CS29003 ALGORITHMS LABORATORY
ASSIGNMENT 5
Date: 16th Sep, 2021

Important Instructions

1. **Input:** To be taken from command line
2. **Format fo files to be submitted to Moodle and HackerRank:** ROLLNO_A5_p1.c, ROLLNO_A5_p2.c and ROLLNO_A5_p3.c/.cpp
3. Take inputs from command line in the specified format
4. You are to **stick to the file input output formats strictly** as per the instructions.
5. Submission through **.zip files are not allowed**.
6. Write your name and roll number at the beginning of your program.
7. Do not use any global variable unless you are explicitly instructed so.
8. Use proper indentation in your code.
9. Please follow all the guidelines. Failing to do so will cause you to lose marks.
10. **There will be part marking.**

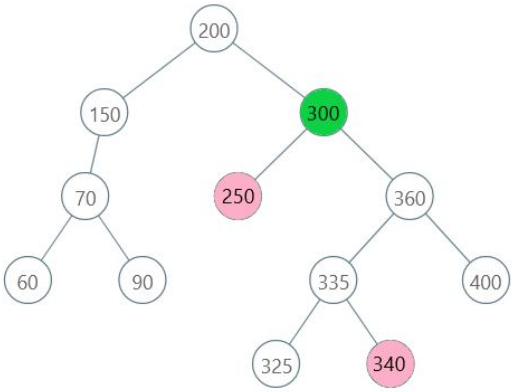
Binary Trees

Part 1: Help Luke and Leia escape! (60 marks)

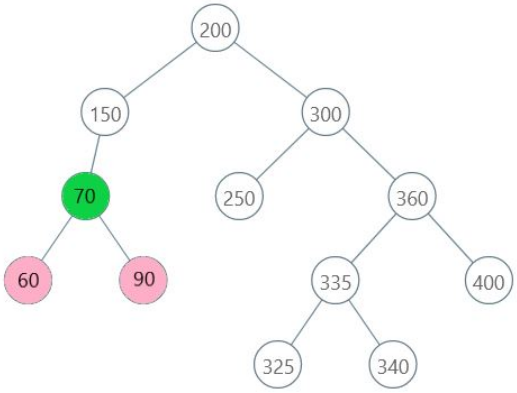
Welcome to the DeathStar! Princess Leia and Luke Skywalker are stuck on the deathstar and are trying to get out of the spaceship. They have been separated and are on different floors and rooms of the spaceship. Your job, as Yoda, is to guide them and make sure they escape the ship before it bursts!

Deathstar has been destroyed and is now shaped like a cone. The top level has a single room and from each room you can climb down to two rooms on either side (unless the room has been destroyed, of course). Leia and Luke have been separated and Yoda needs to help them meet and escape. Their strategy has two steps:

- ▷ Luke and Leia first need to rendevous at a common point before they can escape. However, they can only climb floors and cannot go down as the death star is burning from the bottom. As Yoda, who knows and sees all, you can see their position at all times, so you have to guide them to the place which would be closest to them both. This closest place is defined as the lowest room on the deathstar where both Luke and Leia can reach. You can refer to Figure 1 for examples. Each room has a value which defines the amount of damage the room has sustained. You have to tell Luke and Leia the room they need to meet at by telling them the damage the room has.
- ▷ Great! Now that Luke and Leia have met, they need to get out of the spaceship once and for all. But oh no! Darth Vader is there and he has some conditions:



(a) Luke and Leia are on 250 and 340 node. They can meet in room 300



(b) Luke and Leia are on 60 and 90 node. They can meet in room 70

Figure 1: Illustration: Luke and Leia meet

- He will only let Luke and Leia escape if the the total damage of the rooms on your floor is below a threshold. Yoda has to step in again! You have to first find the sum of the damage of the rooms on the level Luke and Leia meet at and then check against the threshold to see if they can escape or not.
- If the damage on the floor is above the threshold you still have a chance to escape. Keep travelling up until you find a room suitable to Darth Vaders' conditions.

NOTE: A few properties of the room: For each room, all rooms on lower level left side would contain lesser damage than the current room. Similarly, all rooms on lower level right side would contain higher damage than current room. **You have to utilize this property to solve the question. If the property is not utilized, marks would be deducted.**

You have to use the following way to represent a room:

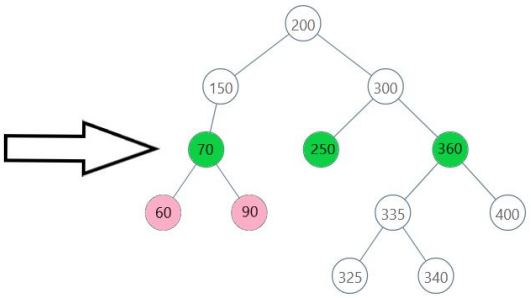
```
typedef struct room {
   int damage;
   struct room *left;
   struct room *right;
}
```

Input constraints:

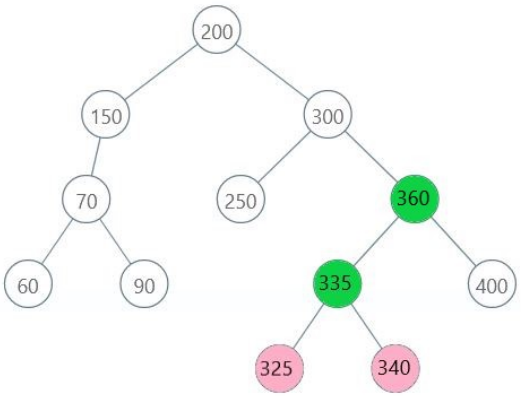
$$\text{Number of total rooms} = [2, 10^5]$$

$$-10^9 \leqslant \text{damage} \leqslant 10^9$$

$$\text{All node values are unique}$$



(a) Luke and Leia are on 60 and 90 node. They can meet in room 70 on level 2. Sum of rooms on level 2 is 70+250+360=680



(b) Luke and Leia are on 325 and 340 node. They can meet in room 335 on level 3. However, sum of rooms on that level is below threshold, so they can only escape from room

Figure 2: Illustration: Luke and Liea Escape Plan

Sample Input

FILE: *input-part1.txt*

```
4
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
250 340 600
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
60 90 50
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
325 340 800
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
60 800 350
```

The first line of the input file contains number of possible scenarios. For each scenario, the first line would contain the number of elements used to represent deathstar. Next line would contain the representation of the deathstar. The third line contains 3 integers: the position of Luke and Leia and the threshold.

Starter code to create a tree from the input would be provided to you. Starter code would give the root node of the tree as output which you have to use in your program.

Sample Output

FILE: *output-part1.txt*

```
Case #1: 300 1 450 Escaped! 300
Case #2: 70 2 680 Trapped!
Case #3: 335 3 885 Escaped! 360
Case #4: -1
```

Explanation: Each scenario begins with the Case#{scenario number}. For each case, each line starts with 3 integers. First integer represents the damage in the room Luke and Leia meet. Second integer is the level of the room (Top level is 0). Third integer is the sum of the damage on the level. Next is a string either Escaped! or Trapped!. If they escape, it contains details of the room they escape from.

For the first example, the common room that Luke and Leia can meet at is 300 which is on level 1 and there are 2 rooms on that level, with a total damage of 150 + 300 = 450< 600. They are able to escape from room 300. For the second example, the common room that Luke and Leia can meet at is 70. Room 70 is on level 2. There are 3 rooms on level 1, with a total damage: 70 + 250 + 360 = 680. However, 680>50, so they cannot escape. If they traverse upwards, the sum of rooms on level 1 is 450>50. Sum of damage of rooms on level 0 is 200>50. So, Luke and Lea are Trapped. For the third example, they meet in room 335 which is on level 3. Sum of damage of rooms is 1185>800. So they traverse up. Level 2 has sum 680<800, so they can escape from room 360. For the fourth example, Leia is not present on deathstar so output is -1.

Part 2: Yoda is dead (40 marks)

Yoda has been killed! And now Hans Solo is the only one who can save everyone. However, Hans Solo is not as powerful as Yoda so he cannot monitor all the rooms. He can however, move through the deathstar and install sensors in rooms which can monitor the damage. One sensor in a room can monitor the damage to itself, its immediate children and its parent. Budget is tight, and even Star Wars characters have to save money, so Hans needs to install as minimum number of sensors as possible. You have to return the minimum number of sensors that can monitor the entire deathstar. To help You out, we have provided with a basic understanding of the solution approach below:

Consider covering the deathstar from a bottom-up approach. We'll try placing our sensors from the deepest nodes and working our way up. We will work by using the following statements (you can verify that they are always true):

- ▷ If for a particular node, its children are covered, and it has a parent, then it's always better to place the sensor at this node's parent.
- ▷ If a node has children that are not covered, then we must place a camera here.
- ▷ If a node has no parent and it is not covered, we must place a camera here.

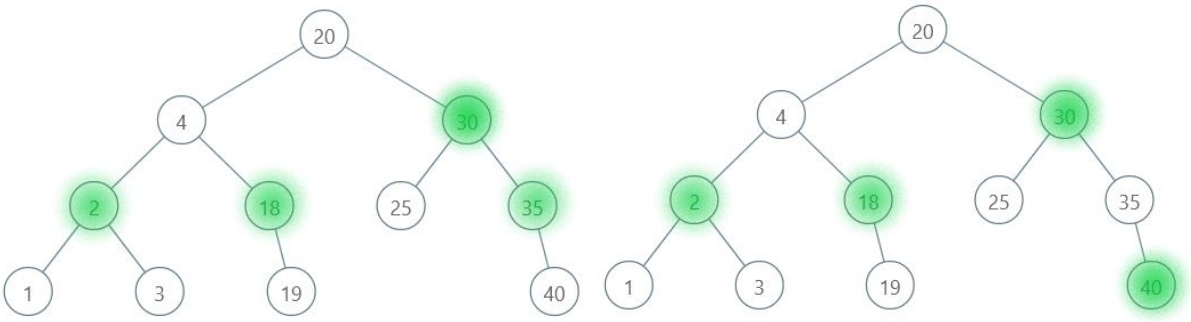


Figure 3: As can be seen from this example (first example in input file), there can be multiple places where the sensors might be placed, you are only required to output the minimum number of sensors

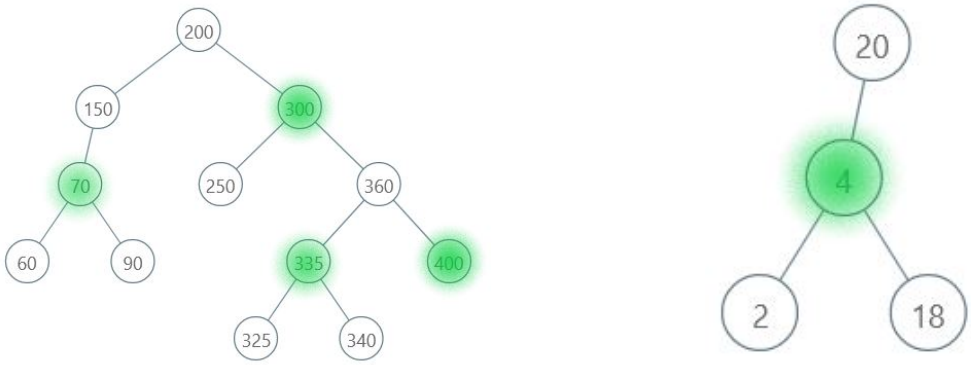


Figure 4: Placement of sensors for Example 2 and 3 given in the input file 3

```
Algorithm:
    func minSensorRequired(struct room topRoom) {
        int sensors = 0;
        covered = [];
        dfs(root)
        return sensors;
    }

    func dfs(struct room currRoom, parent = None) {
        Base case where if node does not exist, return
        left = dfs(currRoom->left, currRoom);
        right = dfs(currRoom->right, currRoom);
        if(parent is None and currRoom is not in covered or
        left child not covered or right child child not covered)
            sensors++;
            covered.append([currRoom, parent, left child, right child])
    }
```

Sample Input

FILE: *input-part2.txt*

```
3
15
20 4 30 2 18 25 35 1 3 -1 19 -1 -1 -1 40
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
5
20 4 -1 2 18
```

The input would first contain an integer describing the number of scenarios. For each scenario, the first line would contain the number of elements used to represent deathstar. Next line would contain the representation of the deathstar.

Sample Output

FILE: *output-part2.txt*

```
Case #1: 4
Case #2: 4
Case #3: 1
```

To test your own code, take the input from command line similar to formatting specified in the test cases. HackerRank will have some test cases so you can directly submit the code in Hackerrank to run on those test cases. HackerRank test cases are only provided to check the correctness of the code partially. You have to create additional test cases to handle corner cases (if any) and check the efficiency of the code.

Bonus: Luke and Leia cannot be found (30 marks)

In Part 1, Yoda printed "-1" whenever Luke or Leia could not be found and moved on. However, now he tries harder. He finds that a player cannot be seen only when his room is too damaged and has been hidden. To discover the room, you have to find the position it might have been at and then go to its parent. If Luke's room is hidden, Leia will use the parent room and traverse to it. You have to help find the parent room and the path to traverse from one player to this parent room.

Sample Input

FILE: *input-bonus.txt*

```
3
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
180 335
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
180 250
19
200 150 300 70 -1 250 360 60 90 -1 -1 335 400 -1 -1 -1 -1 325 340
180 400
```

The input would first contain an integer describing the number of scenarios. For each scenario, the line would contain the representation of the deathstar. Next line would contain the last known positions of Luke and Leia

Sample Output

FILE: *output-bonus.txt*

```
Case #1: 150
360 300 200 150
Case #2: 150
300 200 150
Case #3: -1
```

Here for case 1, possible position of Luke and Leia are 180 and 335. Since 180 is not in the tree, 150 would be its closest parent if 180 did exist in the tree. And the path from 335 to 150 would be 360 - 300 - 200 - 150. for case 2, possible position of Luke and Leia are 180 and 250. Since 180 is not in the tree, 150 would be it's closest parent if 180 did exist in the tree. And the path from 250 to 150 would be 300 - 200 - 150. For example 3, both luke and leia are hidden, so one cannot rescue the other. So, we output -1.

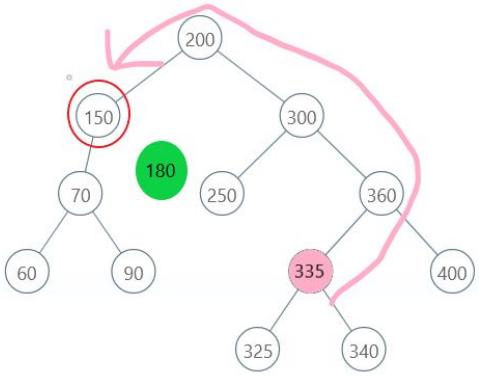


Figure 5: Here possible position of Luke and Leia are 180 and 335. Since 180 is not in the tree, 150 would be its parent if 180 did exist in the tree. And the path from 335 to 150 would be 360 - 300 - 200 - 150