

## 1. 아이디어

Minimum spanning tree를 구하는 kruskal 알고리즘을 구현한다. 그래프의 모든 Edge를 weight의 오름차순으로 정렬한 뒤, cycle을 형성하지 않도록 edge를 선택한다. 이때 union&find 알고리즘을 활용한다.

## 2. 구현

```
private boolean weightedUnion(int source, int destination) {
    int sourceRoot = collapsingFind(source);
    int destinationRoot = collapsingFind(destination);

    if(sourceRoot == destinationRoot) return false;
    parent[sourceRoot] = destinationRoot;
    return true;
}

private int collapsingFind(int vertex) {
    if(vertex == parent[vertex]) return vertex;
    return parent[vertex] = collapsingFind(parent[vertex]);
}
```

두 method를 사용하여, unionfind 알고리즘을 구현하였다.

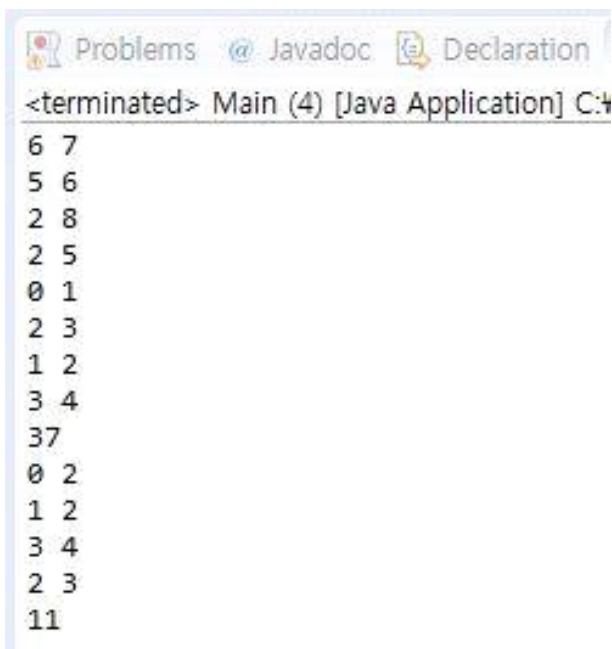
```
public int kruskal() {
    int result = 0;
    int count = 0;

    Collections.sort(edgeArray, Collections.reverseOrder());

    for(Edge edge : edgeArray) {
        if(weightedUnion(edge._source, edge._destination)) {
            System.out.println(edge._source + " " + edge._destination);
            result += edge._weight;
            if(++count == _vertex - 1) return result;
        }
    }
    return result;
}
```

그래프의 모든 edge를 반복문을 통하여 순회한다.

## 3. 결과



```
Problems @ Javadoc Declaration
<terminated> Main (4) [Java Application] C:\
6 7
5 6
2 8
2 5
0 1
2 3
1 2
3 4
3 7
0 2
1 2
3 4
2 3
11
```