2019년 알고리즘

- HW 13 -

제출일자	2019.12.12.
이 름	장수훈
학 번	201402414
분 반	01

1 Prim

```
public class AdjList {
    ArrayList<Node> nodes;
}

public class Graph {
    int size;
    AdjList[] adjLists;
}

public Graph createGraph(int v) {
    Graph graph = new Graph();
    graph.size = v;
    graph.adjLists = new AdjList[v];
    for (int i = 0; i < v; i++) {
        AdjList adjList = new AdjList();
        adjList.nodes = new ArrayList<Node>();
        graph.adjLists[i] = adjList;
    }
    return graph;
}
```

그래프 클래스를 생성한 뒤 출발노드, 도착노드를 생성해 주었다.

```
public void getPrimsMST(Graph graph) {
    Node keys[] = new Node[graph.size];
    char parent[] = new char[graph.size];
    boolean mstSet[] = new boolean[graph.size];
    int sum = 0;

    for (int i = 0; i < graph.size; i++) {
        char c = (char) (i + 'a');
        keys[i] = new Node(c, Integer.MAX_VALUE);
        mstSet[i] = false;
    }
    keys[0].key = 0;
    PriorityQueue<Node> pQueue = new PriorityQueue<>();
    pQueue.addAll(Arrays.asList(keys));
    while (!pQueue.isEmpty()) {
        Node u = pQueue.remove();
        mstSet[u.vertice - 'a'] = true;


        for (Node v : graph.adjLists[u.vertice - 'a'].nodes) {

            if (mstSet[v.vertice - 'a'] == false && v.key < keys[v.vertice - 'a'].key) {
                pQueue.remove(keys[v.vertice - 'a']);
                keys[v.vertice - 'a'].key = v.key;
                parent[v.vertice - 'a'] = u.vertice;
                pQueue.add(keys[v.vertice - 'a']);
            }

        }
        sum = sum + keys[u.vertice - 'a'].key;
        System.out.println("w(" + parent[u.vertice - 'a'] + "," + keys[u.vertice - 'a'].vertice + ") = "
                + keys[u.vertice - 'a'].key);
    }
    System.out.println();
    System.out.println("w(MST) = " + sum);
}
```

시작 벡터를 제외한 벡터에는 max값을 넣고 시작벡터엔 0을 넣고 시작한다.
모든 벡터를 q에 넣고 큐가 비어있지 않는동안 q에서 빼셔 node u에 넣는다.
모든 노드들을 전부 검색해서 v가 q에 존재하고 w(u,v) < key[v]이면 q에서 v를 없애고 key[v] 에 w(u,v)를 넣고 v의 부모배열에 u를 넣어주는 식으로 했다.

```
fr = new FileReader("C:\\Users\\micke\\eclipse-workspace\\ALG\\src\\data13_prim.txt");
BufferedReader br = new BufferedReader(fr);
String line = br.readLine();
String[] array1 = new String[51];
int index = 0;
while (line != null) {
    StringTokenizer st = new StringTokenizer(line, ",");
    while (st.hasMoreTokens()) {
        array1[index] = st.nextToken();
        index++;
    }
    line = br.readLine();
}

String[] static_point = new String[9];
for (int i = 0; i < 9; i++) {
    static_point[i] = array1[i];
}
String[] array2 = new String[42];
for (int i = 9; i < 51; i++) {
    array2[i - 9] = array1[i];
}
String[] start_point = new String[14];
String[] end_point = new String[14];
int[] weight = new int[14];

for (int i = 0, k = 0; i < 14; i++, k = k + 3) {
    start_point[i] = array2[k];
    end_point[i] = array2[k+1];
    weight[i] = Integer.parseInt(array2[k+2]);
}

Prim MST = new Prim();
Graph graph = MST.createGraph(9);

for(int i =0; i < 14; i++) {
    MST.add(graph, start_point[i].charAt(0), end_point[i].charAt(0), weight[i]);
}

MST.getPrimsMST(graph);
```

txt 읽어오고 배열로 저장하는 건 기존의 코드를 사용하였다.

결과 화면

```
w( ,a) = 0
w(a,b) = 4
w(a,h) = 8
w(h,g) = 1
w(g,f) = 2
w(f,c) = 4
w(c,i) = 2
w(c,d) = 7
w(d,e) = 9

w(MST) = 37
```

1 Prim

```
class MinHeap {
    private ArrayList<Node> tree = new ArrayList<Node>(54);

    public MinHeap() {
        tree.add(null);
    }

    public void insert(Node n) {
        tree.add(n);

        int childt = tree.size() - 1;
        int parentt = childt / 2;

        while (parentt >= 1 && tree.get(childt).freq < tree.get(parentt).freq) {
            Collections.swap(tree, childt, parentt);

            childt = parentt;
            parentt = childt / 2;
        }
    }

    public boolean isEmpty() {
        return (tree.size() <= 1);
    }

    public Node extractMinNode() {
        if (isEmpty())
            return null;

        Node min = tree.get(1);

        int top = tree.size() - 1;

        tree.set(1, tree.get(top));
        tree.remove(top);

        int parentt = 1;
        int leftPos = parentt * 2;
        int rightPos = parentt * 2 + 1;

        while (leftPos <= tree.size() - 1) {
            int targetPos;
            if (rightPos > tree.size() - 1) {
                if (tree.get(leftPos).freq >= tree.get(parentt).freq)
                    break;
                targetPos = leftPos;
            } else {
                if (tree.get(leftPos).freq >= tree.get(parentt).freq
                        && tree.get(rightPos).freq >= tree.get(parentt).freq)
                    break;

                targetPos = (tree.get(leftPos).freq < tree.get(rightPos).freq) ? leftPos : rightPos;
            }

            Collections.swap(tree, targetPos, parentt);

            parentt = targetPos;
            leftPos = parentt * 2;
            rightPos = parentt * 2 + 1;
        }
        return min;
    }

    public void printTree() {
        for (Node n : tree)
            if (n != null)
                System.out.print(n.freq + " ");
        System.out.println("");
    }
}
```

우선 최소힙이다. 알파뱃 대소문자 총 52개 만큼 생성을 하고 첫 번째를 비워줬다. 이후 freq를 기준으로 heap을 구성하였다. 밑에 함수들은 heap이 비어있어면 반환, 최소노드 반환, 출력 함수등이 있다.

```
public class HuffmanTree {

    public static HashMap<Character, Integer> freq = new HashMap<Character, Integer>
    public static Node huffmanTree = null;

    public static void countAlphabetFrequency(String src) {
        try {

            BufferedReader in = new BufferedReader(new FileReader(src));
            String s;

            while ((s = in.readLine()) != null) {

                for (int i = 0; i < s.length(); i++) {
                    char c = s.charAt(i);
                    if (freq.containsKey(c))
                        freq.put(c, freq.get(c) + 1);
                    else
                        freq.put(c, 1);
                }
            }
            in.close();
        } catch (IOException e) {
            System.err.println(e);
            System.exit(1);
        }
    }

    public static void makeHuffmanTree() {
        MinHeap mh = new MinHeap();

        if (freq.isEmpty())
            return;

        for (char key : freq.keySet())
            mh.insert(new Node(freq.get(key), key));

        while (true) {

            Node leftChild = mh.extractMinNode();
            Node rightChild = mh.extractMinNode();

            huffmanTree = new Node(leftChild.freq + rightChild.freq, '.');

            huffmanTree.leftNode = leftChild;
            huffmanTree.rightNode = rightChild;

            if (mh.isEmpty()) {
                return;
            }
            mh.insert(huffmanTree);
        }
    }

    static String[] str2 = new String[6];
    static int count = 0;
    static char[] strI = new char[6];

    public static void printEachCharacterCode(Node htRoot, int[] trace, int top) {

        if (htRoot.leftNode != null) {
            trace[top] = 0;
            printEachCharacterCode(htRoot.leftNode, trace, top + 1);
        }
        if (htRoot.rightNode != null) {
            trace[top] = 1;
            printEachCharacterCode(htRoot.rightNode, trace, top + 1);
        }

        if (htRoot.leftNode == null && htRoot.rightNode == null)
        {
            strI[count] = htRoot.alphabet;
            str2[count] = printArr(trace, top);
            count++;
        }

    }

    public static String printArr(int[] arr, int top) {
        String temp = "";
        for (int i = 0; i < top; i++) {
            temp = temp + arr[i];
        }
        return temp;
    }
```

허프만트리 클래스는 알파벳 빈도수를 저장하는 해시맵을 만들어 최소힙에 각각의 빈도 수랑 알파벳을 저장했다. left를 탐색할 경우 0을 출력하고, right를 탐색할 경우 1을 출력하고, 단말 노드를 만났을 경우, 즉, left right 모두 null일 경우 단말 노드의 character를 출력했다.

```
public static void main(String[] args) throws IOException {
    countAlphabetFrequency("C:\\Users\\micke\\eclipse-workspace\\ALG\\src\\data13_huffman.txt");
    makeHuffmanTree();

    int[] arr = new int[freq.size() - 1];

    printEachCharacterCode(huffmanTree, arr, 0);

    BufferedOutputStream bs = null;

    try {
        bs = new BufferedOutputStream(
                new FileOutputStream("C:\\Users\\micke\\eclipse-workspace\\ALG\\src\\201402414_table.txt"));
        String str = "";
        for (int i = 0; i < 6; i++) {
            str = str + str1[i] + "," + str2[i] + "\n";
        }
        bs.write(str.getBytes());

    } catch (Exception e) {
        e.getStackTrace();
        // TODO: handle exception
    } finally {
        bs.close();
    }

    String change = "";
    FileReader fr;
    try {
        fr = new FileReader("C:\\Users\\micke\\eclipse-workspace\\ALG\\src\\data13_huffman.txt");
        BufferedReader br = new BufferedReader(fr);
        String line = br.readLine();

        char[] ch = new char[line.length()];
        for (int i = 0; i < line.length(); i++) {
            ch[i] = line.charAt(i);
        }

        for (int i = 0; i < line.length(); i++) {
            for (int j = 0; j < 6; j++) {
                if (ch[i] == str1[j]) {
                    change = change + str2[j];
                }
            }
        }

    } catch (IOException e) {

    }

    try {
        bs = new BufferedOutputStream(
                new FileOutputStream("C:\\Users\\micke\\eclipse-workspace\\ALG\\src\\201402414_encoded.txt"));

        bs.write(change.getBytes());

    } catch (Exception e) {
        e.getStackTrace();
        // TODO: handle exception
    } finally {
        bs.close();
    }
    System.out.println(change.length());
}
```

파일을 읽고 반환된 값을 다시 txt로 저장하는 부분이다.

결과화면

txt 파일 저장

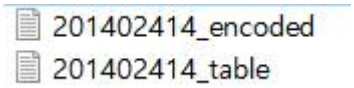
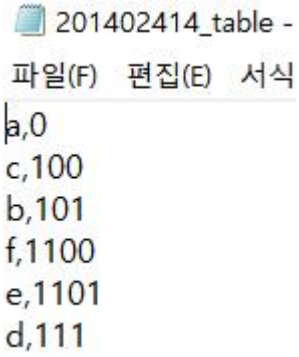


table txt저장



encoded txt저장