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 same 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 = [ ]    // union of all possible pairs of all
                  points.
    for i in range (0, length (points)):
        for j in (i+1, length (points)):
            U. append (points[i], points[j])

    return U    // returns Union set 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 horizontal axises.
S = [ ] → solution SET
while U is not Empty :

Let countV ← no. of axis cut by a vertical axis. (declared as dict)
and countH ← no. of axis cut by a horizontal axis. (declared as dict).

Count the total pairs from U separated by a vertical Axises and store in countV as KEY-value pair.

Do the same for horizontal Axises and store in countH as KEY-value pair

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

S ← store that axis in solution set.
U ← U − S(pairs) // Remove those pairs from Union set that are cut by S.
// S(pairs) → pairs cut by S.

return S // S is our solution set

2. ANALYSIS OF RUNTIME:

1 • Reading Files takes ~O(n) (1) (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 ~O(n²) (2).

3 • SEPARATE_POINTS() function has following assignments and comparisons :—

3.1 – Creating all vertical and horizontal axises takes → ~O(n). Vertical and Horizontal lines are stored in a List. Addition to List is ~O(n) (3)

* P.S → Here the small nos. in circles above the complexity is the ordering of the sequence in which code is running and not powers.

3.2 – Theres a first while loop. The while loop has following assignments and comparison :— → ~O(U - sc pairs)) → ~O(n/2) (4)

3.2.1 ~~3.1.1~~ o First for loop for iterating in vertical or Horizontal axises List. This is → O(size of each List) → ~O(n) (4)

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

(4)

3.2.1.1 ⊳ This sub for loop iterates in U and has an if condition to check for those u in U that satisfies the condition. Hence this for loop has (n+1) assignments and n comparisons in U. (411)

This is ⟶ $O(n+1) + O(n) \rightarrow \sim O(n)$

3.2.2 ⊳ Here comparing for maximum between two axis takes $2 * O(1) \sim O(1)$. (412)

3.2.3 ⊳ Here addition of axises to solution set (List in Python) takes a runtime of O(min. no. of axises) ⟶ $\sim O(n)$. (43)

3.2.2 ∘ There's another for loop at the last to remove pairs from U, which runs in O(U-size) ⟶ $\sim O(n)$ (44)

So our runtime is something as follows (in hierarchy):

$O(n)$ (1)
↓
$O(n^2)$ (2)
↓
$O(n)$ (3)
↓
$O(n/2)$ (4) (*)

(*) $O(n/2)$ (4)
- → $O(n)$ (41)
  - → $O(n)$ (411)
- → $O(1)$ (42)
- → $O(n)$ (43)
- → $O(n)$ (44)

~~We can see from the diagram, that~~

Adding all the runtime we get:-

$$\text{Total Runtime} = O(n)^{(1)} + O(n^2)^{(2)} + O(n)^{(3)} + \left(O(n/2)^{4} \times O(n)^{(41)} \times O(n)^{(411)}\right) + \left(O(n/2) \times O(1)\right) + \left(O(n/2) \times O(n)\right) + \left(O(n/2) \times O(n)\right)$$

$$= 2O(n) + O(n^2) + O(n^3/2) + O(n/2) + O(n^2/2)$$

Here $O(n^3/2)$ has the highest degree.

So we can avoid all lower ordered terms.

$$\therefore \boxed{\text{Total Runtime } (T(n)) \cong \underline{\underline{O(n^3)}}}.$$

3. One instance where my greedy solution fail to return optimum solution is when all the points 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 4.5
h 1.5

Hence all the points were seperated using maximum 4 parallel axises. Hence this type of instance will not be optimal.

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

~~Another instance~~

Another instance where I did not get optimal solution was:

instance06.txt from website.

output: my output 15 axises

15.5
v 15.5
h 23.5
h 9.5
v 8.5
v 21.5
h 18.5
v 4.5
v 28.5
h 26.5
h 11.5
h 5.5
h 22.5
h 17.5
h 1.5

(Better Solution)
orisinal output

14 axises.

v 15.5
h 14.5
h 22.5
h 5.5
v 22.5
v 7.5
v 26.5
v 11.5
h 10.5
v 4.5
v 13.5
v 16.5
v 20.5
v 27.5