

알고리즘 9 주차 201902694-박재우

RANK	TEAM	SCORE	1-PRIMALGORITHM [2 POINTS]	2-KRUSKALALGORITHM [3 POINTS]	3-CRITICALPATH [5 POINTS]
19	201902694	10 1258	271 6 tries	297 2 tries	490 5 tries

Submissions			
time	problem	lang	result
11/02/20-22:17	3-CRITICALPATH	JAVA	CORRECT
11/02/20-21:34	3-CRITICALPATH	JAVA	WRONG-ANSWER
11/02/20-21:34	3-CRITICALPATH	JAVA	WRONG-ANSWER
11/02/20-21:33	3-CRITICALPATH	JAVA	COMPILER-ERROR
11/02/20-21:31	3-CRITICALPATH	JAVA	COMPILER-ERROR
11/02/20-21:07	3-CRITICALPATH	JAVA	WRONG-ANSWER
11/02/20-21:03	3-CRITICALPATH	JAVA	WRONG-ANSWER
11/02/20-19:05	2-KRUSKALALGORITHM	JAVA	CORRECT
11/02/20-18:48	2-KRUSKALALGORITHM	JAVA	RUN-ERROR
11/02/20-18:47	2-KRUSKALALGORITHM	JAVA	COMPILER-ERROR
11/02/20-18:38	1-PRIMALGORITHM	JAVA	CORRECT
11/02/20-18:34	1-PRIMALGORITHM	JAVA	WRONG-ANSWER
11/02/20-15:32	1-PRIMALGORITHM	JAVA	WRONG-ANSWER
11/02/20-15:10	1-PRIMALGORITHM	JAVA	WRONG-ANSWER
11/02/20-14:20	1-PRIMALGORITHM	JAVA	WRONG-ANSWER
11/02/20-14:15	1-PRIMALGORITHM	JAVA	WRONG-ANSWER
11/02/20-14:14	1-PRIMALGORITHM	JAVA	COMPILER-ERROR
11/02/20-14:11	1-PRIMALGORITHM	JAVA	COMPILER-ERROR

Clarifications
No clarifications.
Clarification Requests
No clarification request.
request clarification

문제 1

- 문제/목표
Prim을 사용하여 MST구하기
- 해결방법

```

public class prim {
    static int V,E;
    static boolean visit[];
    static ArrayList<Edge>[] graph;
    static ArrayList<Edge> MST;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        V = sc.nextInt();
        E = sc.nextInt();
        sc.nextLine();
        HashMap<String,Integer> map = new HashMap<>();
        String nodes[] = sc.nextLine().split(" ");
        for(int i=0;i<nodes.length;i++)
            map.put(nodes[i],i);
        visit = new boolean[V+1];
        graph = new ArrayList[V+1];
        for(int i=0; i<=V; i++)
    
```

```

        graph[i] = new ArrayList<>();
        MST = new ArrayList<>();
        for(int i=1; i<=E; i++) {
            String test[] = sc.nextLine().split(" ");
            graph[map.get(test[0])].add(new Edge(test[0], test[1], Integer.parseInt(test[2])));
            graph[map.get(test[1])].add(new Edge(test[1], test[0], Integer.parseInt(test[2])));
        }
        int point = 1;
        solve(point, map);
        int weight = 0;
        for(int i=0; i<MST.size(); i++) {
            weight+=MST.get(i).value;
            System.out.println("{ "+MST.get(i).begin + " " +MST.get(i).end+" "+MST.get(i).value+"}");
        }
        if(MST.size()!= map.size()-1)
            System.out.print(0);
        else
            System.out.print(weight);
    }

    private static void solve(int P, HashMap map) {
        PriorityQueue<Edge> pq = new PriorityQueue<>();
        Queue<Integer> queue = new LinkedList<>();
        queue.add(P);
        while(!queue.isEmpty()) {
            int now = queue.poll();
            visit[now] = true;
            for(Edge e : graph[now]) {
                if(!visit[(int) map.get(e.end)])
                    pq.add(e);
            }

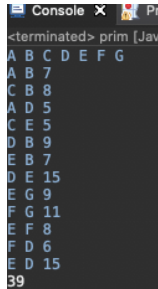
            while(!pq.isEmpty()) {
                Edge e = pq.poll();
                if(!visit[(int) map.get(e.end)]) {
                    queue.add((int) map.get(e.end));
                    visit[(int) map.get(e.end)] = true;
                    MST.add(e);
                    break;
                }
            }
        }
    }

    public static class Edge implements Comparable<Edge>{
        String begin;
        String end;
        int value;
        public Edge(String b, String e, int v) {
            this.begin = b;
            this.end = e;
            this.value = v;
        }
        @Override
        public int compareTo(Edge o) {
            // TODO Auto-generated method stub
            return this.value - o.value;
        }
    }
}

```

이론시간에 구현했던 prim을 이용하여 풀었다. 이론시간에는 입력을 전부 숫자로 해서 이를 크게 고치지 않고 사용하기위해 Map을 이용하여 입력받은 node의 값 개수만큼 순서대로 숫자로 할당해 주었다. 들어온 문자를 key값으로 순서대로 숫자로 value값을 설정해 주고 이를 대입했다.

- 결과



```
<terminated> prim [Java]
A B C D E F G
A B 7
C B 8
A D 5
C E 5
D B 9
E B 7
D E 15
E G 9
F G 11
E F 8
F D 6
E D 15
39
```

- 시간복잡도 : $O(n^2)$

- 자신만의 생각

Prim알고리즘에 익숙했다면 금방 풀었겠지만 수업을 듣고도 머리속에서 이해가 되지 않아 다시 구현하고 문제에 맞게 수정하는데에 2시간 넘게 걸렸다. 물론 이문제뿐만 아니라 동시에 Kruskal 도 해결하려고 해서 더 머리가 복잡했다.

- 난이도 : 4/5

문제 2

- 문제/목표

Kruskal을 사용하여 MST 구하기

- 해결방법

```
public class k {
    static int V, E;

    static PriorityQueue<Edge> pq;
    static ArrayList<Edge> MST;
    static int parent[];

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        V = sc.nextInt();
        E = sc.nextInt();
        sc.nextLine();
        HashMap<String,Integer> map = new HashMap<>();
        String nodes[] = sc.nextLine().split(" ");
        for(int i=0;i<nodes.length;i++)
            map.put(nodes[i],i);

        pq = new PriorityQueue<>();
        MST = new ArrayList<>();
        parent = new int[V+1];
    }
}
```

```

for(int i=0; i<=V; i++)
    parent[i] = i;
for(int i=1; i<=E; i++) {
    String test[] = sc.nextLine().split(" ");
    pq.add(new Edge(test[0], test[1], Integer.parseInt(test[2])));
}
Edge e;
while(MST.size() < (V-1) && (e = pq.poll()) != null) {
    if(find((int) map.get(e.begin)) != find((int) map.get(e.end))) {
        MST.add(e);
        union((int) map.get(e.begin), (int) map.get(e.end));
    }
}
int weight = 0;
for(int i=0; i<MST.size(); i++) {
    weight+=MST.get(i).val;
}
System.out.print(weight);
}

public static int find(int n) {
    if(n==parent[n])
        return n;
    else {
        int p = find(parent[n]);
        parent[n] = p;
        return p;
    }
}

public static void union(int n1, int n2) {
    int p1 = find(n1);
    int p2 = find(n2);
    if(p1!=p2) {
        parent[p1] = p2;
    }
}

public static class Edge implements Comparable<Edge>{
    String begin;
    String end;
    int val;
    public Edge(String b, String e, int v) {
        this.begin = b;
        this.end = e;
        this.val = v;
    }
    @Override
    public int compareTo(Edge o) {
        return this.val - o.val;
    }
}
}

```

이론시간에 구현했던 krusikal을 이용하여 풀었다. 이론시간에는 입력을 전부 숫자로 해서 이를 크게 고치지 않고 사용하기 위해 Map을 이용하여 입력받은 node의 값 개수만큼 순서대로 숫자로 할당해 주었다. 들어온 문자를 key값으로 순서대로 숫자로 value값을 설정해 주고 이를 대입했다. 이방법은 문제 1번과 같다.

- 결과

```

<terminated> prim [Java Application]
A B C D E F G
A B 7
C B 8
A D 5
C E 5
D B 9
E B 7
D E 15
E G 9
F G 11
E F 8
F D 6
E D 15
39

```

- 시간복잡도 : $O(e \log_2 e)$
- 자신만의 생각
위에서 언급한 대로 두가지 모두 전혀 익숙하지 않아 해결하는데 있어서 많은 어려움이 있었다. 기말은 어떻게 봐야할지 막막하다....
- 난이도 “ 4/5

문제 3

- 문제/목표
응답속도 최소를 보장하기 위한 가장 중요한 길이 몇개인지 출력하기
- 해결방법

```

package algorithm.week10;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.PriorityQueue;
import java.util.Scanner;

public class third_1 {

    static int V, E;

    static PriorityQueue<Edge> pq;
    static ArrayList<Edge> MST;
    static int parent[];
    static ArrayList<Integer> save = new ArrayList<>();

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        V = sc.nextInt();
        E = sc.nextInt();
        sc.nextLine();
        HashMap<String,Integer> map = new HashMap<>();
        String nodes[] = sc.nextLine().split(" ");
        for(int i=0;i<nodes.length;i++)
            map.put(nodes[i],i);
        ArrayList<String> tests = new ArrayList<>();
        for(int i=0; i<E; i++) {
            String t = sc.nextLine();
            String[] temp = t.split(" ");
            if(!tests.contains(t) && !tests.contains(temp[1]+" "+temp[0]+" "+temp[2]))
                tests.add(t);
        }
    }
}

```

```

String test[][] = new String[tests.size()][3];
for(int i=0; i<tests.size(); i++) {
    test[i] = tests.get(i).split(" ");
}
E = test.length;
pq = new PriorityQueue<>();
MST = new ArrayList<>();
parent = new int[V+1];

for(int i=0; i<=V; i++)
    parent[i] = i;
for(int i=0; i<test.length; i++) {
    pq.add(new Edge(test[i][0], test[i][1], Integer.parseInt(test[i][2])));
}
Edge es;
while(MST.size() < (V-1) && (es = pq.poll()) != null) {
    if(find((int) map.get(es.begin)) != find((int) map.get(es.end))) {
        MST.add(es);
        union((int) map.get(es.begin), (int) map.get(es.end));
    }
}
int min = 0;
for(int i=0; i<MST.size(); i++) {
    min+=MST.get(i).val;
}
String[] b = new String[MST.size()];
String[] ee = new String[MST.size()];
for(int i = 0; i<MST.size(); i++) {
    b[i] = MST.get(i).begin;
    ee[i] = MST.get(i).end;
}

//
for(int k=0; k<b.length; k++) {
    pq = new PriorityQueue<>();
    MST = new ArrayList<>();
    parent = new int[V+1];

    for(int i=0; i<=V; i++)
        parent[i] = i;
    for(int i=0; i<E; i++) {
        if(b[k]==test[i][0] && ee[k] == test[i][1]) i+=1;
        if(i == E)
            break;
        pq.add(new Edge(test[i][0], test[i][1], Integer.parseInt(test[i][2])));
    }
    Edge e;
    while(MST.size() < (V-1) && (e = pq.poll()) != null) {
        if(find((int) map.get(e.begin)) != find((int) map.get(e.end))) {
            MST.add(e);
            union((int) map.get(e.begin), (int) map.get(e.end));
        }
    }
    int weight = 0;
    for(int i=0; i<MST.size(); i++) {
        weight+=MST.get(i).val;
    }
    save.add(weight);
}
int imp=0;
for(int i=0; i<save.size(); i++) {
    if(min != save.get(i))
        imp++;
}System.out.print(imp);
}

public static int find(int n) {
    if(n==parent[n])
        return n;

```

```

        else {
            int p = find(parent[n]);
            parent[n] = p;
            return p;
        }
    }

    public static void union(int n1, int n2) {
        int p1 = find(n1);
        int p2 = find(n2);
        if(p1!=p2) {
            parent[p1] = p2;
        }
    }

    public static class Edge implements Comparable<Edge>{
        String begin;
        String end;
        int val;
        public Edge(String b, String e, int v) {
            this.begin = b;
            this.end = e;
            this.val = v;
        }
        @Override
        public int compareTo(Edge o) {
            return this.val - o.val;
        }
    }
}

```

Kruskal을 응용해서 풀었다.

먼저 입력받는 edge정보들 중 중복을 제거해서 저장을 한다.

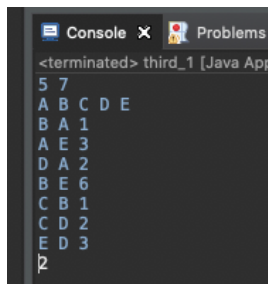
먼저 MST를 계산해서 MST의 간선과 값을 구해 저장한뒤에

MST edge에 해당 되는 간선들을 입력받은 여러개의 간선들 중에서 제거를 한뒤에

전체간선 -1 이 된 간선 개수로 다시 MST를 돌려 저장을 한다.

이 결과에서 나온 값을 맨처음 구한 값과 비교하여 보다 크면 중요한 간선, 작으면 node가 분리된 것 임으로 역시 중요한 간선이다.

- 결과



- 시간복잡도 : $O(e \log_2 e)$

- 자신만의 생각

여태까지 풀어왔던 알고리즘 문제중에 이문제가 가장 어려웠던것 같다..

총 5시간 넘게 걸렸다.

이런 어려운 문제보다는 이론을 이해할만한 적당한 난이도의 문제가 나왔으면 좋겠다.

- 난이도
1000000000/5!!!!