6/1) Worst case is O(n2). It happens when the range search is such that all the elements are selected in both the x and y search. And hence to match one element from x search, we have to go through n elements in worst case.

6/2)

Sample Input:

Rectangle 1 coordinates:- (0,0),(2,0),(2,2), (0,2)

Rectangle 2 coordinates:- (1,1), (1,3), (3,3), (3,1)

Sample output: [[1,2],[2,1]]

6/3)

n/16

n/16

n/16

n/16

n/4

n/4

n

=( log4n ) times

T(n)=1+2+4+8+……………2log4n

T(n)= 2log4n-1

T(n)=

Hence,Proved.

6/4)

a) def rangeQuery(lower,upper,root):

if root.axis=='x':

if root.x<lower[0]:

rangeQuery(lower,upper,root.right)

elif root.x>upper[0]:

rangeQuery(lower,upper,root.left)

else:

print (root.x,root.y)

rangeQuery(lower,upper,root.left)

rangeQuery(lower,upper,root.right)

else:

if root.y<lower[1]:

rangeQuery(lower,upper,root.right)

elif root.y>upper[1]:

rangeQuery(lower,upper,root.left)

else:

print (root.x,root.y)

rangeQuery(lower,upper,root.right)

rangeQuery(lower,upper,root.left)

b)

def nNeighbour(root,p,minPt,outPt):

nod=root

while nod!=None:

if nod.axis=='y':

if nod.keyy<minPt[1]:

nod=nod.right

else:

nod=nod.left

else:

if nod.key<minPt[0]:

nod=nod.right

else:

nod=nod.left

nod=nod.parent

if p>(nod.key-minPt[0])\*\*2+(nod.key-minPt[1])\*\*2:

p=(nod.key)\*\*2+(nod.key)\*\*2

outPt[0]=nod.key

outPt[1]=nod.key

if root.parent!=None:

if root.parent.axis=='x':

if p>minPt[0]-root.parent.key:

if root==root.parent.left:

start(root.parent.right,p,minPt,outPt)

else:

start(root.parent.left,p,minPt,outPt)

else:

if p>minPt[1]-root.parent.key:

if root==root.parent.left:

start(root.parent.right,p,minPt,outPt)

else:

start(root.parent.left,p,minPt,outPt)

def start(root,p,minPt,outPt):

x=root

while x!=None:

if x.axis=='y':

if x.key<minPt[1]:

x=x.right

else:

x=x.left

else:

if x.key<minPt[0]:

x=x.right

else:

x=x.left

nNeighbour(x,p,minPt,outPt)

6/5)

Sample Input:-

Enter number of points whose coordinates you want to enter3

Enter x coordinate of 0 point1

Enter y coordinate of 0 point3

Enter x coordinate of 1 point2

Enter y coordinate of 1 point2

Enter x coordinate of 2 point3

Enter y coordinate of 2 point1

Sample Output:-

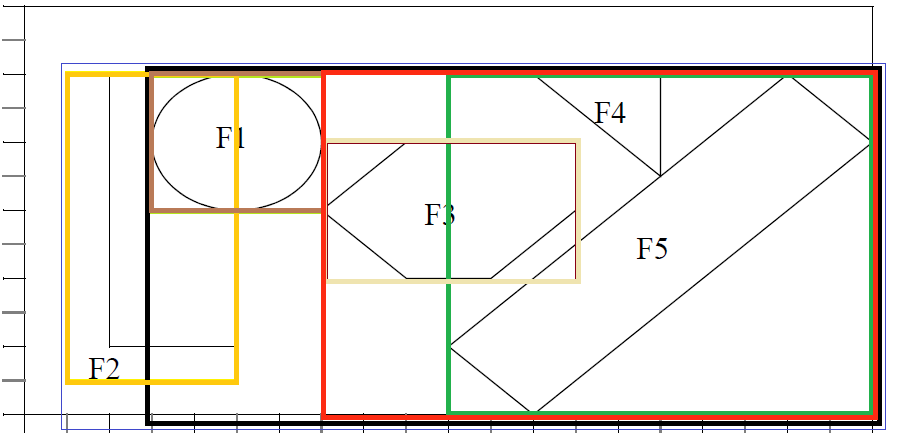
P1= [[1, 3], [2, 2]]

P2= [[3, 1]]

P1'= [[2, 2], [3, 1]]

P2'= [[1, 3]]

6/6) Sample input will be files containing input of arbitrary length.

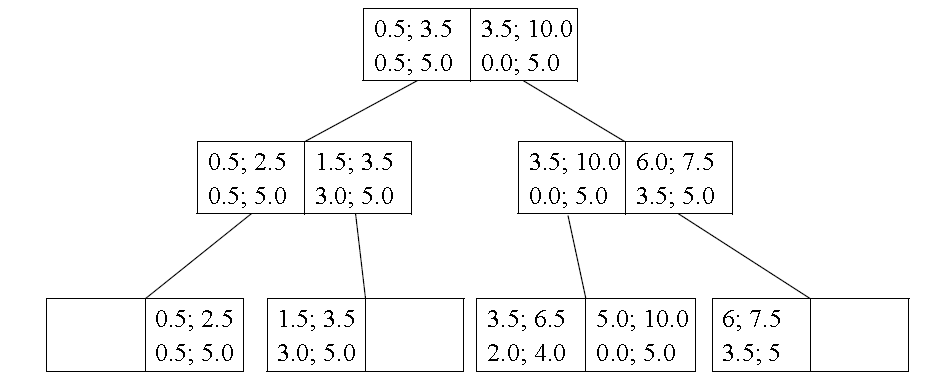
6/7) 

Structure of entries: x-coor. bottom, left x-voor. top, right

y-coor. bottom, left y-coor. top, right

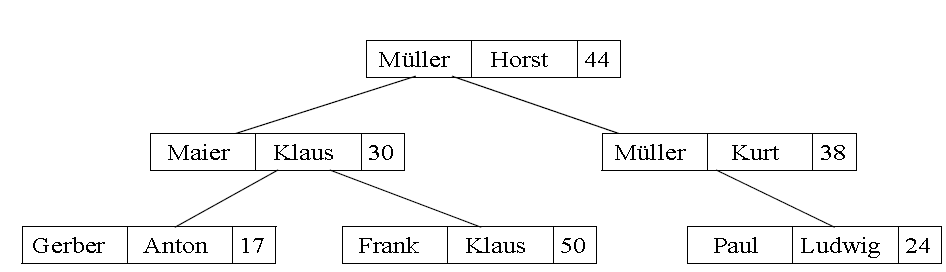
1. discriminator : x-coor bottom, left,

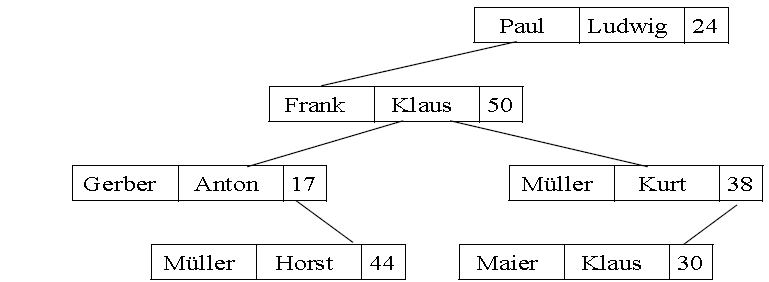
2. discriminator : x-coor. top, right

After inserting all object areas.

1. The search returns all possible candidate as well as objects F3 and F5. Because the surrounding rectangle must not be completely equal to the requested object, for all objects, the containment criteria must be checked. By doing so, only F3 remains as solution.
2. F3 and F4.
3. F1 and F3.

6/8)

(a)

(b)

(d) Because we cannot quantify the alphabetical input in space.