CS 430 Introduction to Algorithm

Final Project

Li Xu (A20300818)

Introduction

In this project, I implemented a greedy algorithm that can separate n points in the two-dimensional plane, with minimal axis-parallel lines.

According to Section 35.3 in the CLRS, this is a minimal set cover problem. The minimal axis-parallel lines are required, so we break the largest subset of points until we have no connections.

To implement this algorithm, I consider every point in the plane is connected with each other, an axis-parallel line which separates points will break the connection between those points. The greedy algorithm choose the axis-parallel line that breaks the most connections in every round, until all the connections are broken.

However, this greedy algorithm sometimes doesn’t yield the best solution, which is a nature of the greedy algorithm

Pseudocode for the Algorithm

```
Initialize connection between points
    for i = 1 to n
        for j = 1 to n
            point[i] connects with point[j]

Initialize axis-parallel lines between each pair of points
    for i = 1 to n
        line = vertical.((point[i] + point[i + 1])/2)
    for j = 1 to n
        line = horizontal. ((point[i] + point[i + 1])/2)

Closest points to the left or bottom of the specific intersection
    for i = 1 to n
        if line is vertical
            coordinate = point[i].x
        if line is horizontal
            coordinate = point[i].y

Commit a line
    call closest point function to find closestpoint.id
    for i = 1 to closestpoint.id
        for j = 1 to closestpoint.id – 1
            break connection between point[i] and point[j]

Number of connections to break
    call closest point function to find closestpoint.id
    for i = 1 to closestpoint.id
        for j = closestpoint.id + 1 to n
            if there is a connection between point[i] and point[j]
                break the connection

Main function
    Initialize Connections
    Initialize Lines
    for i = 1 to n
        for j = 1 to n
            find Max[number of connections to break]
            commit that line
            stop when all connections are broken
```

Analysis of the Running Time

1. The initialize connection function between line 1 to line 4 is a $O(n^2)$ function. There are two nested for loops that connects every point with each other.
2. The initialize lines function between line 6 to 10 is a O(n) function . It draws lines between every other points, which generates the worst solution to this problem.
3. Closest point function is a O(n) algorithm that returns the id of the point that closest to the left or the bottom of the specific intersect.
4. Commit lines function ln line 19 to 23 is a $O(n^2)$ function that finalize a line and breaks the connections.
5. Number of connections to break function runs in $O(n^2)$, which returns the number of connections that a line breaks.
6. In the main function, there are two nested for loops and two functions with maximum $O(n^2)$ running time, inside the loop, the highest running time is $O(n^2)$, therefore, the whole algorithm doesn't exceed $O(n^4)$, which meets the requirement of the project.

A Better Solution

As we all known, the greedy algorithm doesn't guarantee to give a optimal solution. For our project, this algorithm sometimes fails to give to best solution, in other word, it doesn't yield the minimal axis-parallel lines.

For example, in the instance01 given in the description of the project:

```
10
1 10
2 6
3 8
4 1
5 3
6 7
7 2
8 9
9 5
10 4
```

There are 10 points in the two-dimensional plane and my algorithm gives the following solution:

```
6
v 5.5
h 4.5
h 6.5
v 7.5
v 1.5
v 4.5
```

However, here is a better solution with only 5 axis-parallel lines to separate all points:

```
5
v 5.5
h 2.5
```

h 4.5

h 6.5

h 8.5