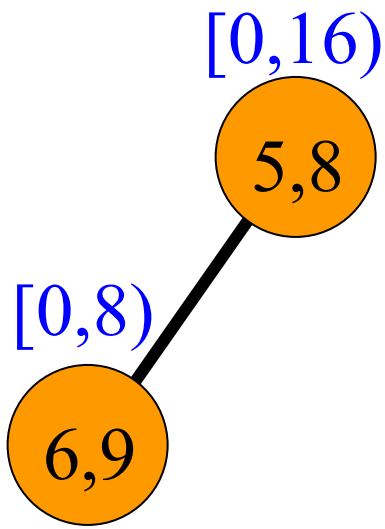
- Các giá trị của x là khác nhau và thuộc đoạn [0, k – 1].
- Giá trị y của nút w phải nhỏ hơn hoặc bằng giá trị y của các nút con của nút w. (y values define a min tree).
- Khoảng của gốc được định nghĩa là [0,k).
- Khoảng của nút w là [a,b).
  - Khoảng của nút con trái là [a, floor((a+b)/2)).
  - Khoảng của nút con phải là [floor((a+b)/2, b)).

Insert

- Ban đầu cây RPST rỗng.
- Giả sử k = 16 => Root interval is [0,16).
- Theo định nghĩa các giá trị của x thuộc [0-15]
- **Insert (5,8).** => Vì gốc rỗng nên cặp key này được chèn vào gốc



- Chèn tiếp: Insert (6,9).
  - (5,8) vẫn ở lại nút gốc vì 8 < 9.
  - (6,9) được chèn vào nút con trái, vì giá trị x=6 thuộc khoảng của nút con bên trái.

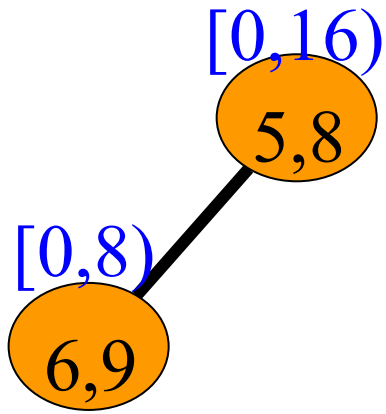


Insert

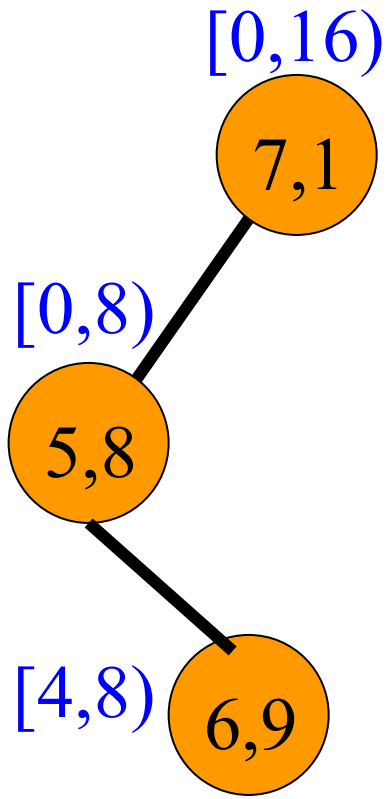
- Qui tắc chèn cặp khóa (x,y) vào RPST như sau:
  - Giá x phải thuộc khoảng của nút đó.
  - Giá trị y phải lớn hơn giá trị y ở nút cha.

Insert

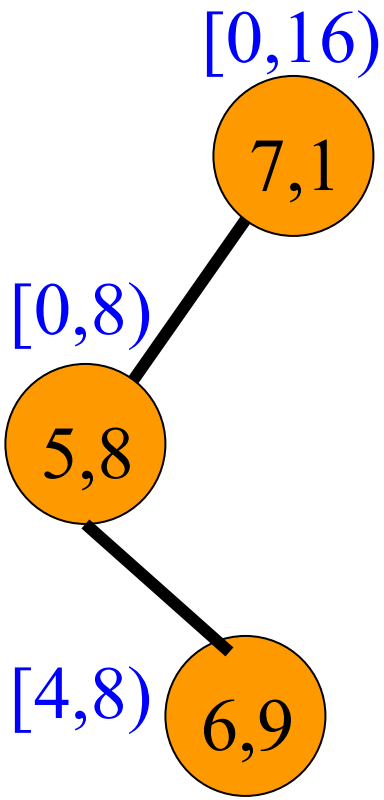
- Insert (7,1).
- (7,1) sẽ thay thế cặp khóa ở gốc, vì 1 < 8.



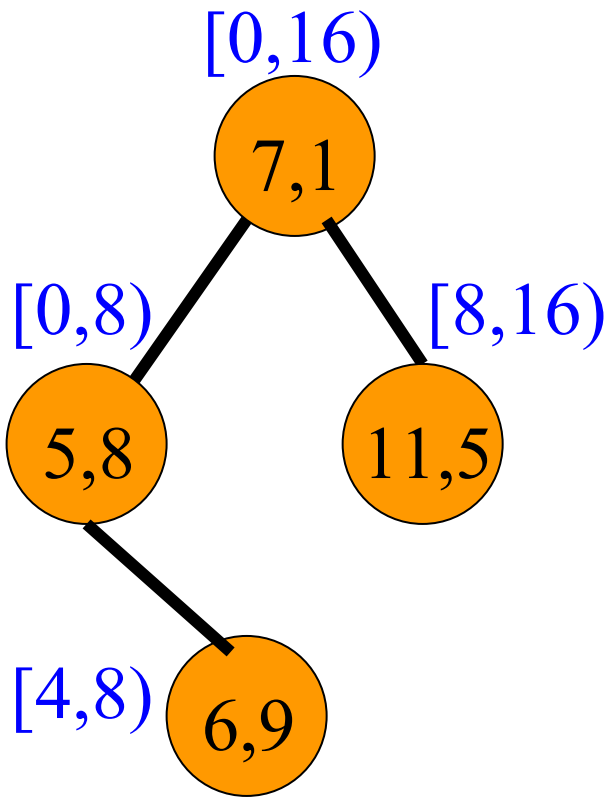
- (5,8) được đẩy xuống con trái,vì 5 thuộc khoảng của nút con trái.
- (5,8) sẽ thay thế vào vị trí của (6,9), vì 8 < 9.
- (6,9) được đẩy xuống nút con phải của (5,8), vì 6 thuộc khoảng của nút con phải và 8 < 9.



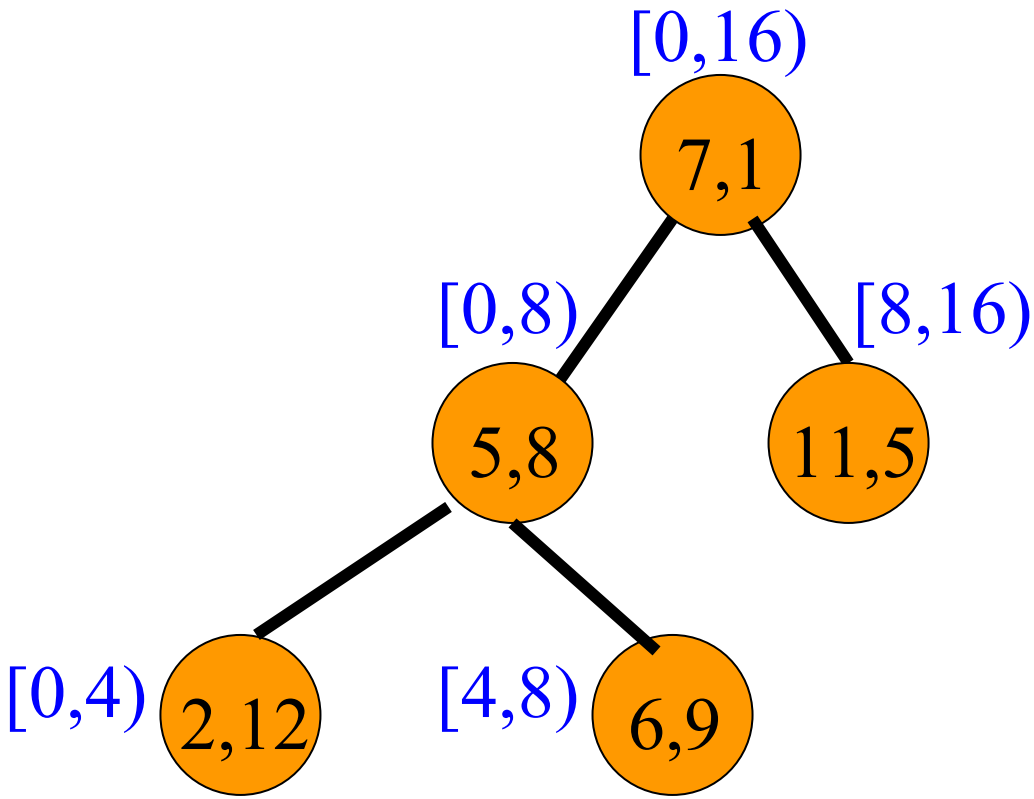
Insert



- Insert (11,5).

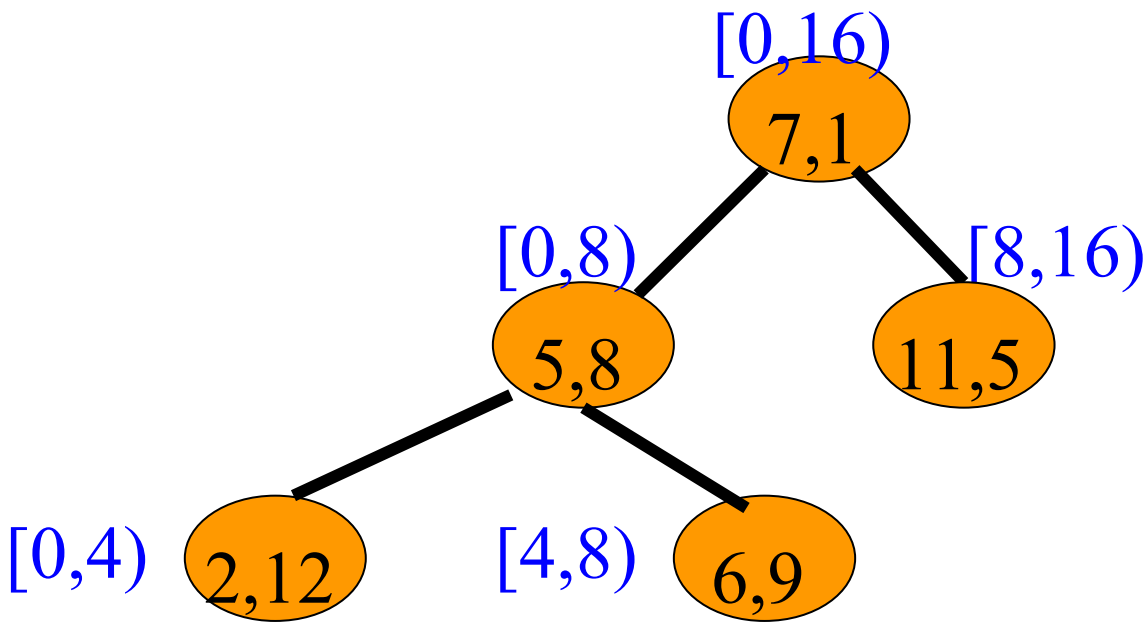


Properties



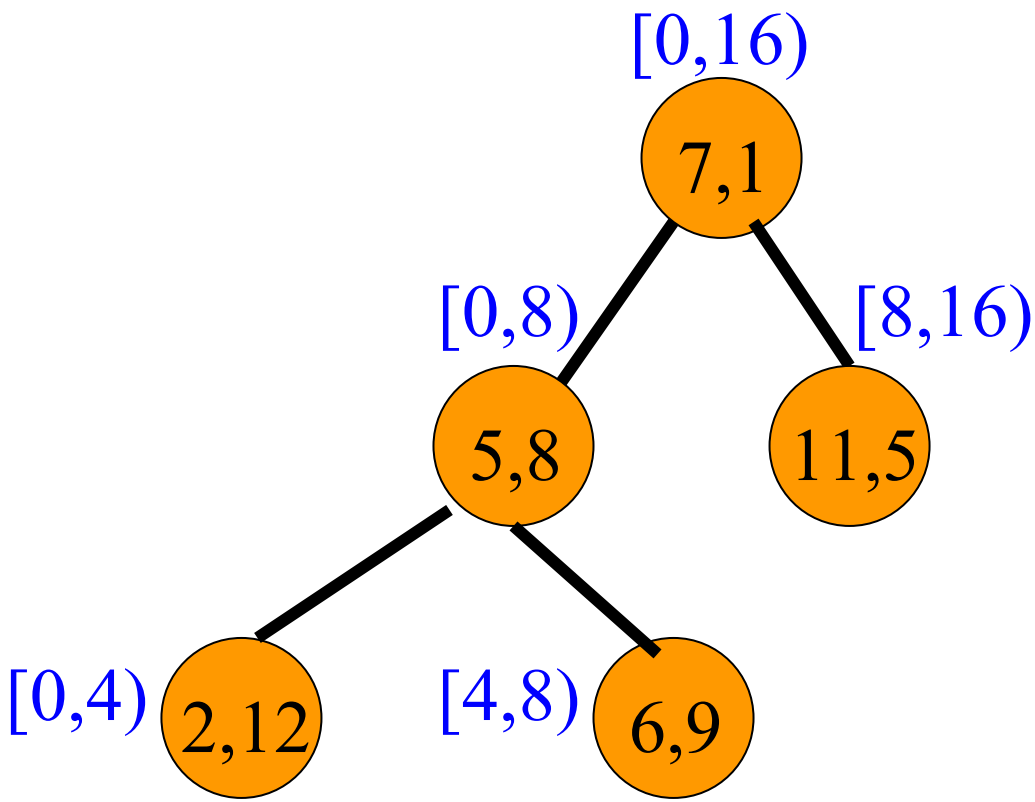
- Chiều cao H của cây = O(log k).
- Thời gian chèn = O(log k).

Search



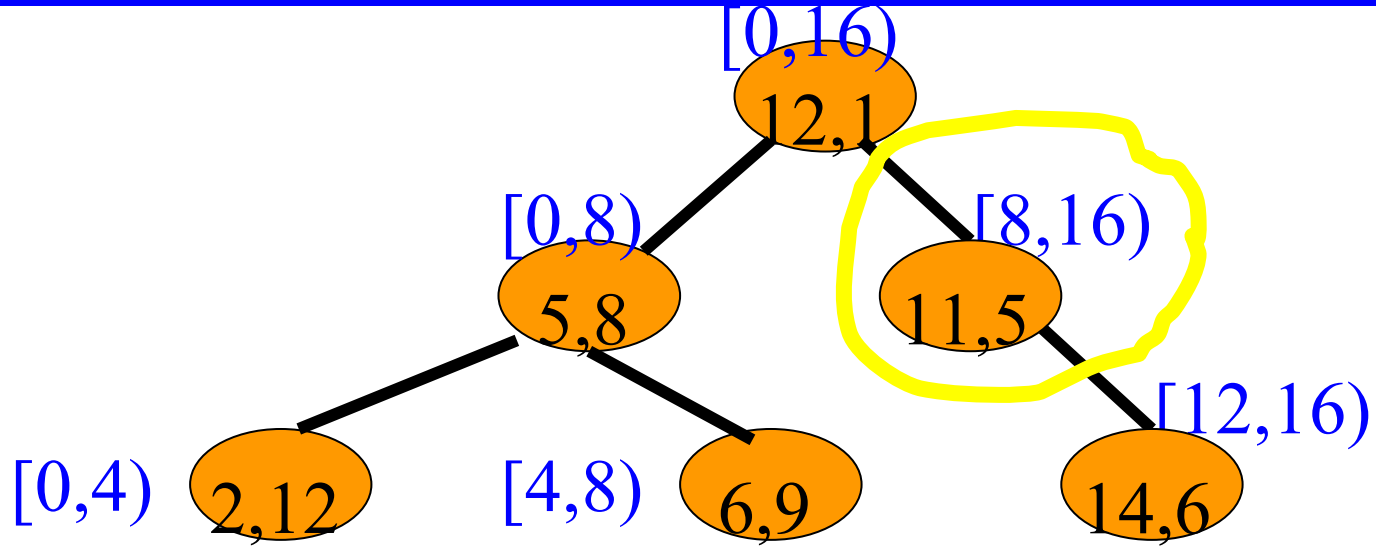
- Search time is O(log k).

Delete



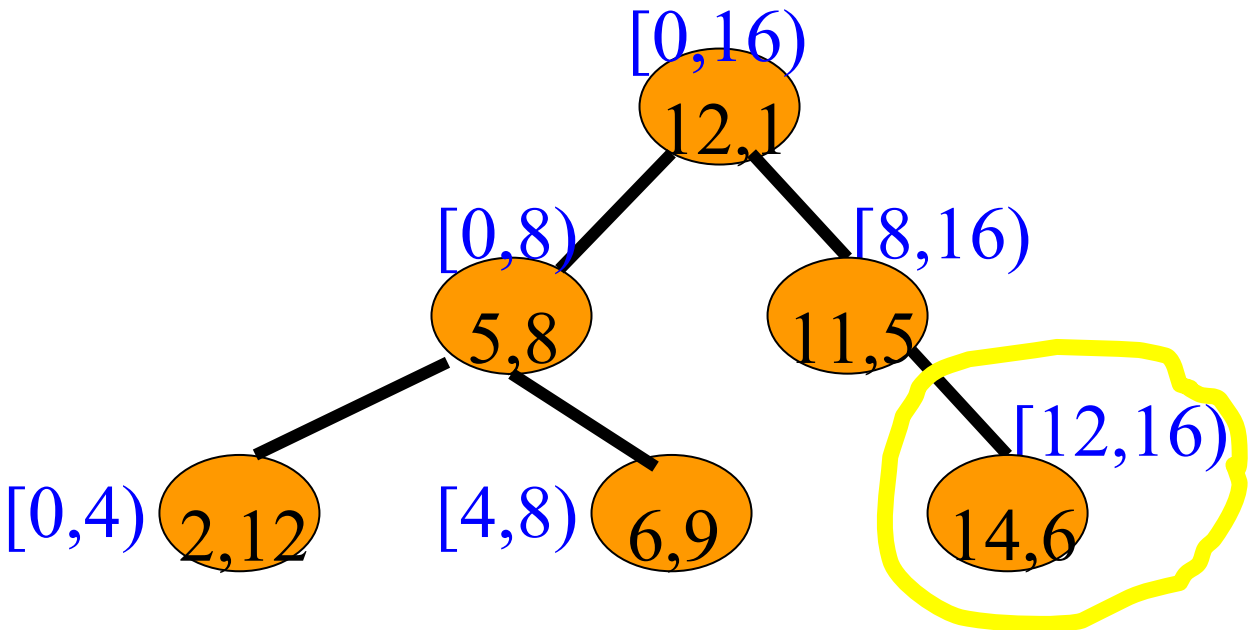
- Similar to delete min of min heap.
- Delete time is O(log k).

minXinRectangle(x_L,x_R,y_T)



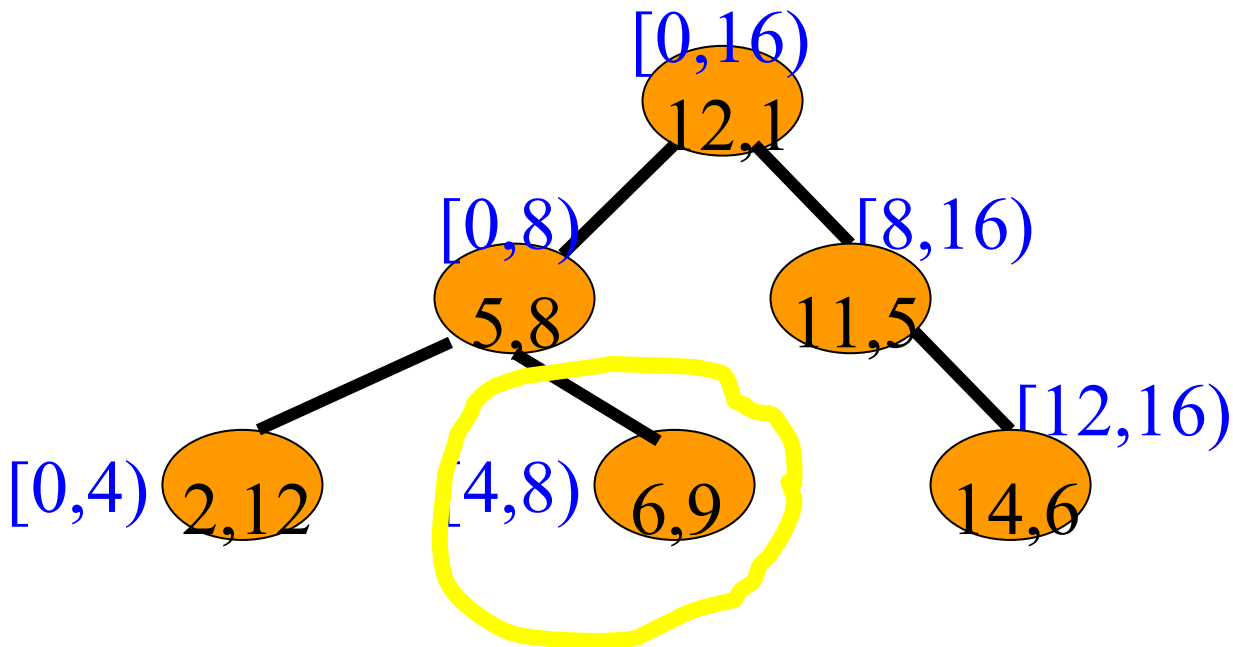
- minXinRectangle(6,12,7). Tìm phần tử có x nhỏ nhất trong HCN: 6 <= x <= 12 and y <= 7.
- Từ gốc: trong khi y<=7 và khoảng của nút có thể chứa giá trị x thuộc đoạn [6,12] thì cứ đi tiếp xuống để tìm x min thuộc 6 <= x <= 12.
- Time is O(log k).

maxXinRectangle(x_L,x_R,y_T)



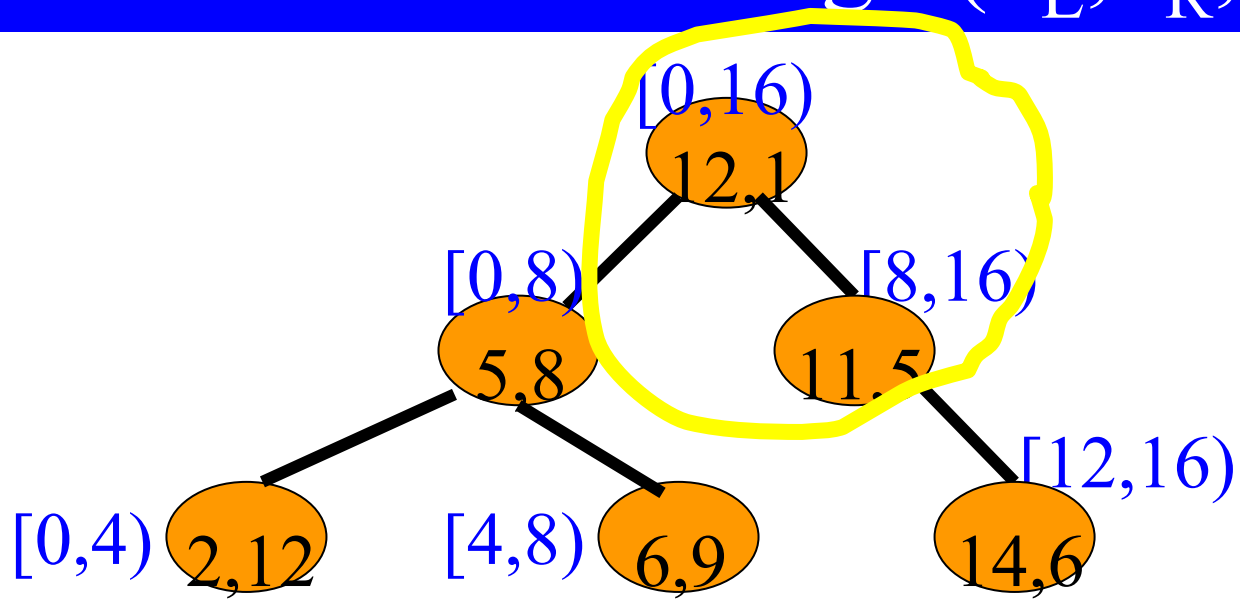
- maxXinRectangle(6,15,7).
- Từ gốc: trong khi y<=7 và khoảng của nút có thể chứa giá trị x thuộc đoạn [6,15] thì cứ đi tiếp xuống để tìm x max thuộc 6 <= x <= 12.
- Time is O(log k).

minYinXrange(x_L,x_R)



- minYinXrange(6,10).
- Từ gốc: trong khi khoảng của nút có thể chứa các giá trị x thuộc đoạn [6,10] thì cứ đi tiếp xuống để tìm y min.
- Như vậy min y=9 với x=6
- Time is O(log k).

enumerateRectangle(x_L,x_R,y_T)



- enumerateRectangle(6,12,8).
- Từ gốc: trong khi khoảng của nút có thể chứa các giá trị x thuộc đoạn [6,12] thì cứ đi tiếp xuống để tìm các giá trị y<=8.
- Time is O(log k + s), where s is #points in rectangle.