2019년 알고리즘

- HW 12 -

제출일자	2019.11.28.
이 름	장수훈
학 번	201402414
분 반	01

1 Dijkstra Algorithm

```
public static void main(String[] args) {

    FileReader fr;
    try {
        fr = new FileReader("C:\\Users\\micke\\eclipse-workspace\\ALG\\src\\data12.txt");
        BufferedReader br = new BufferedReader(fr);
        String line = br.readLine();
        int[] array1 = new int[32];
        int index = 0;
        while (line != null) {
            StringTokenizer st = new StringTokenizer(line, ",");
            while (st.hasMoreTokens()) {
                array1[index] = Integer.parseInt(st.nextToken());
                index++;
            }
            line = br.readLine();
        }
        int[] static_point = new int[5];
        for (int i = 0; i < 5; i++) {
            static_point[i] = array1[i];
        }
        int[] array2 = new int[27];
        for (int i = 5; i < 32; i++) {
            array2[i - 5] = array1[i];
        }
        int[] start_point = new int[9];
        int[] end_point = new int[9];
        int[] weight = new int[9];

        int index_count = 0;
        for (int i = 0; i < 27; i = i + 3) {
            start_point[index_count] = array2[i];
            end_point[index_count] = array2[i + 1];
            weight[index_count] = array2[i + 2];
            index_count++;
        }
        char[] S = new char[static_point.length];
        for (int i = 0; i < static_point.length; i++) {
            S[i] = alp(static_point[i]);
        }

        Dijkstra di = new Dijkstra(S);

        for (int i = 0; i < start_point.length; i++) {
            di.add(alp(start_point[i]), alp(end_point[i]), weight[i]);
        }
        di.Dijkstra_Alg(S);
```

입력받는 부분은 저번 과제랑 똑같다고 봐도 무방하다.

```
public Dijkstra(char[] static_arr) {
    int size = static_arr.length;
    Vertices = static_arr;
    Di_arr = new int[size][size];
    for (int i = 0; i < size; i++)
        for (int j = 0; j < size; j++) {
            Di_arr[i][j] = Integer.MAX_VALUE;
            if (i == j) {
                Di_arr[i][j] = 0;
            }
        }

}
```

Di_arr를 무한대로 초기화해준다 주면서 자기 자신으로 향할때는 0을 넣어주었다.

```
public void Dijkstra_Alg(char[] static_point) {

    int size = static_point.length;
    int[] Distance = new int[size];
    Distance[0] = 0;
    int count1 = 0;
    Point u, v;
    PriorityQueue<Point> pri = new PriorityQueue<>();

    for (int n = 1; n < size; n++) {
        Distance[n] = Integer.MAX_VALUE;
    }
    for (int n = 0; n < size; n++) {
        pri.add(new Point(Distance[n], Vertices[n]));
    }
    while (!pri.isEmpty()) {
        int count = 0;
        u = pri.remove();
        System.out.println("-----------------------------------------------------");
        System.out.println("S[" + count1++ + "] : " + "d[" + u.vertex + "] = " + u.dis);
        System.out.println("-----------------------------------------------------");

        Iterator<Point> it = pri.iterator();
        while (it.hasNext()) {
            v = it.next();
            int x = u.vertex - 63;
            int y = v.vertex - 63;
            System.out.print("Q[" + count++ + "] : " + "d[" + v.vertex + "] " + "= " + v.dis);
            if (u.dis + Di_arr[x][y] < v.dis && Di_arr[x][y] != Integer.MAX_VALUE) {
                v.dis = u.dis + Di_arr[x][y];
                System.out.print(" -> " + "d[" + v.vertex + "] " + "= " + v.dis);
            }
            System.out.println();
        }
        System.out.println();
        System.out.println();
```

1. d[v]를 무한대로 초기화한다. 단 d[s] =0이다.
2. 최소힙을 기반으로 하는 우선순위 큐 pri에 모든 점을 넣는다.
3. 추출한 노드 u를 Q에 남아있는 모든 노드 v와 비교하고 d[u]+w[u][v] < d[v]이면 d[v]를 갱신한다.
4. 큐가 공집합이 될 때까지 반복한다.

결과

```
-----------------------------------------------------
S[0] : d[A] = 0
-----------------------------------------------------
Q[0] : d[D] = 2147483647
Q[1] : d[E] = 2147483647
Q[2] : d[C] = 2147483647 -> d[C] = 3
Q[3] : d[B] = 2147483647 -> d[B] = 10


-----------------------------------------------------
S[1] : d[C] = 3
-----------------------------------------------------
Q[0] : d[B] = 10 -> d[B] = 7
Q[1] : d[E] = 2147483647 -> d[E] = 5
Q[2] : d[D] = 2147483647 -> d[D] = 11


-----------------------------------------------------
S[2] : d[E] = 5
-----------------------------------------------------
Q[0] : d[B] = 7
Q[1] : d[D] = 11


-----------------------------------------------------
S[3] : d[B] = 7
-----------------------------------------------------
Q[0] : d[D] = 11 -> d[D] = 9


-----------------------------------------------------
S[4] : d[D] = 9
-----------------------------------------------------
```