Solution for Assignment 4

Hanzhe Wu

June 6, 2022

Contents

1 Problem 1: Symmetric Binary Tree 2

2 Problem 2: Non-repeating Numbers 2

3 Problem 3: Wandering 2

4 Problem 4: Pandemic 2

1 Problem 1: Symmetric Binary Tree

To find the maximum size of the symmetric binary trees, we can first find out the size of subtree rooted at each node u, and then check whether that subtree is symmetric.
To find out the size of subtrees, we can use DFS. As the tree is binary, the size of subtree rooted at node u is the sum of the size of its left and right subtree plus 1.
To check whether the subtree rooted at a certain node is symmetric, we only need to compare whether the subtrees rooted at its left child and its right child are symmetric to each other.
To check whether two subtrees rooted at nodes v and w are symmetric to each other, we can show that we only need to guarantee that the subtrees rooted at v and w are both empty, *or* the following statements are *all* true:

- the subtrees rooted at v and w are neither empty.
- the weights at nodes v and w are the same.
- the left subtree of subtree rooted at v is equal to the right subtree of subtree rooted at w.
- the right subtree of subtree rooted at v is equal to the left subtree of subtree rooted at w.

For the last two conditions, we need to check them recursively.

2 Problem 2: Non-repeating Numbers

We can use hashing to solve the problem.
For each test case, when inputting a number, we first search in the hash table to see whether it has appeared previously. If not, we output the number and add it to the hash table.
Remember to clear up the hash table for new test case.
There are many other ways to solve the problem, and some data structures in the libraries may help you to simplify the process.
The time complexity for search(and then adding) an element is expected to be $\mathcal{O}(1)$, so the total time complexity is $\mathcal{O}(Tn)$.

3 Problem 3: Wandering

We can modify the Dijkstra's algorithm to find the strictly second shortest path.
Each time we try to relax an edge from u to v, consider the following situations:

- If the distance of the shortest path to u plus the distance from u to v is smaller than the distance of the orginal shortest path to v, then we set the orginial shortest path to be the new *second* shortest path, *and* set the the distance of the shortest path to be the new distance.
- If the distance of the shortest path to u plus the distance from u to v is smaller than the distance of the orginal *second* shortest path to v, *but greater* than the distance of the orginal shortest path to v, then we set the the distance of the *second* shortest path to be the new distance.
- If the distance of the *second* shortest path to u plus the distance from u to v is smaller than the distance of the orginal *second* shortest path to v, then we set the distance of the *second* shortest path to be the new distance.

The other parts are similar to the Dijkstra's algorithm for the shortest path.

4 Problem 4: Pandemic

The problem can be transformed into a minimum spanning tree problem.
We can construct a graph by taking the planets as vertices and the airlines whose distance is no less than C as edges. Then we can run Kruskal's or Prim's algorithm to solve the problem.
The time complexity is $\mathcal{O}(N^2)$ or $\mathcal{O}(N^2 \log N)$ (for Prim's), or $\mathcal{O}(N^2 \log N)$ (for Kruskal's). Notice that the number of edges is $\mathcal{O}(N^2)$ in the worst case.