레포트 #3

알고리즘 3분반, 32170578, 김산

소스코드

Initialize.java

파일 이름을 입력받아 각 영문자를 key, 빈도수를 value로 가진 map을 생성합니다.

```java
import java.util.Scanner;
import java.util.TreeMap;
import java.io.File;
import java.io.FileNotFoundException;

public class Initialize {
    public static TreeMap<Character, Integer> parseInput() {
        Scanner sc_file = new Scanner(System.in);
        System.out.print("Input file name : ");
        String path = sc_file.nextLine();
        String text = "";
        try{
            File file = new File(path);
            String abs_path = file.getAbsolutePath();
            System.out.println(abs_path);
            file = new File(abs_path);
            Scanner scan = new Scanner(file);
            while(scan.hasNextLine()){
                text += scan.nextLine();
            }
            scan.close();
        }catch (FileNotFoundException e) {
            System.out.println("Error : no such file or directory" + path);
            sc_file.close();
            return null;
        }

        sc_file.close();
        TreeMap<Character, Integer> map = new TreeMap<>();
        for (int i = 0; i < text.length(); i++) {
            Character c = text.charAt(i);
            Integer val = map.get(c);
            if(val != null) {
                map.put(c,val + 1);
            }
            else {
                map.put(c,1);
            }
        }
        return map;
    }
}
```

Huffman.java(Main함수)

Initialize클래스의 parseInput()함수의 리턴값으로 받은 map을 통해 huffman code를 생성합니다.

```java
import java.util.TreeMap;
import java.util.PriorityQueue;

public class Huffman {
    public static void main(String[] args) throws Exception {
        TreeMap<Character, Integer> map = Initialize.parseInput();
        Character[] key_arr = map.keySet().toArray(new Character[map.size()]);
        Integer[] value_arr = map.values().toArray(new Integer[map.size()]);

        System.out.print("Char : ");
        for (int i = 0; i < key_arr.length; i++) {
            System.out.print(String.format("%s ", key_arr[i]));
        }
        System.out.println();

        System.out.print("Freq : ");
        for (int i = 0; i < value_arr.length; i++) {
            System.out.print(String.format("%s ", value_arr[i]));
        }
        System.out.println();

        PriorityQueue<Node> q
            = new PriorityQueue<Node>(key_arr.length, new MyComparator());

        for(int i = 0; i < key_arr.length; i++) {
            Node hn = new Node();
            hn.ch = key_arr[i];
            hn.weight = value_arr[i];

            hn.left = null;
            hn.right = null;

            q.add(hn);
        }

        Node root = null;

        while(q.size() > 1) {
            Node x = q.peek();
            q.poll(); // Priority Queue의 최솟값을 poll합니다.

            Node y = q.peek();
            q.poll();

            Node f = new Node();
            f.weight = x.weight + y.weight;
            f.ch = '-';
            f.left = x;
            f.right = y;
            root = f;
            q.add(f);
        }
        printCode(root, "");
    }

    public static void printCode(Node root, String s) {
        if(root.left == null && root.right == null) {
            if(Character.isLetter(root.ch) || root.ch == ' '){
                System.out.println(root.ch + " : " + s);
                return;
            }
        }
        printCode(root.left, s + "0");
        printCode(root.right, s + "1");
    }
}
```

Tree.java

호프만 코드를 생성하기 위한 트리자료구조입니다.

```
public class Tree {
    public Node root;

    public void setRoot(Node node) {
        this.root = node;
    }

    public Node getRoot() {
        return root;
    }
}
```

Node.java

각 트리의 노드는 가중치(weight = frequency)와 문자를 멤버 변수로 가집니다.

```
public class Node {
    Integer     weight;
    Character   ch;
    Node        left;
    Node        right;
}
```

MyComparator.java

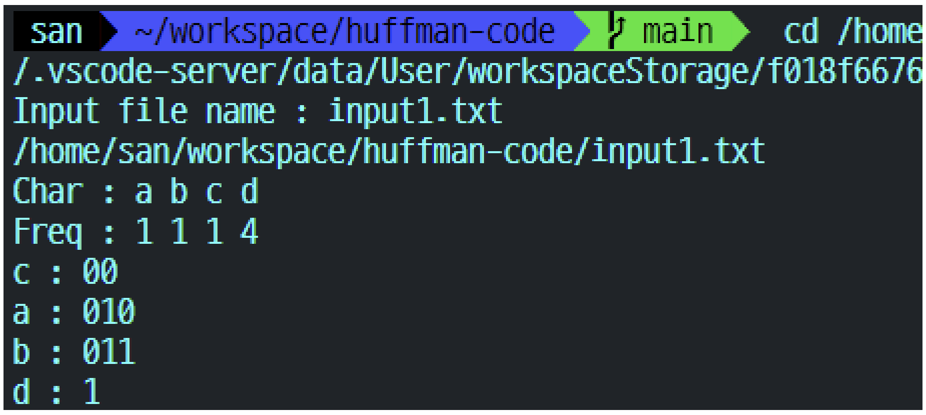
우선순위 큐의 비교 기준을 전달합니다.(가중치 값이 낮은 순으로 힙 구현)

```
import java.util.Comparator;
public class MyComparator implements Comparator<Node> {
    public int compare(Node x, Node y) {
        return x.weight - y.weight;
    }
}
```

실행결과

File 1

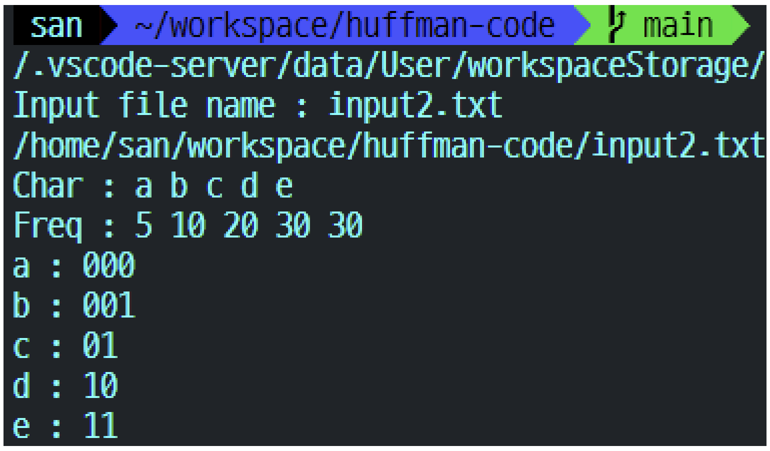
```
dbcddad
```



```
san  ~/workspace/huffman-code  main  cd /home
/.vscode-server/data/User/workspaceStorage/f018f6676
Input file name : input1.txt
/home/san/workspace/huffman-code/input1.txt
Char : a b c d
Freq : 1 1 1 4
c : 00
a : 010
b : 011
d : 1
```

File 2

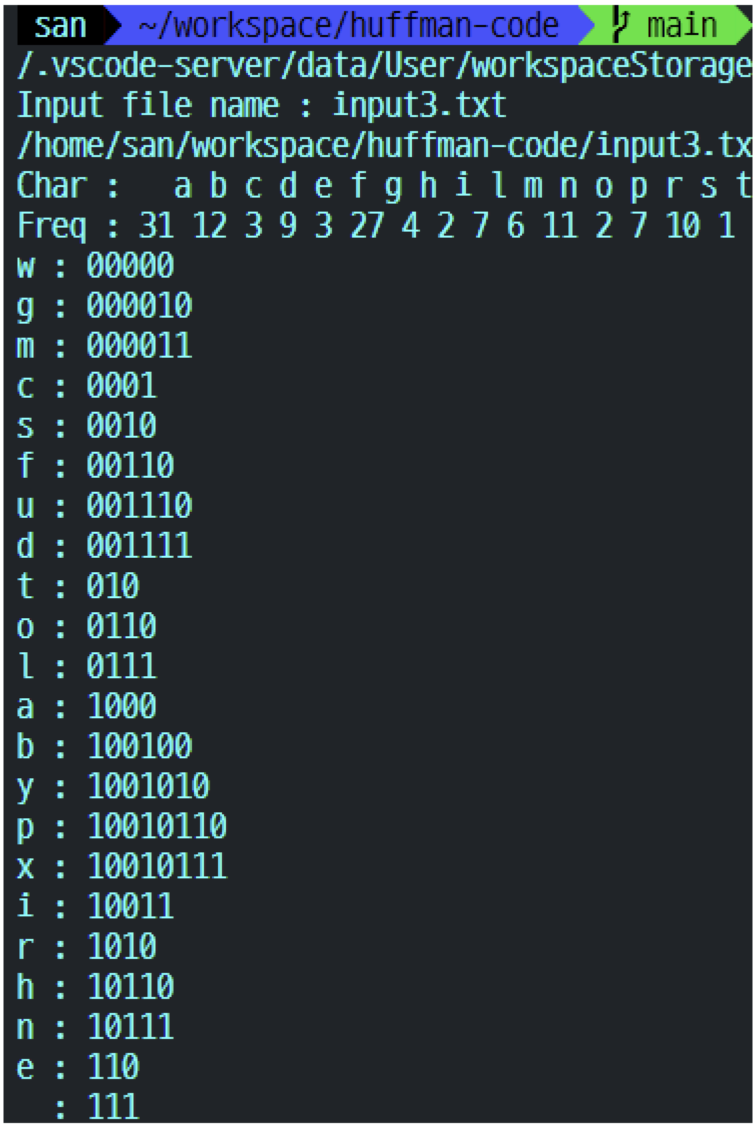
```
aaaaa
bbbbbbbbbb
cccccccccccccccccccc
dddddddddddddddddddddddddddddd
eeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```



```
san  ~/workspace/huffman-code  main
/.vscode-server/data/User/workspaceStorage/
Input file name : input2.txt
/home/san/workspace/huffman-code/input2.txt
Char : a b c d e
Freq : 5 10 20 30 30
a : 000
b : 001
c : 01
d : 10
e : 11
```

File 3

```
enter the text to create a huffman tree
the following characters will be used to create the tree
it must be consist of only lowercase letters and spaces
all other characters will be ignored
```



```
 san  ~/workspace/huffman-code  main
/.vscode-server/data/User/workspaceStorage
Input file name : input3.txt
/home/san/workspace/huffman-code/input3.tx
Char :   a b c d e f g h i l m n o p r s t
Freq : 31 12 3 9 3 27 4 2 7 6 11 2 7 10 1
w : 00000
g : 000010
m : 000011
c : 0001
s : 0010
f : 00110
u : 001110
d : 001111
t : 010
o : 0110
l : 0111
a : 1000
b : 100100
y : 1001010
p : 10010110
x : 10010111
i : 10011
r : 1010
h : 10110
n : 10111
e : 110
  : 111
```