알고리즘

과제번호	09주차
날 짜	2018.11.08
학 번	201302395
이 름	류경빈

과제 1	Prim's algorithm을 통해 MST를 만드는 프로그램을 구현하라.

```java
public class Prim {
    private int[] key;
    private char[] nodeName;
    private int vectorSize;
    private final int INFINITE = Integer.MAX_VALUE;

    public Prim(int node) {
        this.vectorSize = node;
        this.key = new int[this.vectorSize];
        this.nodeName = new char[this.vectorSize];
    }

    public void shortestPath(int[][] w, int v) {
        Queue Q = new Queue(new Heap());
        for (int i = 0; i < this.vectorSize; i++) {
            if (i == v) {
                this.key[i] = 0; // key[s] <- 0
                this.nodeName[i] = ' ';
            } else {
                this.key[i] = INFINITE; // key[v] <- infinite
            }
            Q.insert(new Node(i, this.key[i])); // Q <- V
        }

        int total = 0;

        while (!Q.isEmpty()) { // Q != 공집합
            Node u = Q.extract_min(); // EXTRACT-MIN(Q)
            total += key[u.getVertex()]; //
            System.out.println("w(" + this.nodeName[u.getVertex()] + "," + (char)(u.getVertex() + 97) + ") = " + key[u.getVertex()]);
            // Q의 노드.. 즉 노드의 해당 vertex의 path를 갱신하는 과정.. 즉 선택한 vertex에서 경로를 정했고, 해당 경로가 정해져 weight를 가졌을 때..
            for (int i = 0; i < w[u.getVertex()].length; i++) { // 해당 노드, vertex의 모든 점들을 다 확인한다.
                if (w[u.getVertex()][i] != 0 && Q.hasVertex(i) && w[u.getVertex()][i] < key[i]) {
                    // 해당 vertex의 경로가 0일 경우는 edge가 없는 경우 && Q에 해당 i값을 가진 vertex가 있을 경우(경로가 정해지지 않은 vertex, Q에서 EXTRACT-MIN으로 나오지 않은)
                    // w[EXTRACT-MIN.getVertex(Q에서 최소 우선순위로 나온 값)][i]이 key의 i 값에 저장 된 값과 비교해서 작을 경우(처음엔 무한대여서 해당한다.)
                    key[i] = w[u.getVertex()][i]; // 해당하는 key[i]값을 w배열에 들어있던 값으로 바꿔준다.
                    nodeName[i] = (char)(u.getVertex() + 97);
                    Q.updateDistance(i, key[i]); // Q에 들어있는 nodes[i]의 path값을 갱신해준다.
                }
            }
        }
        System.out.println("\nw(MST) = " + total);
    }
}
```

- 기본적으로 Prim 알고리즘에서 필요로 하는 큐 Q, key(safetry vertex)값을 저장하는 배열, EXTRACT-MIN값을 가지고 있는 u를 기본 변수로 사용한다.

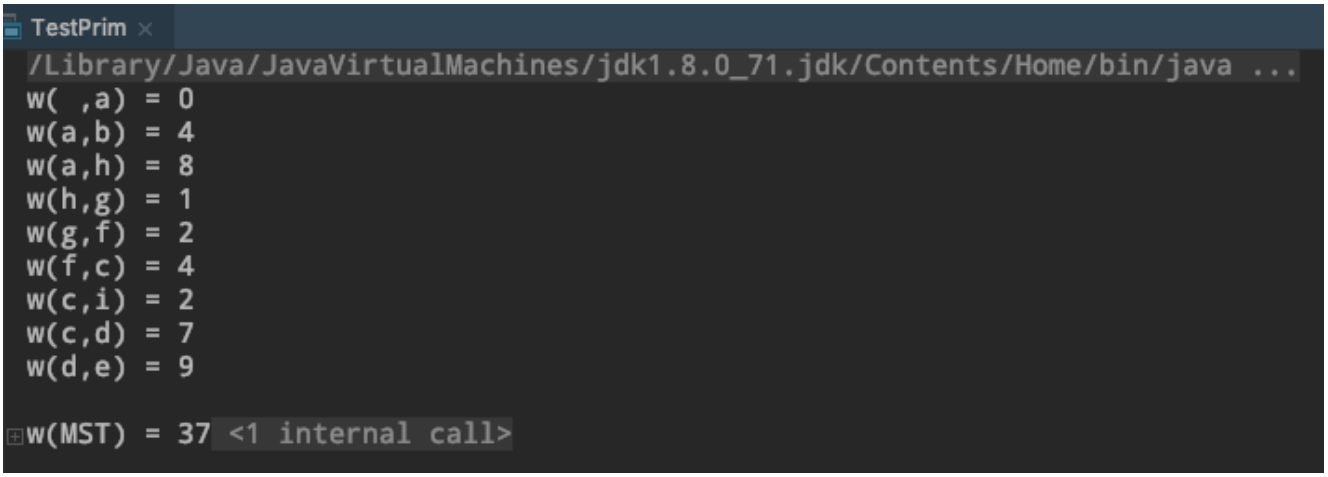
```java
public Node extract_min() {
    Node temp = this.nodes[1];
    this.nodes[1] = this.nodes[this.heapSize--];
    this.minHeap.setHeapSize(this.heapSize);
    this.minHeap.BUILD_MIN_HEAP(this.nodes);
    return temp;
}

public void updateDistance(int index, int path) {
    for (int i = 1; i <= heapSize; i++) {
        if (index == nodes[i].getVertex()) {
            nodes[i].setPath(path);
            this.minHeap.BUILD_MIN_HEAP(this.nodes);
            return;
        }
    }
}

public boolean hasVertex(int index) {
    for (int i = 1; i <= heapSize; i++) {
        if (index == nodes[i].getVertex()) {
            return true;
        }
    }
    return false;
}
```

- Queue의 구조는 기본적으로 nodes와 heapSize를 구조로 가지고 있다. Queue의 구조를 통해서 필요한 함수들은 대표적으로 extract_min():가장 작은 원소를 반환하고 삭제, updateDistance():nodes의 Path를 최신화, hasVertex():입력 받은 index값과 동일한 vertex를 확인해주는 함수들을 구현했다.

결과 화면


TestPrim
/Library/Java/JavaVirtualMachines/jdk1.8.0_71.jdk/Contents/Home/bin/java ...
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 internal call>