9COL 215: SOFTWARE ASSIGNMENT 2

REPORT

Neha Kumari: 2023CS50205

Avani Komalkar : 2023CS50325

Problem Statement

Given:

1. A set of rectangular logic gates g1,g2....gn
2. Width and height of each gate g_i
3. the input and output pin locations (x and y co-ordinates) on the boundary of each gate $g_i.p_1$, $g_i.p_2$, ..., $g_i.p_m$ (where gate g_i has m pins)
4. The pin-level connections between the gates.

   Write a program to assign locations to all gates in a plane so that:

1. No two gates are overlapping
2. The sum of estimated wire lengths for all wires in the whole circuit is minimized.

Approach

KEY IDEA:

- 0 <= length of a wire <= (len(bounding box of gates)/2
- 0 <= l(all wires) <= n* (len(bounding box of gates)/2
- So ans <= n* (len(bounding box of gates)/2
- To minimize answer we minimize (len(bounding box of gates) => hence solution is Assignment 1
- Improvising it, we clubbed gates whose pin is connected with same wire to make their bounding box and then packed them efficiently.

Gate Representation and Initialization:

1. *Classes: Pin and Gates:*
   - Each gate is initialized by parsing input strings to extract its dimensions (width, height) and pins.

- Pins have coordinates (x, y) and a Boolean flag is_left to indicate whether they are on the left or right of the gate.
- Gates track the count of pins on the left and right for layout purposes.

Connections Between Pins:

2. *Class: Connections:*
   - The Connections class manages wire connections between pins on different gates.

Connections are formed between two pins, where each pin is identified by its gate number and pin number.

- The class stores connections in a list, where each element contains pairs of connected pins.
- The function cal_len_overall calculates the total wire length by summing up the Manhattan distances between the minimum and maximum coordinates of the connected pins. This function helps in assessing the compactness of the gate placement.

Grouping and Sorting Gates:

- First, we make a list say (outer list) which has a list of gates having same wire connection.
- Then, sort this list using key= len(internal list) and reverse =True such that we can placed the maximum number of gates connected via same wire first after bounding it as bounding box by using the assignment 1
- After placing this we made another bounding box of the next internal list of outer lists removing all the gates which I have already placed and process goes on until we covered all the internal list of connection.l (outer list)
- Treating this bounding box as a gate, we applied the assignment 1 one more time, and placed all of them in a compact manner

Time Complexity Analysis:

Class Pin:

- The __init__ method has a constant time complexity O(1) as it simply initializes instance variables.

2. Class Gates:

- The __init__ method:
    - Splitting the line1 string has a complexity of O(m) where m is the length of the string.
    - The loop for adding pins runs O(n) where n is the number of pins specified in line2. The complexity is O(n) for creating Pin instances and counting left pins.
    - Overall complexity: O(m+n).

3. Class Connections:

- The __init__ method has a constant time complexity O (1) since it only initializes an empty list.
- The new_connection method:
    - The split operations are O(p+q) where p and q are the lengths of the strings for the pins.
    - The nested loop iterates through existing connections, which can have a worst-case complexity of O(k·l) where k is the number of connection lists and l is the average length of each list. So overall complexity is O(p+q+k·l).
- The print _connections method has a complexity of O(k) where k is the number of connections.
- The cal_len_overall method:
    - The loop iterates over the list of connections, which has a length of O(k). For each connection, finding min and max over its pins takes O(r) where r is the number of pins in the connection. So the overall complexity is O(k·r).

4. Class Clubed_gates:

- The __init__ method has a constant time complexity O(1).

5. Function read_gate_dimensions:

- The first loop iterating through list_of_gates has a complexity of O(n) where n is the number of gates.
- The subsequent sorting operations, including calls to sorted(), are ￼O(nlogn).
- All loops which we used seems that it has multiple loops so time complexity would be n^3 or something, but my outer loop will iterates i< n and in the inner loop we increase i by i++ so, overall time complexity will be O(n) only

6. Input Output file:

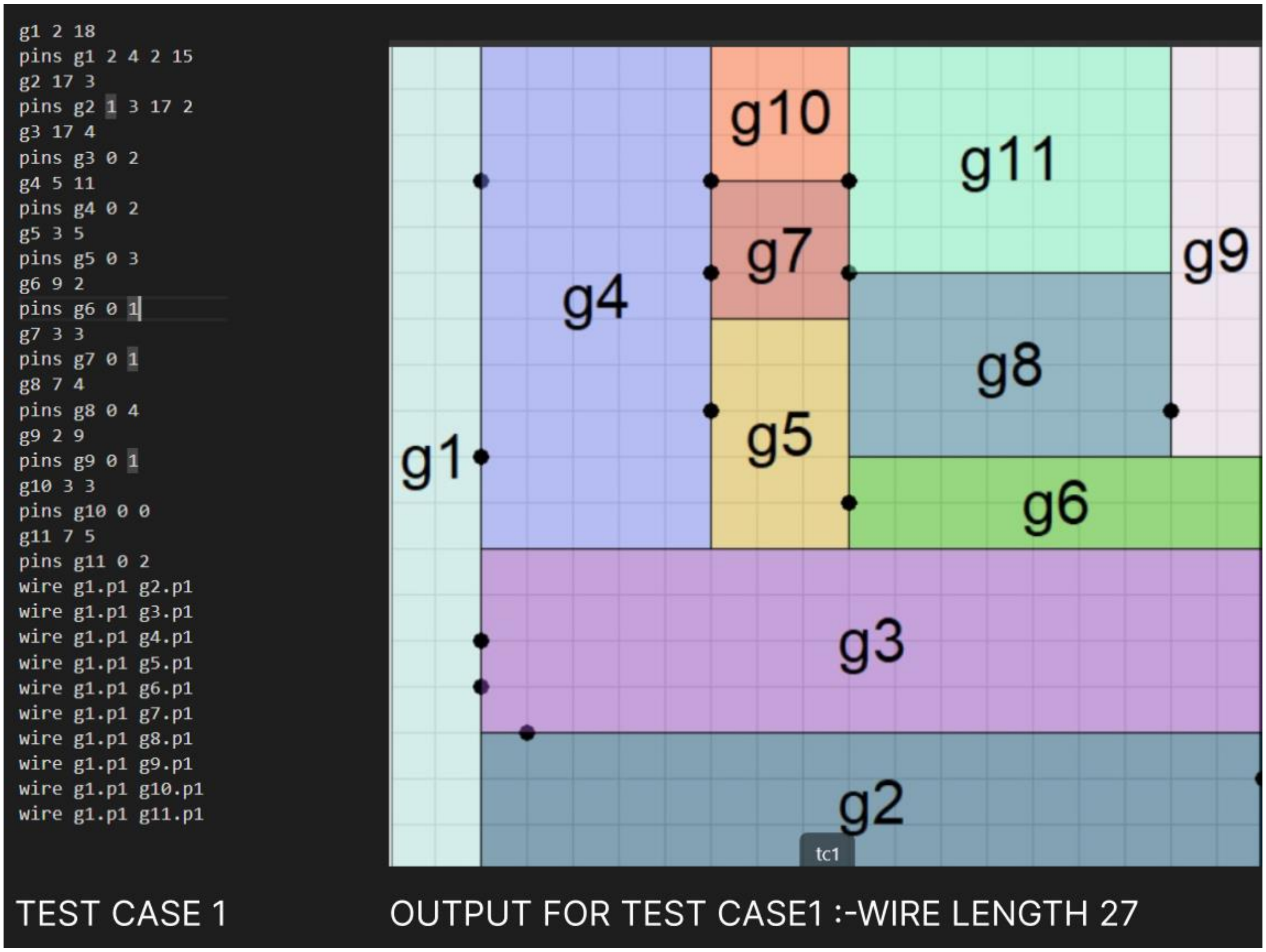
- Line 7: no. Of lines in input text O (l * avg length of string in each line n) => O(l*n)
- Line 24: O (no of gates)
- Line 28, 29: O (w^2 log(w) where w is no, of times wire written in input file
- Line 40: O (g log(g)) where g is total number of gates
- Line 44 to 48: O(g) where g is total number of gates
- Cal_len_overall has time complexity O(w)
- Read_gate_dimension has time complexity O(nlog(n)
- Line 64 to 70: O(g)
- New_connections O(w) where w is total number of times wire is being written in input file

OVER ALL TIME COMPLEXITY:

O(l*n) + O (w^2 log(w) +O (g log(g)

TEST CASES:

Test case 1



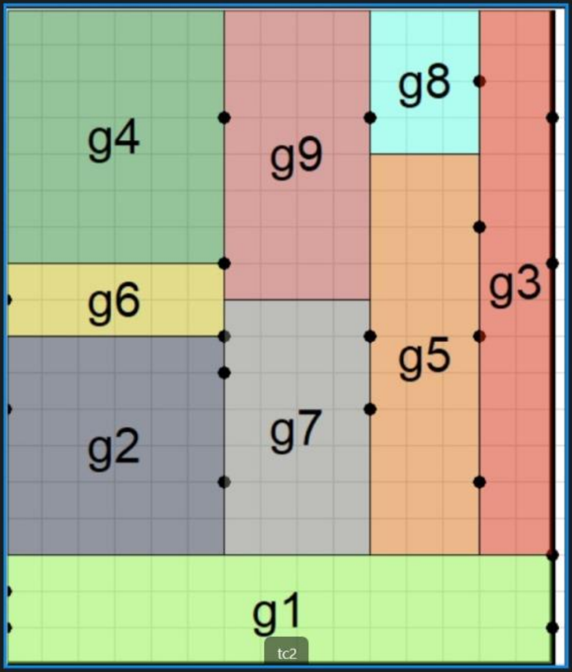
```
g1 2 18
pins g1 2 4 2 15
g2 17 3
pins g2 1 3 17 2
g3 17 4
pins g3 0 2
g4 5 11
pins g4 0 2
g5 3 5
pins g5 0 3
g6 9 2
pins g6 0 1
g7 3 3
pins g7 0 1
g8 7 4
pins g8 0 4
g9 2 9
pins g9 0 1
g10 3 3
pins g10 0 0
g11 7 5
pins g11 0 2
wire g1.p1 g2.p1
wire g1.p1 g3.p1
wire g1.p1 g4.p1
wire g1.p1 g5.p1
wire g1.p1 g6.p1
wire g1.p1 g7.p1
wire g1.p1 g8.p1
wire g1.p1 g9.p1
wire g1.p1 g10.p1
wire g1.p1 g11.p1
```

TEST CASE 1

OUTPUT FOR TEST CASE1 :-WIRE LENGTH 27

Test case 2

```
g1 15 3
pins g1 0 1 0 2 15 3 15 1
g2 6 6
pins g2 0 4 6 2
g3 2 15
pins g3 0 2 0 9 2 12 2 8
g4 6 7
pins g4 6 4
g5 3 11
pins g5 0 6 0 4 3 6
g6 6 2
pins g6 0 1 6 0
g7 4 7
pins g7 0 5 4 4
g8 3 4
pins g8 0 1 3 2
g9 4 8
pins g9 0 1 0 5
wire g1.p4 g2.p2
wire g2.p2 g7.p2
wire g7.p2 g5.p3
wire g5.p3 g3.p3
wire g3.p3 g9.p1
wire g3.p3 g8.p1
wire g5.p3 g6.p2
wire g2.p2 g4.p1
```



TEST CASE 2

OUTPUT OF TEST CASE2 :- WIRE LENGTH 23