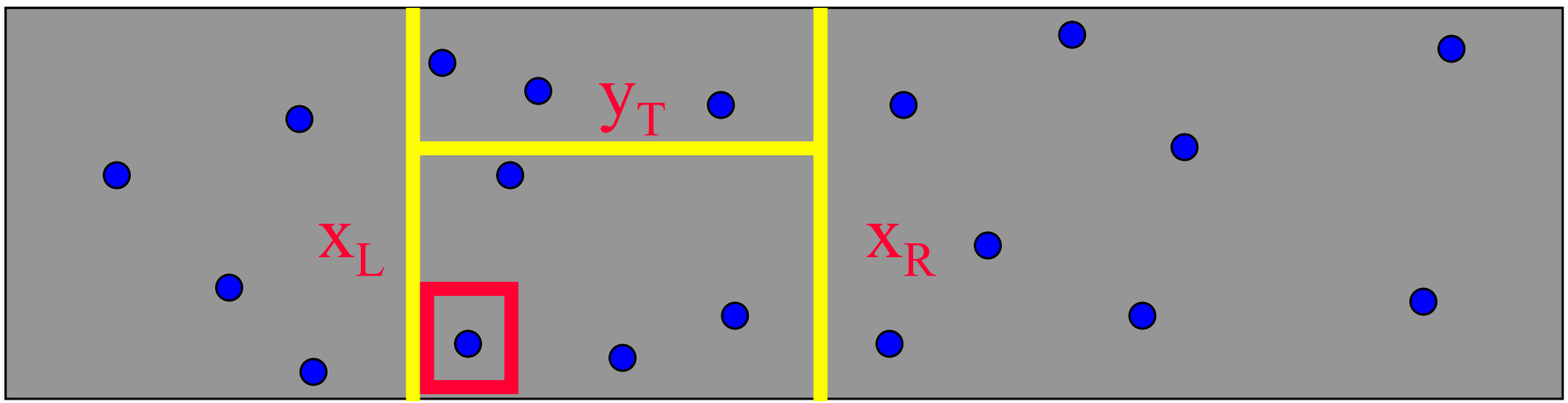
Lecture 5.1: Priority Search Trees

Cây tìm kiếm ưu tiên được định nghĩa như sau:

- Keys là cặp khóa (x_i, y_i).
- Basic operations.
  - get(x,y) … return element whose key is (x,y).
  - delete(x,y) … delete and return element whose key is (x,y).
  - insert(x,y,e) … insert element e, whose key is (x,y).
- Rectangle operations – các phép toán trên HCN

minXinRectangle(x_L,x_R,y_{Top})

- Return phần tử có tọa độ x nhỏ nhất trong HCN được định nghĩa bằng 4 đường thẳng,
- $x = x_L$, $x = x_R$;
- $y = 0$, $y = y_T$ ($x_L <= x_R$, $0 <= y_T$)

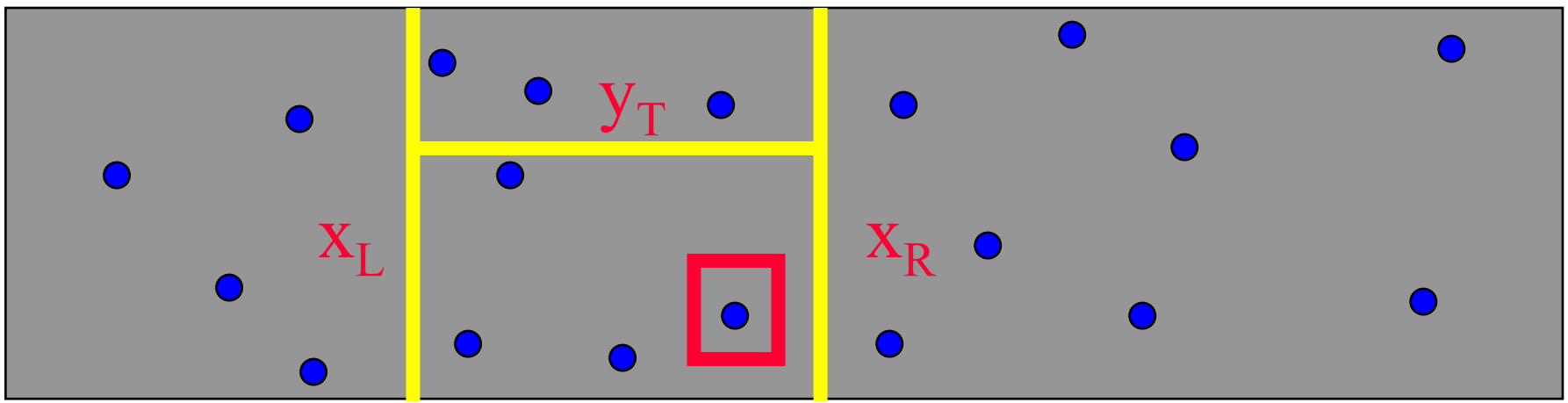


- Trả về phần tử có tọa độ x nhỏ nhất thỏa mãn:

$x_L <= x <= x_R$ and $0 <= y <= y_T$.

maxXinRectangle(x_L,x_R,y_T)

- Trả về phần tử có tọa độ x lớn nhất trong HCN được định nghĩa bằng 4 đường thẳng:
- $x = x_L$, $x = x_R$, $y = 0$, $y = y_T$, $x_L <= x_R$, $0 <= y_T$.

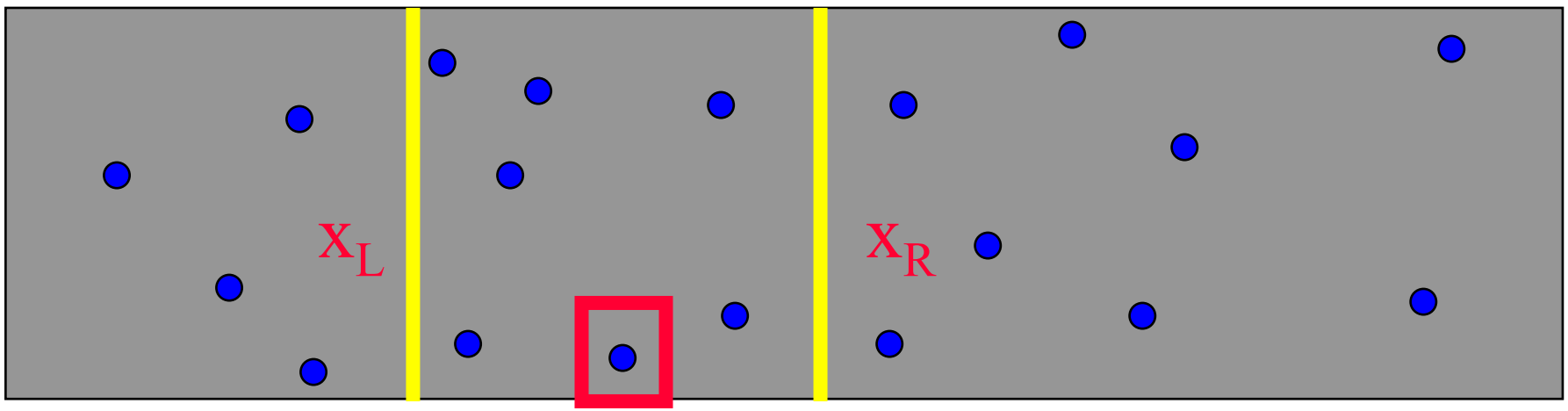


- Trả về phần tử có tọa độ x lớn nhất thỏa mãn:

  $x_L <= x <= x_R$ and $0 <= y <= y_T$.

minYinXrange(x_L,x_R)

- Trả về phần tử có tọa độ y nhỏ nhất trong HCN:
- $x = x_L$, $x = x_R$, $y = 0$, $y =$ infinity (vô cùng), $x_L <= x_R$.

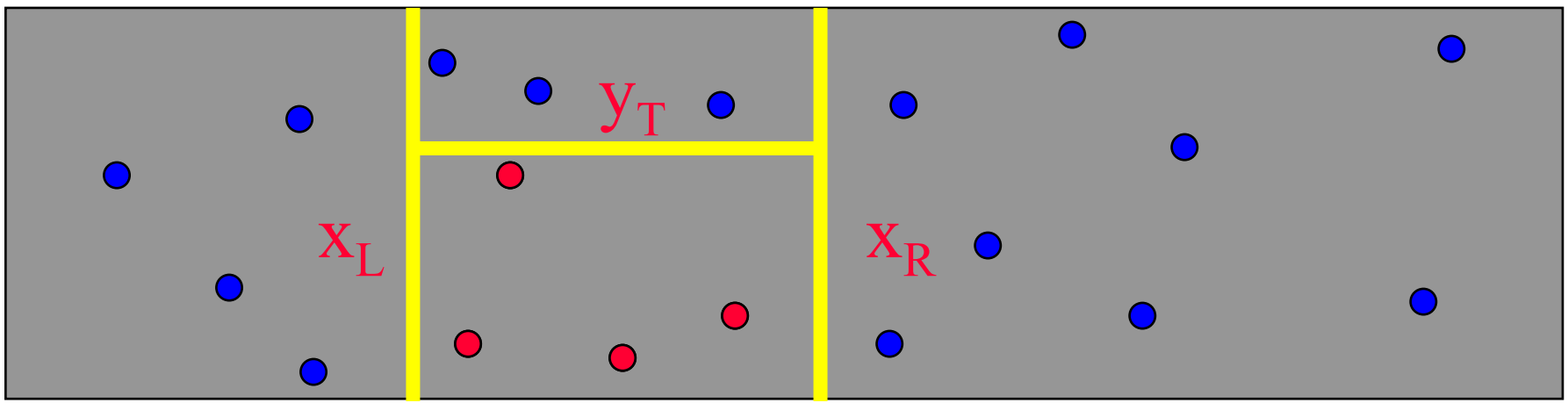


- Trả về phần tử có tọa độ y nhỏ nhất thỏa mãn:

$x_L <= x <= x_R$.

enumerateRectangle(x_L,x_R,y_T) –liệt kê các phần tử trong HCN

- Trả về tất cả các phần tử trong HCN
- , $x = x_L$, $x = x_R$, $y = 0$, $y = y_T$, $x_L <= x_R$, $0 <= y_T$.



- Trả về các phẩn tử có tọa độ thỏa mãn:

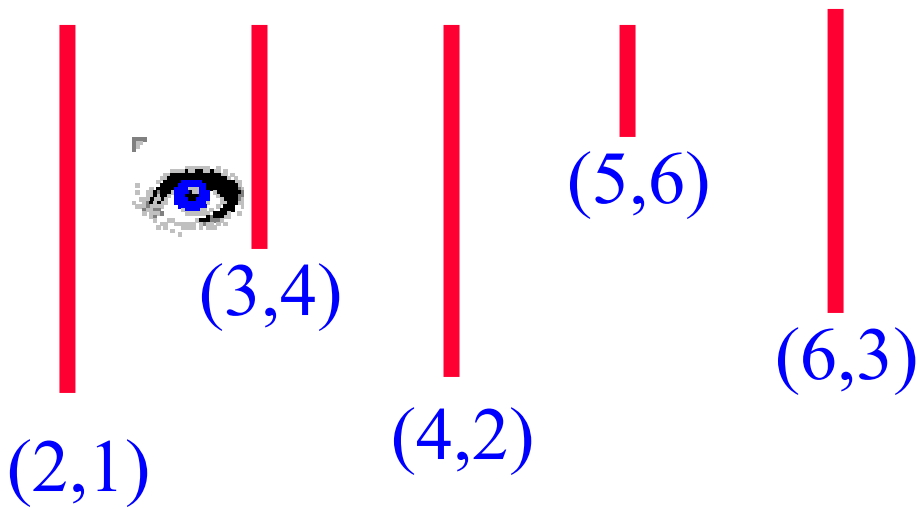
$x_L <= x <= x_R$ and $0 <= y <= y_T$.

Complexity – Độ phức tạp của các phép toán

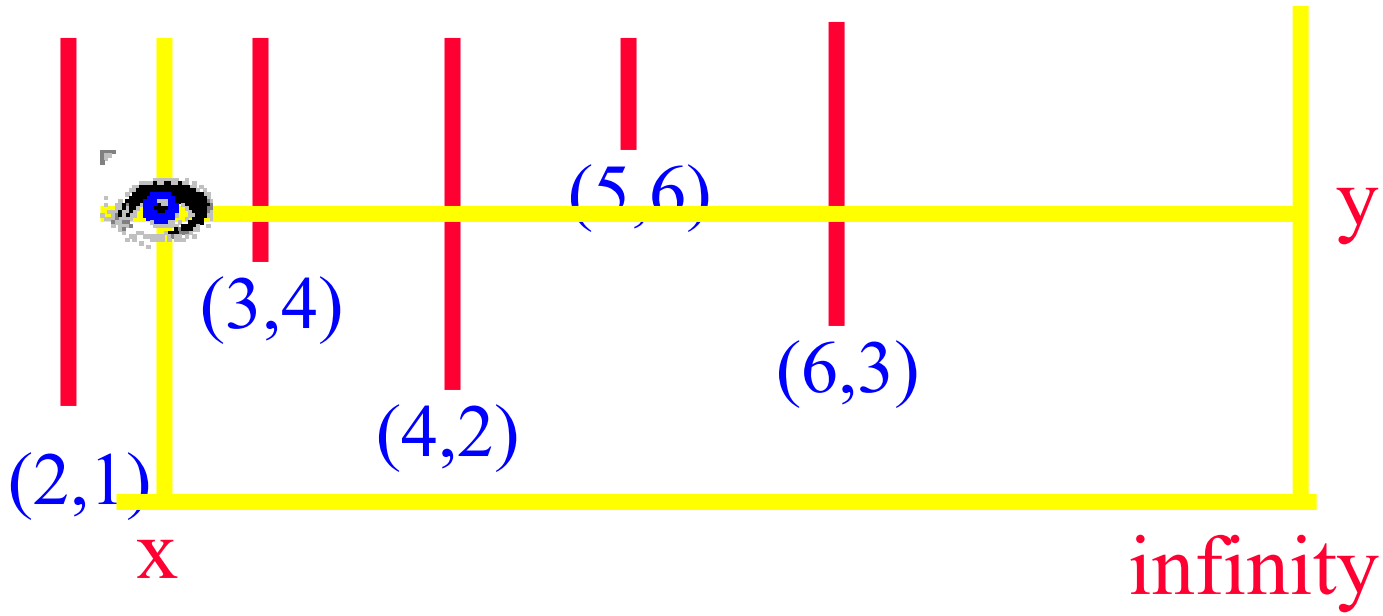
- Độ phức là O(log n) với tất cả các phép toán trừ phép toán enumerateRectangle. Với n số nút trên cây.
- Độ phức tạp của phép toán enumerateRectangle là O(log n + s), với s là số phần tử trong HCN

Applications – Visibility Các ứng dụng về tầm nhìn

- Tập của các đoạn thẳng đứng nửa vô hạn:
  - Các đường thẳng đứng có điểm đầu (x_i,y_i)và điểm cuối $(x_i,\text{infinity})$.

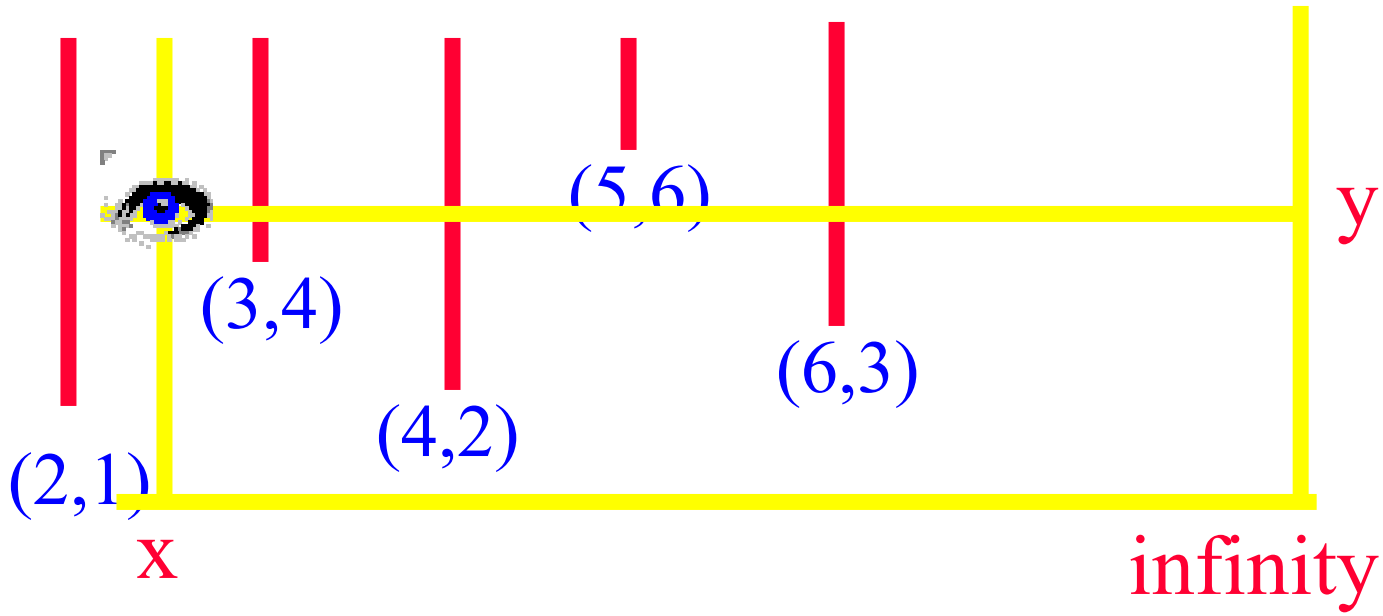


Translucent Lines – Các đường trong suốt



- Mắt Eye đặt tại (x,y) => xác định các các đường sẽ nhìn thấy.
  - Xây dựng cây: Priority search tree of line end points.
  - Thực hiện phép toán enumerateRectangle(x, infinity, y).

Opaque Lines (Các đường mờ)



- Mắt Eye đặt tại (x,y), xác định đường thẳng sẽ nhìn thấy
  - Xây dựng cây: Priority search tree of line end points.
  - Thực hiện phép toán: minXinRectangle(x, infinity, y).

Bài toán 1: Bin Packing – Bài toán cái túi

Định nghĩa: Cần xếp m phần tử vào n túi.

Một số cách xếp:

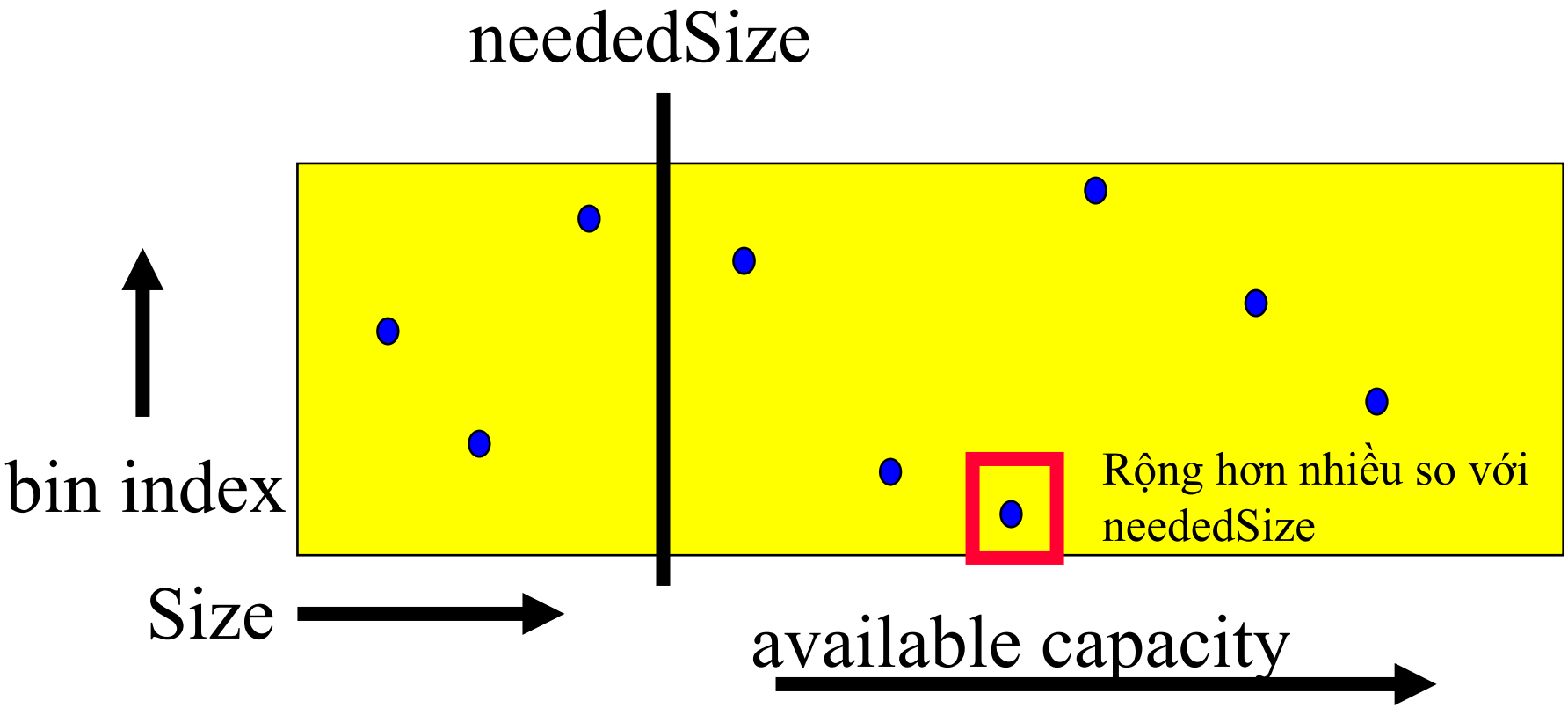
- First fit. Sử dụng cách đóng phù hợp đầu tiên
- Best fit. Sử dụng cách đóng phù hợp nhất
- Combination- một vài phần tử được đóng thùng theo First fit, số khác theo Best fit.

Combined First And Best Fit

- Các túi được đánh số 0, 1, …, n – 1.
- Kích thước mỗi túi là c.
- Khởi tạo cây với các cặp (c, j), 0 <= j < n.

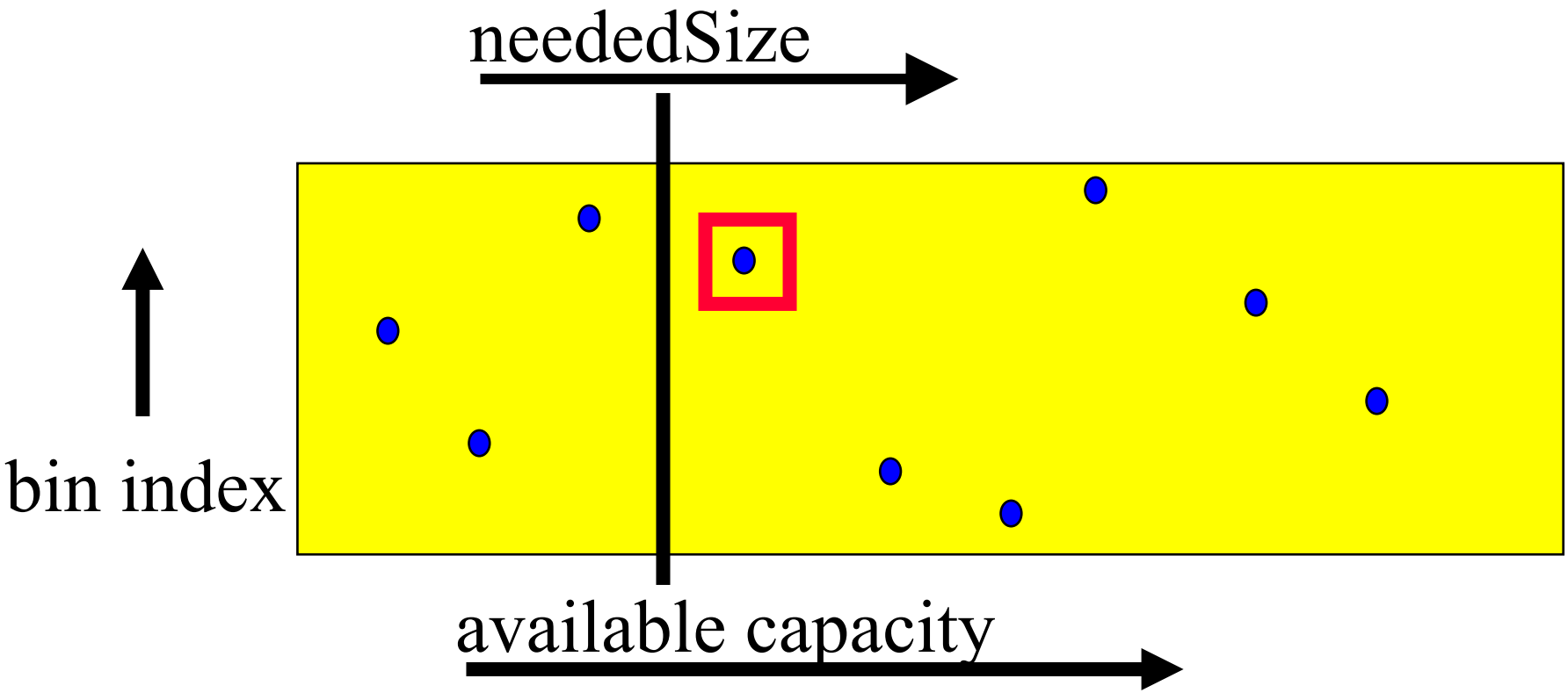
First Fit – Tìm túi có chỉ số nhỏ nhất có thể chứa được kích thước yêu cầu

- minYinXrange(neededSize, infinity)



Best Fit – Tìm túi có kích thời phù hợp nhất

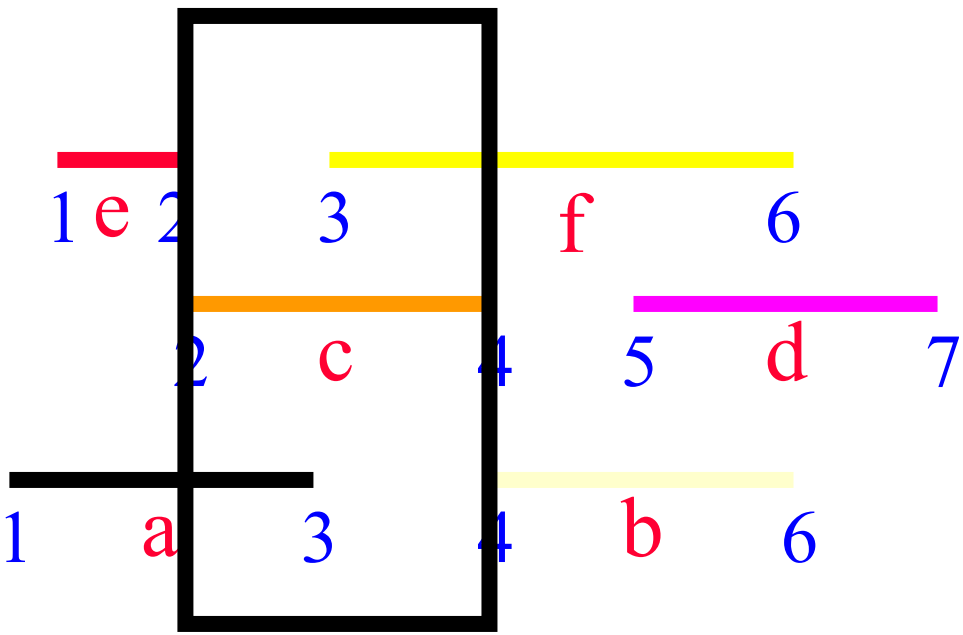
- minXinRectangle(neededSize, infinity, infinity)



Bài toán 2: Giao giữa các đoạn -

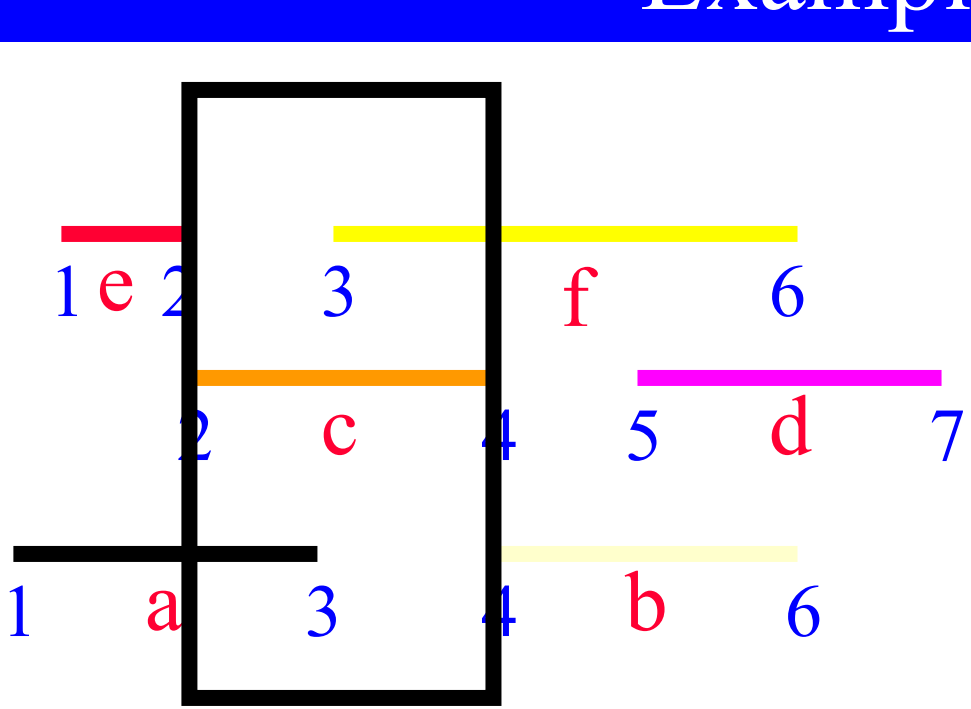
- Các đoạn chính là cặp số: [i,j], i < j.
- Ý nghĩa của đoạn[i,j] có thể là máy bận trong khoảng thời gian i đến j.
- Cần trả lời các câu hỏi dạng: các khoảng nào giao/gối lên với khoảng [u,v], u < v.
  - Danh sách các máy bận trong khoảng thời gian từ u đếm v.

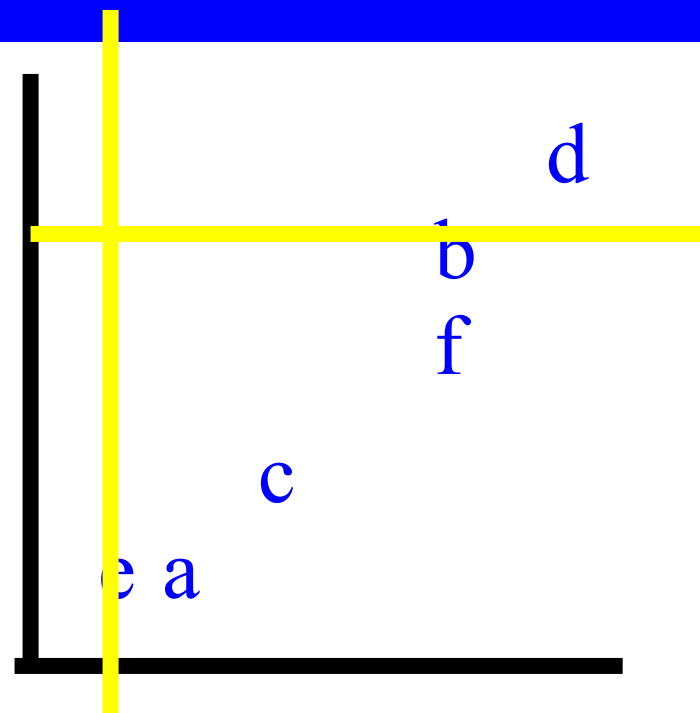
Example



- Machine a is busy from time 1 to time 3.
- Machines a, b, c, e, and f are busy at some time in the interval [2,4].

Example

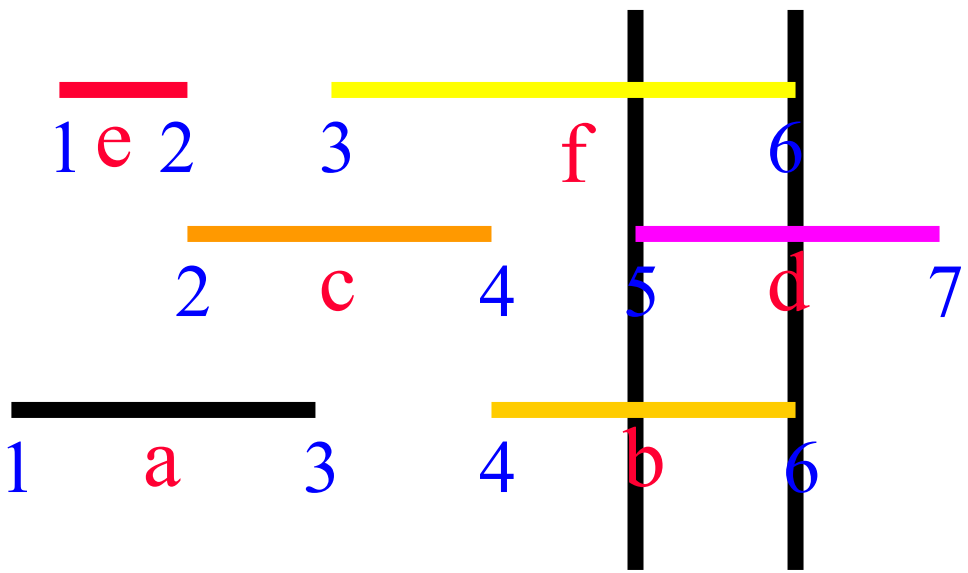




- Đoạn [i,j] tương ứng với điểm (x,y), với x = j và y = i. Trên trục tọa độ: điểm e(2,1); a(3,1); c(4,2);f(6,3); b(6,4);d(7,5)
- enumerateRectangle(u, infinity, v).
- enumerateRectangle(2, infinity, 4).- các máy bạn trong khoảng thời gian [2,4]

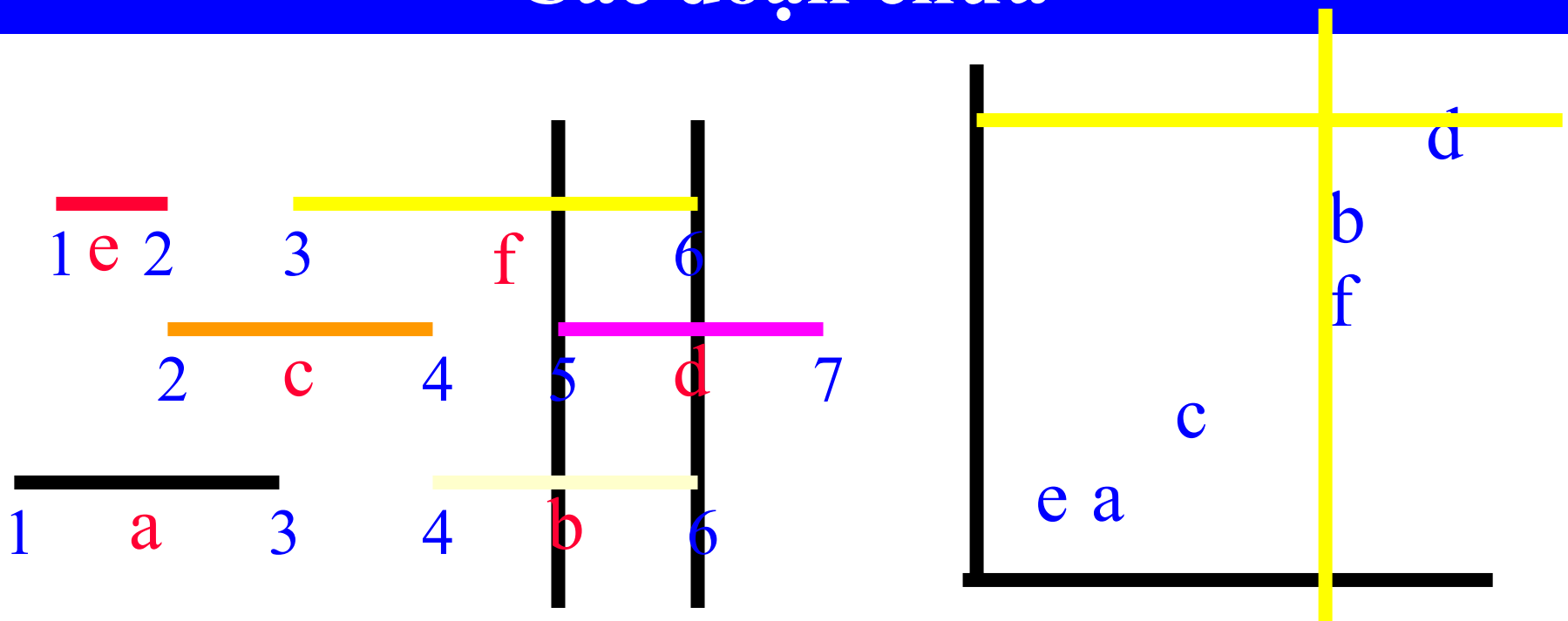
Bài toán 3: Các đoạn chứa

- Danh sách các đoạn [i,j] chứa đoạn [u,v].
- [i,j] chứa [u,v] iff i <= u <= v <= j.



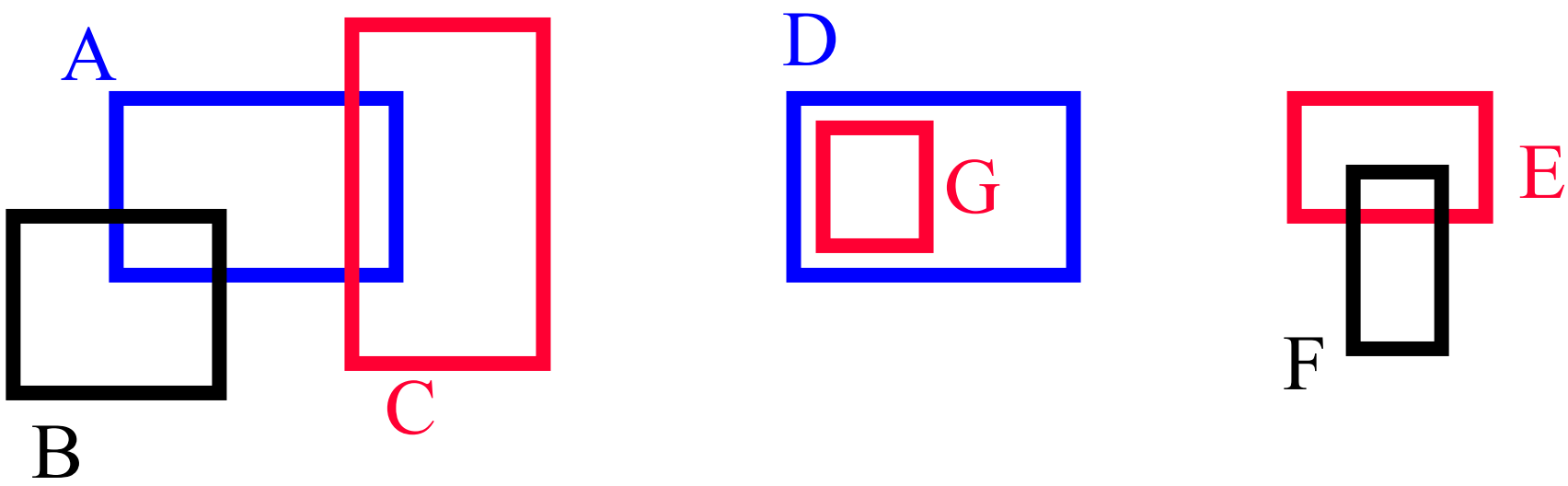
[u,v] = [5,6]

Các đoạn chứa



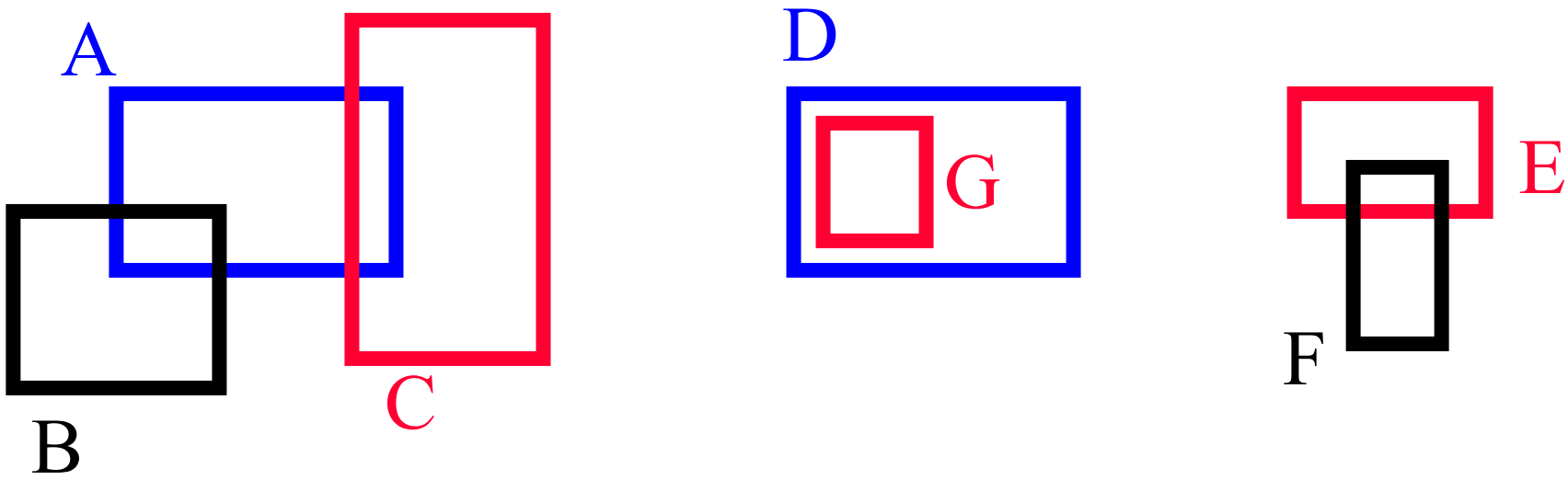
- Đoạn [i,j] tương ứng với điểm (x,y), với x = j và y = i.
- enumerateRectangle(v, infinity, u).
- enumerateRectangle(6, infinity, 5).- các đoạn chứa đoạn [5,6]

Bài toán 4:Intersecting Rectangle Pairs – Giao của các cặp HCN



- (A,B), (A,C), (D, G), (E,F)

Algorithm



- Examine horizontal edges in sorted y order –Kiểm tra các cạnh nằm ngang được xếp theo y.
  - Bottom edge => insert interval into a priority search tree.
  - Top edge => report intersecting segments and delete the top edge's corresponding bottom edge.

Complexity

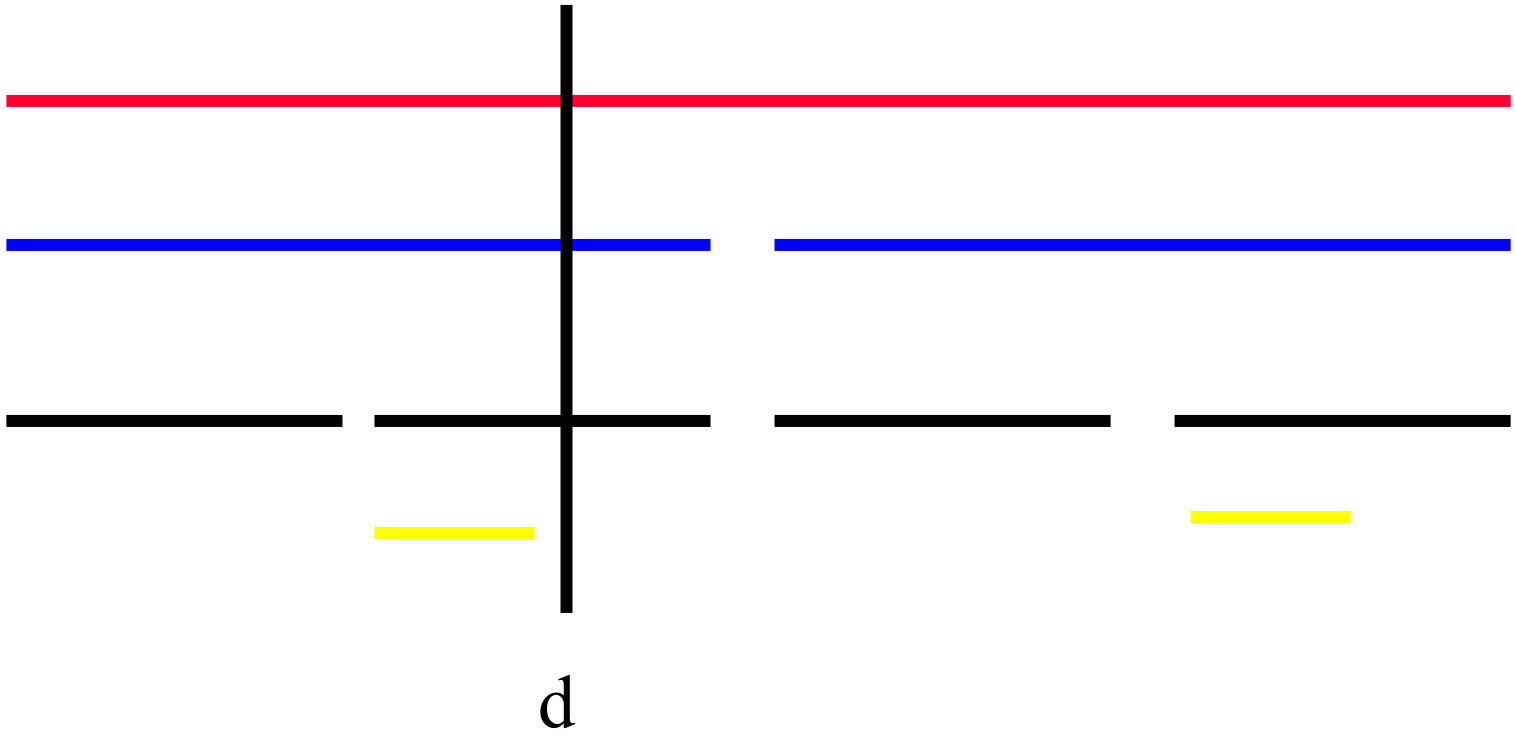
- Examine edges in sorted order.
  - Bottom edge => insert interval into a priority search tree.
  - Top edge => report intersecting segments and delete the top edge's corresponding bottom edge.

- O(n log n) to sort edges by y, where n is # of rectangles.
  - Insert n intervals … O(n log n).
  - Report intersecting segments … O(n log n + s).
  - Delete n intervals … O(n log n).

Bài toán 5: IP Router Table

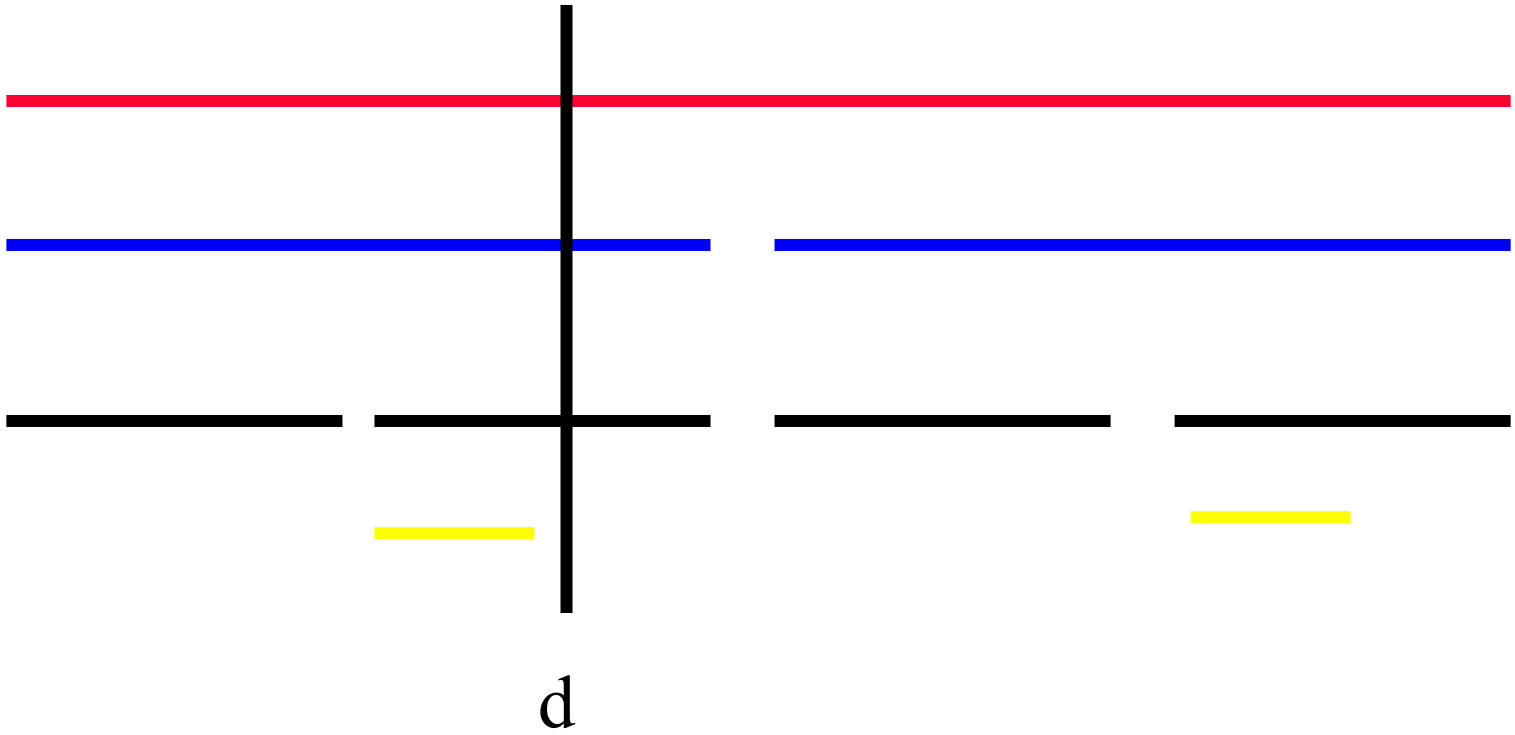
- Longest-prefix matching
  - 10*, 101*, 1001*
  - Destination address d = 10100
  - Longest matching-prefix is 101*
- Prefix is an interval
  - d is 5 bits => 101* = [10100, 10111] = [20,23]
- 2 prefixes may nest but may not have a proper intersection (Proper intersection means one is not contained in the other and there is an overlap)

IP Router Table



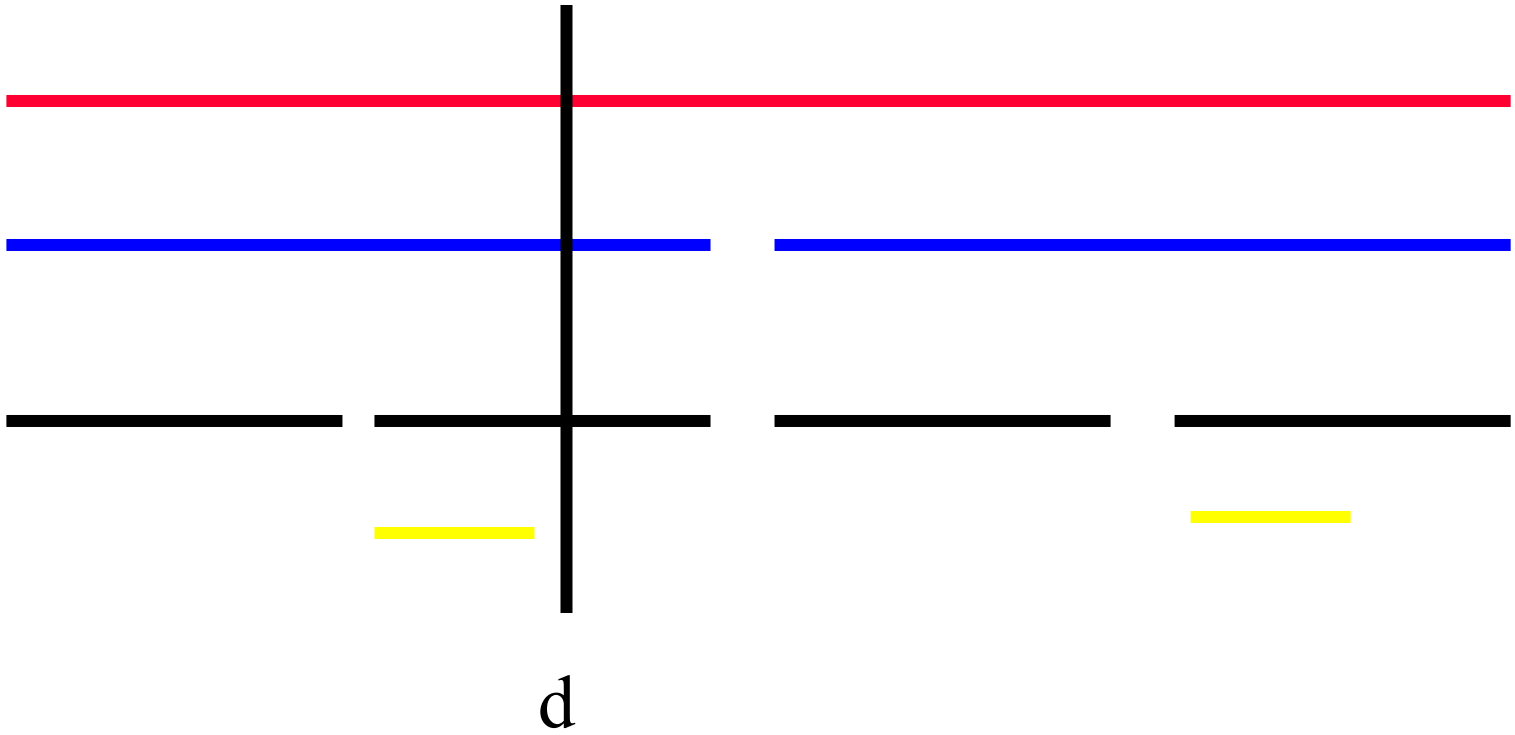
- p(d) = Các tiền tố khớp với địa chỉ d. Khi đó:
  - p(d) = enumerateRectangle(d,infinity,d)

IP Router Table



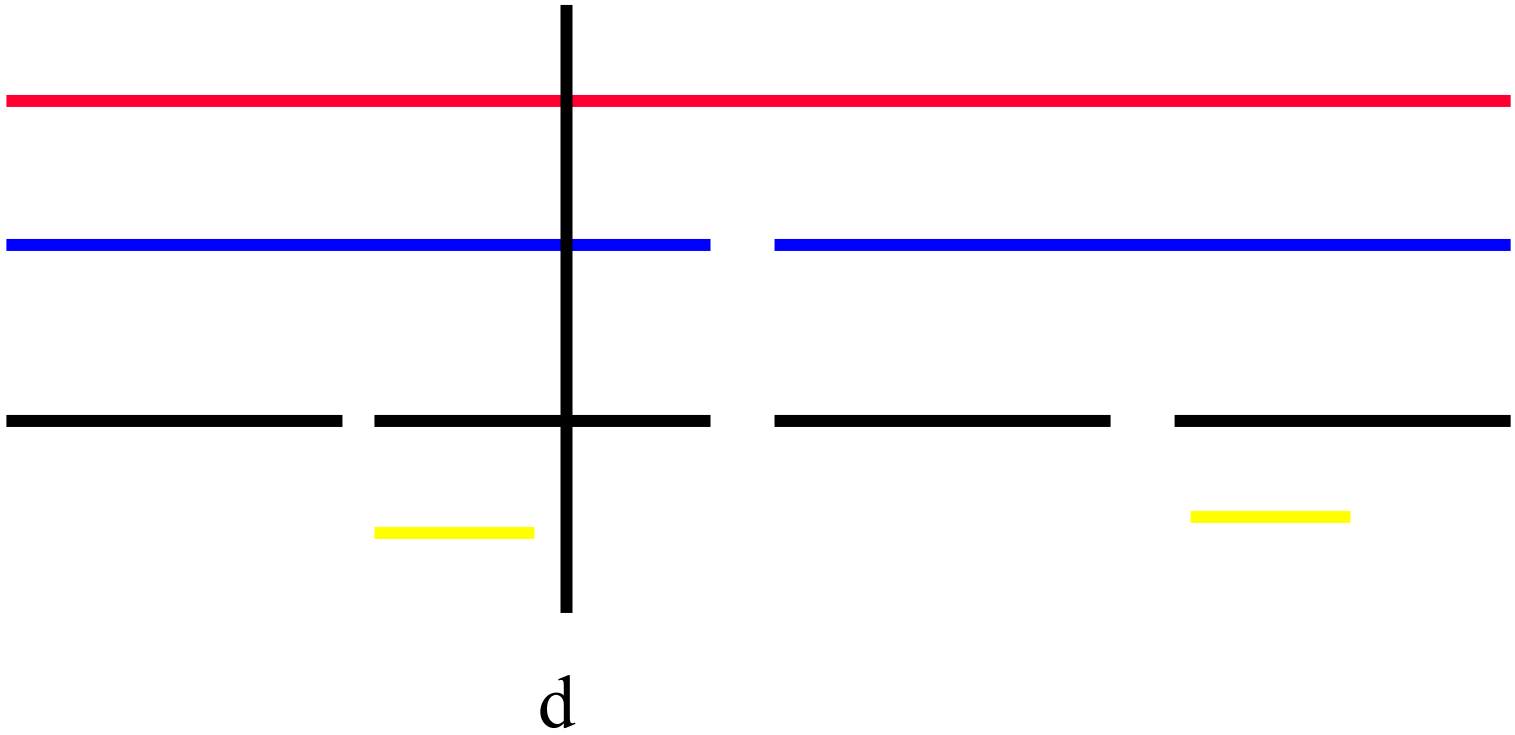
- lpm(d) = [maxStart(p(d)), minFinish(p(d))]
- minXinRectangle(d,infinity,d) finds lpm(d) except when >1 prefixes have same finish point.

IP Router Table



- Remap finish points so that all prefixes have different finish point.
- f' = $2^w f - s + 2^w - 1$, w = length of d
- f' is smaller when s is bigger

IP Router Table



- Complexity is O(log n) for insert, delete, and find longest matching-prefix.