PROJECT Sami Ahmad Khan 1

A20352677

PROBLEM: SEPARATING POINTS BY AXIS PARALLEL LINES

GIVEN:

Input -> set of n points in 2D plane

Output -> Set S of combinations of vertical and Horizontal lines seperating all points.

Assumptions: No two points have some X or Y coordinates.

MEASURE: minimise S.

GREEDY HEURISTIC

1 PSEUDOCODE:

READ_FILES ():

Read all files from "Input" folder.

Read all points.

return points

CREATE_PAIRS (points):

U=[] Il union of all possible pairs of all points.

for i in range (0, length (points)):

for j in (it, length (points)):

U. append (points [i], points [j])

return U // returns Union of all pairs

SEPARATE_POINTS (U,n):

Here n is the total no. of points.

Let vaxis
set of all vertical axises.

and haxis
set of all torizontal axises.

S=[]
solution SET

while U b not Empty:

let count V

no. of axis cut by an vertical axis. (declared as dict) and count H

no. of axis cut by a horizontal axis. (declared as dict).

count the total Lease point from U
separated by Vertical Axis es and store
in count V as REY-Value pair

Do the same for and horizontal Axises and store in count H as KEY-Value pair

Select on an axis that cuts the maximum pair out of both vertical and horizontal axis.

S
Store that axis in solution set.

U
U - S // Remove those pairs from

Union set that are cut by S.

// S(pairs)
pairs aut by S.

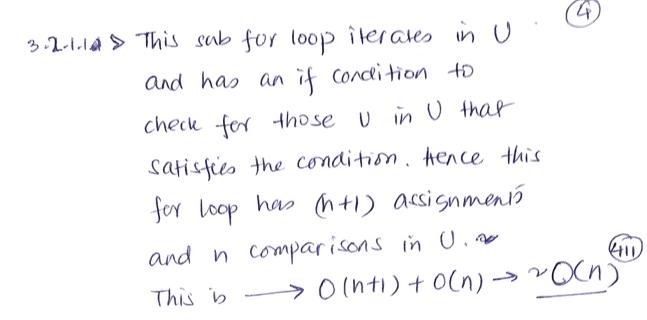
Heturn S // s is our solution set

2. ANALYSIS OF RUNTIME:

- Reading Files takes $\sim O(n)$. (depending on how large n is.). Here n can be no of lines in each all the files.
- 2. CREATE_PAIRS() function creates pair of a point with all other possible points. This runs in T(n) = n * (n-1). Hence runtime complexity of creating pairs is $\sim 0 (n^2)^{\frac{3}{2}}$.
- 3 · SEPARATE_POINTS () function has following assignments and comparisons:
- # P.S -> Here 3-1 Creating all vertical and horizontal the small nos. axises takes -> ~ O(n). Vertical in civiles above and Horizontal lenes are stored in a the complexity tist. Adaition to list to mo(n) bethe ordering of the sequence 3.2 in which code is 300 Thereo a first while loop. The while running and not loop has following assignments and powers \(\circ \comparison: \rightarrow \circ (U-scpairs)) \rightarrow \circ (n/2) \)
 - 3.2.10 o First for wop for iterating in vertical or Horizontal axises List.

 This is -> O (size of each List) > 20(n)

 This for wop has subloop to iterate in U (Union set)



3.2.2 De Here comparing for maximum (42) between two axis takes 2 * O(1), ~ O(1).

3.23 % Here addition of axises to solution set (List in python) takes

a runtime of O (min. aio. of axises)

-> ~ o(n).

3.2.2 o There's another for loop at the last to remove pains for from U, which runs in O(U)-size) -> ~ O(n)

So our runtime is something as follows (in hierarchy): $0(n)^{0} \quad \textcircled{*} \quad 0(n/2)^{0}$ $0(n^{2})^{0} \quad \textcircled{*} \quad 0(n)^{0}$ $0(n)^{0} \quad \textcircled{*} \quad 0(n)^{0}$

We can see from the diagram, that 3Adding all the runtime we get:
Total Runtime = $O(n)^0 + O(n^2)^0 + O(n)^3 + (O(n/2)^4 \times O(n)^4 \times O(n)^4$

Total Runtime
$$(T(n)) \cong O(n^3)$$

3. One instance where my greedy solution fail to return optim um solution is when all the point are in linearly increasing order or decreasing order.

For example, instance = [(1,1),(2,2),(3,3),(4,4),(5,5)]

The output was = 4

h 3.5

h 2.5

h 9.5

Hence all the points were seperated using maximum a parallel axise. Hence this type of instance will not be optimal.

h 1.5

And there is no better solution for this kind of 6) problem instance.

Another instance

Another instance where we I did not get optimal solution was:

instance 06. txt from website. my output	(Better Solution) orisinal output
output: 15 axises W 15.5 V 15.5	14 axiss.
h 23.5 h 9.5	h 14-5 h 22-5 h 5-5
v 8.5 v 21.5 h 18.5	V 22.5 V7.5
V 4.5 V 26.5	V26.5 V11.5 h10.5
h 26.5 h 11.5	V4.5 V13.5
h 5.5 h 22:5 h 17.5	V16.5 V20.5 V27.5
h 1.5	