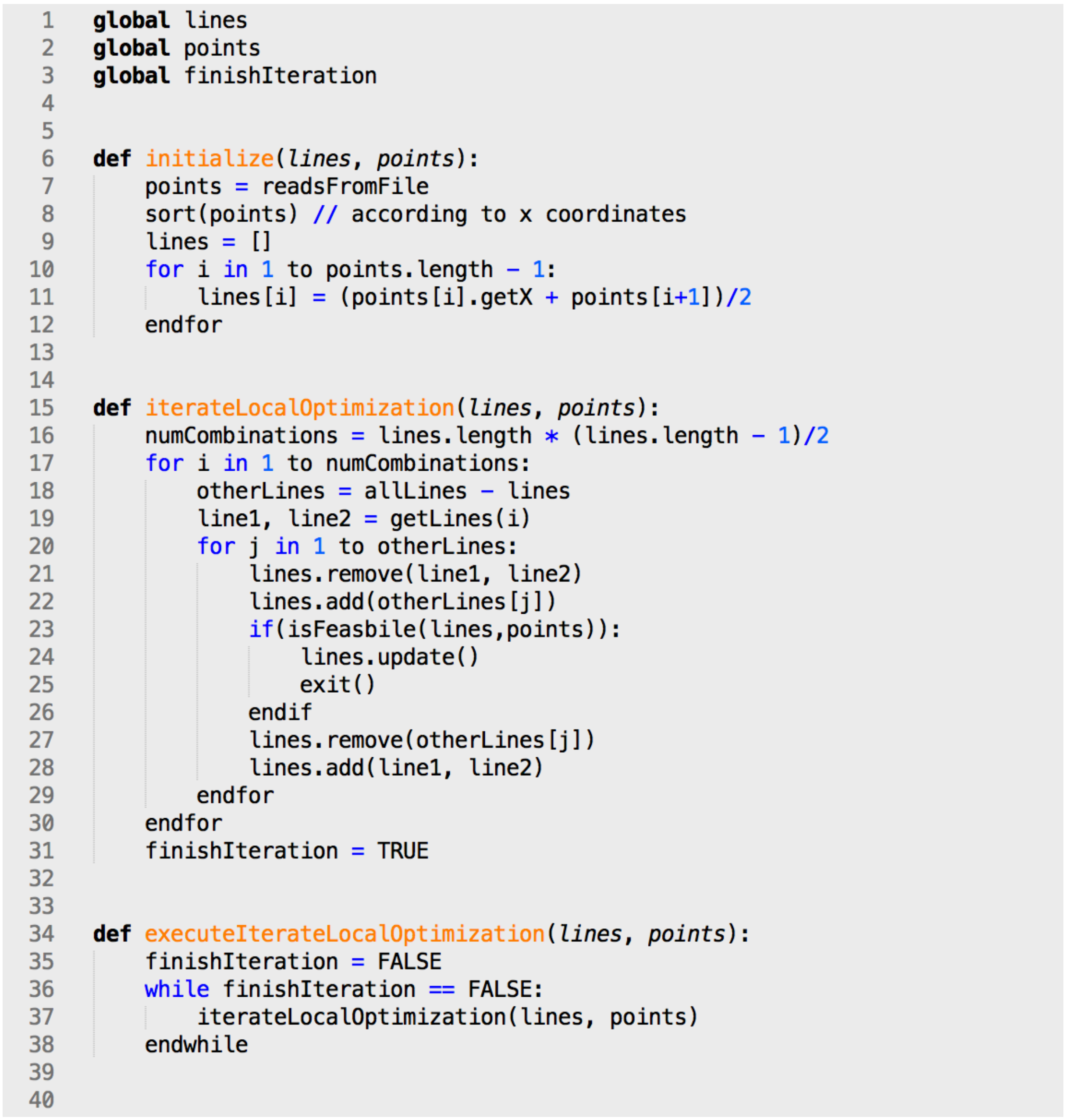
CS430 Introduction to Algorithm
Project Report

Lin Zhuo
20379609

1. Pseudo-code for the algorithm(s)



```
global lines
global points
global finishIteration


def initialize(lines, points):
    points = readsFromFile
    sort(points) // according to x coordinates
    lines = []
    for i in 1 to points.length - 1:
        lines[i] = (points[i].getX + points[i+1])/2
    endfor


def iterateLocalOptimization(lines, points):
    numCombinations = lines.length * (lines.length - 1)/2
    for i in 1 to numCombinations:
        otherLines = allLines - lines
        line1, line2 = getLines(i)
        for j in 1 to otherLines:
            lines.remove(line1, line2)
            lines.add(otherLines[j])
            if(isFeasbile(lines,points)):
                lines.update()
                exit()
            endif
            lines.remove(otherLines[j])
            lines.add(line1, line2)
        endfor
    endfor
    finishIteration = TRUE


def executeIterateLocalOptimization(lines, points):
    finishIteration = FALSE
    while finishIteration == FALSE:
        iterateLocalOptimization(lines, points)
    endwhile
```

```
def isFeasible(lines, points):
    vLines = lines.getVertical()
    hLines = lines.getHorizontal()
    unionNum = 0
    iLine = 0
    iPts = 0
    while iLine < vLines.length && iPts < points.length:
        while iPts < points.length && points[iPts].getX < vLines[iLine]:
            setUnion(points[iPts], unionNum)
            iPts = iPts + 1
        endwhhile
        unioNum = unionNum + 1
        iLine = iLine + 1
    endwhile
    // set the rest points to the last union
    while(iPts < points.length):
        setUnion(points[iPts], unionNum)
        iPt = iPt + 1
    endwhile
    mergesort(points)   // sort according to y coordinates
    iLine = 0
    iPts = 0
    unions = new HashSet()
    while iLine < hLines.length && iPts < points.length:
        while iPts < points.length && points[iPts].getY < hLines[iLine]:
            if unions.contains(getUnion(points[iPts]))
                return FALSE
            endif
            unions.add(getUnion(points[iPts]))
            iPts = iPts + 1
        endwhile
        unions.clear()
        unionNum = unionNum + 1
        iLine = iLine + 1
    endwhile
    // treat the rest points to the upmost horizontal line
    while iPts < points.length:
        if unions.contains(getUnion(points[iPts])):
            return FALSE
        endif
        unions.add(getUnion(points[iPts]))
    endwhile
    return TRUE
```

```
def main:
    initialize(lines, points)
    executeLocalOptimization(lines, points)
    output(lines)
```

2. Analysis of running time

This piece of code contains 4 functions: Initialize, executeLocalOptimization, IterateLocalOptimization and isFeasbile. At last is the main function.

Function initialization (Line6-12):
Reading points from file to memory takes O(n) time, sorting them takes O(nlogn) time. Line 10-11 initialize line according to the read points, which also takes O(n).
Therefore this function is O(nlogn).

Function isFeasible(Line 41-82):

- Lines 42-43, getting the vertical and horizontal lines, both O(n)
- Lines 47-54 are the loops for traversing the points and vertical lines in order. The points are separated by all the vertical lines and the groups are assigned with different union numbers. Line 49-50 and Line 52-53 both take O(1) time. This codes loop at most 2N times, therefore, Lines 47-54 have O(n) time complexity.
- Line 56-59 is O(n).
- Line 60 sort the points, which is O(nlogn)
- Lines 64-75 judge whether each group of points separated by vertical lines is split by the horizontal lines. Line 66-74 is O(1) because Line 66, which the Hash operation takes time O(1) and Line 69 also take O(1) time. Because after each loop iPts and iLine are both increased by one, The while loop execute at most 2n times, thus it is O(n).

As a summary, this function is O(nlogn) time complexity.

Function iterateLocalOptimization(Line 15-31):

- numCombinations is O(n^2)
- Loop 17 is executed O(n^2) times and Loop 20 O(n) time. According to the analysis above, line 23 the function isFeasbile is O(nlogn). Line 21, 22, 27, 28 are all O(n).

As a summary, this function is O(n^2)*O(n)*O(nlogn) = O(n^4logn)

Function executeIterateLocalOptimization(Line 34-38):

- By the definition of local optimization, in each iteration 2 lines are subtracted from line set and 1 another line is added, after one iteration, the number of lines is decreased for 1. Thus the while loop will be executed for at most 2n times.

This function is then O(n^4logn)*O(n) = O(n^5logn)

In sum, the execution time of main function is the running time, which contains initialization, executing the local optimization and outputting the file.
Running Time = O(nlogn) + O(n^5logn) + O(n) = O(n^5logn)

3. One instance for each algorithm on which the algorithm fails to return the optimum solution. Show the run of the algorithm on the instance, and show a better solution.

Take one instance:

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

The location optimization outputs the solution as:

```
6
v 2.5
v 5.5
v 6.5
v 8.5
h 2.5
h 6.5
```

While the global optimization outputs the solution as:

```
5
v 2.5
v 5.5
v 7.5
h 3.5
h 7.5
```

After initialization the lines are:

```
v1.5
v2.5
v3.5
v4.5
v5.5
v6.5
v7.5
v8.5
v9.5
```

The program printed out the subtraction and addition of lines during the iteration of local optimization:

```
subtract v1.5 & v3.5; add h2.5
subtract v4.5 & v6.5; add h6.5
subtract v7.5 & v9.5; add v6.5
```