Laboratory work # 5

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114

Mail: jhlxhrq@163.com

Problem # 1067. Disk Tree

Screenshot from Timus:

9849303	07:51:38 25 Apr 2022	hduads2022_20321114	1067. Disk Tree	Java 1.8	Accepted		0.156	7 796 KB

Explanation of algorithm:

1. For the documents are saved using the data structure of the tree. So I decide to utilize the TreeMap to store the data.

2. After I split each row of the path, I push them into the tree map.

3. Find the entry of the documents, then traverse map tree and format the display, finally we get the answer.

Computational complexity of algorithm:

$$O(KN)$$

Source code:

```
import java.io.*;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

public class DiskTree {
    public static void main(String[] args) throws IOException {
        new DiskTree().run();
    }

    StreamTokenizer in;
    static PrintWriter out;
    static int SIZE;
```

```
    static Node[] NODES;

    void run() throws IOException {
        in = new StreamTokenizer(new BufferedReader(new
InputStreamReader(System.in)));
        out = new PrintWriter(System.out);
        prepare();
        preOrderTraversal(0, 0);
        out.flush();
    }

    int nextInt() throws IOException {
        in.nextToken();
        return (int) in.nval;

    }

    void prepare() throws IOException {
        SIZE = nextInt();
        NODES = new Node[20000];
        for (int i = 0; i < 20000; i++) {
            NODES[i] = new Node(new TreeMap<>());
        }

        in.wordChars(33, 45);
        in.wordChars(64, 64);
        in.wordChars(91, 96);
        in.wordChars(123, 126);
        in.ordinaryChars('0', '9');
        in.wordChars('0', '9');
        in.ordinaryChars('-', '-');
        in.wordChars('-', '-');

        int m;
        int counts = 0;
        for (int i = 0; i < SIZE; i++) {
            in.nextToken();
            String[] str = in.sval.split("\\\\");
            m = 0;
            for (String s : str) {
                if (!NODES[m].getTreeMap().containsKey(s)) {
                    NODES[m].addMap(s, ++counts);
                }
                m = NODES[m].getTreeMap().get(s);
            }
        }
    }

    void preOrderTraversal(int m, int hierarchy) {
        Set<Map.Entry<String, Integer>> set =
NODES[m].getTreeMap().entrySet();
        for (Map.Entry<String, Integer> entry : set) {
            for (int i = 0; i < hierarchy; i++) {
                out.print(" ");
            }
            out.print(entry.getKey() + "\n");
            preOrderTraversal(entry.getValue(), hierarchy + 1);
        }
    }
}

class Node {
    private final TreeMap<String, Integer> treeMap;
```

```
    public Node(TreeMap<String, Integer> treeMap) {
        this.treeMap = treeMap;
    }

    public void addMap(String s, Integer i) {
        this.treeMap.put(s, i);
    }

    public TreeMap<String, Integer> getTreeMap() {
        return treeMap;
    }
}
```