



알고리즘 레포트#3

과 목 명	알고리즘
교 수	우 진 운
학 번	32163006
이 름	이 건 욱

1. Code

1-1. MinHeap

```
public class MinHeap {
    Element heap[];
    int size;

    MinHeap(int length) { // 생성자
        heap = new Element[length+1];
        for (int i = 0; i < length+1; i++) {
            heap[i] = new Element();
            heap[i].tree = new TreeNode();
        }
        size = 0;
    }

    public void insert(Element e) { // 삽입 함수
        int index = ++size;

        while ((index != 1) && (e.key < heap[index/2].key)) {
            heap[index] = heap[index/2];
            index /= 2;
        }
        heap[index] = e;
    }

    public Element remove() { // 삭제 함수
        int parent, child;
        Element data, tmp;

        if (size < 0)
            return null;
        data = heap[1];
        tmp = heap[size--];
        parent = 1;
        child = 2;

        while (child <= size) {
            if ((child < size) && (heap[child].key > heap[child+1].key))
                child++;
            if (tmp.key <= heap[child].key) break;
            heap[parent] = heap[child];
            parent = child;
            child *= 2;
        }
        heap[parent] = tmp;
        return data;
    }
}
```

1-2. Count

```
import java.io.FileReader;

public class Count {
    Huffman h = new Huffman();
    int i, size;
    char alphabet[] = new char[27];           // 알파벳 A~Z 저장하는 배열
    int count[] = new int[27];               // 각 알파벳 빈도수 저장하는 배열

    Count() {
        for (i = 0; i < 26; i++)
            alphabet[i] = (char)(i+97);
        alphabet[i] = ' ';

        for (i = 0; i < 27; i++)
            count[i] = 0;
        size = alphabet.length;
    }

    public void countFunc() throws Exception {    // file을 불러와서 count
        String path = "D:Test.txt";              // 파일 위치 설정
        FileReader fr = new FileReader(path);     // 파일 open

        i = 0;
        while((i = fr.read()) != -1) {
            char ch = (char)i;

            if (ch >= 97 && ch <= 122)
                count[i-97]++;
            else if (ch == 32)
                count[size-1]++;
            else
                continue;
        }
        fr.close();

        int tmp = 0;
        for (int j = 0; j < count.length-tmp; j++) {
            if (count[j] == 0) {
                for (int k = j; k < count.length-1; k++) {
                    count[k] = count[k+1];
                    alphabet[k] = alphabet[k+1];
                }
                j--;
                tmp++;
            }
        }
        size = size - tmp;
    }

    public void print() {                        // 출력함수
        for (i = 0; i < size; i++)              // 문자 출력
            System.out.printf("%5c", alphabet[i]);
        System.out.println();
        for (i = 0; i < size; i++)              // 빈도수 출력
            System.out.printf("%5d", count[i]);
        System.out.println();
    }
}
```

1-3. TreeNode

```
public class TreeNode {
    char symbol;           // 알파벳 저장
    int weight;            // 빈도수 저장
    TreeNode leftChild;    // 왼쪽 자식
    TreeNode rightChild;   // 오른쪽 자식
    TreeNode() {}          // 생성자
    TreeNode(TreeNode left, TreeNode right) {
        leftChild = left;
        rightChild = right;
    }
}
```

1-4. Huffman

```
public class Huffman {
    public TreeNode HuffmanTree() throws Exception {
        MinHeap mh = new MinHeap(27);
        Count cnt = new Count();
        Element a, b, c;

        cnt.countFunc();           // 알파벳 개수 카운트
        cnt.print();               // 카운트한 개수 출력

        for (int i = 0; i < cnt.size; i++) {
            TreeNode node = new TreeNode();
            node.symbol = cnt.alphabet[i];
            a = new Element();
            a.key = node.weight = cnt.count[i];
            a.tree = node;
            mh.insert(a);
        }

        for (int i = 1; i < cnt.size; i++) { // Huffman 트리 생성
            a = new Element();
            b = new Element(mh.remove());
            c = new Element(mh.remove());
            TreeNode p = new TreeNode(b.tree, c.tree);
            a.key = p.weight = b.key + c.key;
            a.tree = p;
            mh.insert(a);
        }
        a = new Element(mh.remove());
        return a.tree;
    }

    // Huffman 코드 출력 함수
    public void HuffmanPrint(TreeNode root, String str) {
        if (root == null)           // 빈 트리일 경우 return
            return;
        else if (root.leftChild == null && root.rightChild == null)
            System.out.println(root.symbol + ": " + str);
        else {
            String code = str;
            code += "0";
            HuffmanPrint(root.leftChild, code);
            code = str;
            code += "1";
            HuffmanPrint(root.rightChild, code);
        }
    }
}
```

1-5. Element

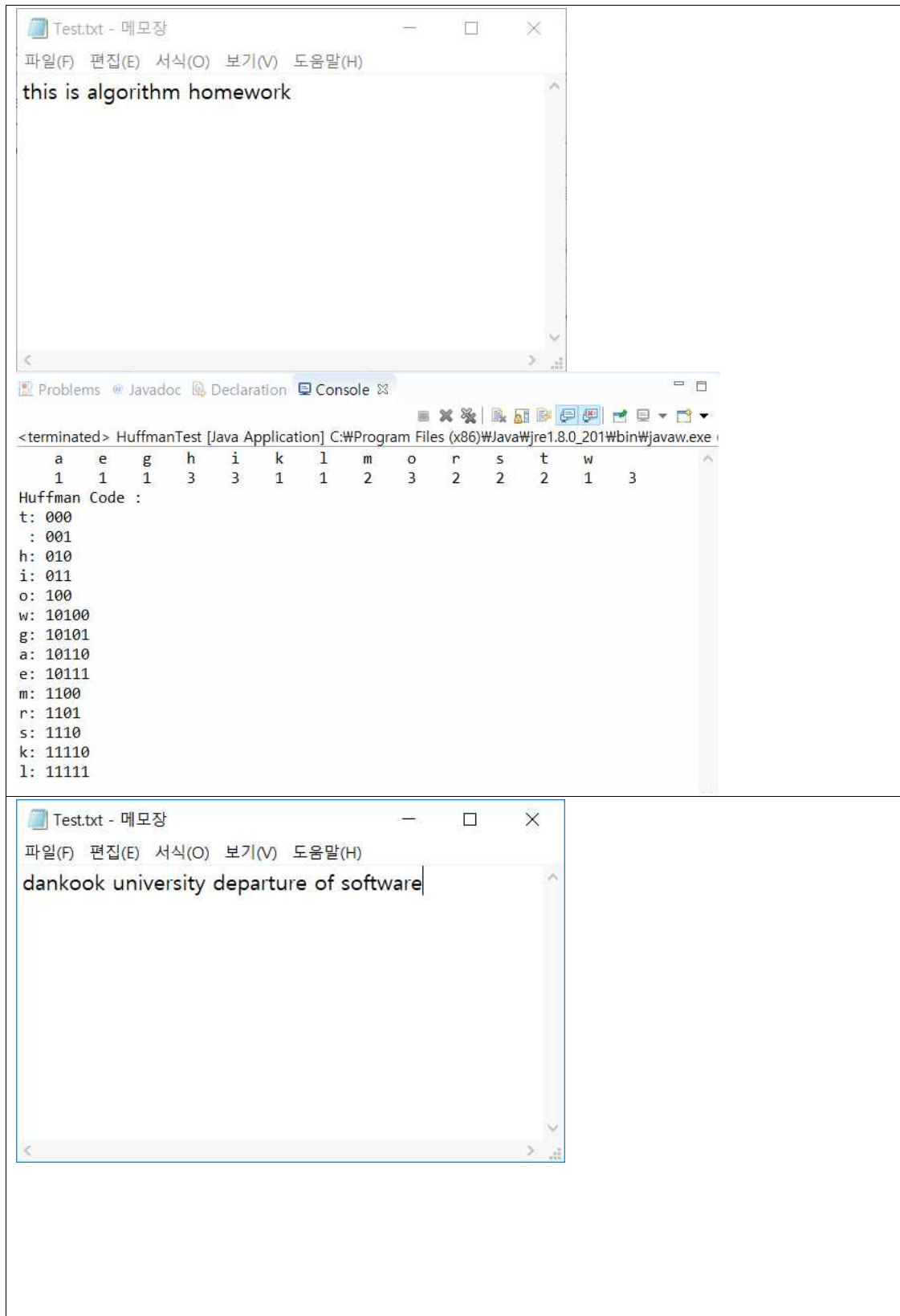
```
public class Element {
    TreeNode tree;
    int key;

    Element() {                // 생성자
        tree = null;           // tree 초기화
        key = 0;
    }
    Element(Element e) {       // 매개변수 갖는 생성자
        tree = e.tree;
        key = e.key;
    }
}
```

1-6. Huffman Test

```
public class HuffmanTest {
    public static void main(String args[]) throws Exception {
        Huffman h = new Huffman();
        String str = "";
        TreeNode root = new TreeNode();
        root = h.HuffmanTree();
        System.out.println("Huffman Code : "); // Huffman 트리 생성
        h.HuffmanPrint(root, str);             // Huffman 코드 출력
    }
}
```

2. 실행화면



```
Problems @ Javadoc Declaration Console
<terminated> HuffmanTest [Java Application] C:\Program Files (x86)\Java\jre1.8.0_201\bin\javaw.exe (2019. 5. 8. 오전 12:00)
a d e f i k n o p r s t u v w y
3 2 4 2 2 2 2 4 1 4 2 3 2 1 1 1 4
Huffman Code :
k: 0000
v: 00010
w: 00011
e: 001
o: 010
d: 0110
f: 0111
: 100
n: 1010
a: 1011
t: 1100
s: 11010
u: 11011
p: 111000
y: 111001
i: 11101
r: 1111
```

Test.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

this is simple test

```
Problems @ Javadoc Declaration Console
<terminated> HuffmanTest [Java Application] C:\Program Files (x86)\Java\jre1
e h i l m p s t
2 1 3 1 1 1 4 3 3
Huffman Code :
s: 00
e: 010
h: 0110
l: 0111
m: 1000
p: 1001
t: 101
: 110
i: 111
```