-알고리즘 과제02-

과목명	알고리즘
교수명	최지웅
학 과	컴퓨터학부
학 번	20203123
이 름	이중곤
제출일	2023.11.22

목차

1.1 Algorithm 4.2 Kruskal's Algorithm 구현 (Java)

```
import java.util.ArrayList;
import java.util.Arrays;

public class Kruskal {

  // number of vertices
  private static int N = 5;
  // number of edges
  private static int M = 7;
  // 교재의 입력 데이터
  private static int[][] set_of_edges = {
        {1, 2, 1},
        {3, 5, 2},
        {1, 3, 3},
        {2, 3, 3},
        {3, 4, 4},
        {4, 5, 5},
        {2, 4, 6}
  };

  // 자작 데이터
  private static int[][] set_of_edges2 = {
        {1, 2, 3},
        {1, 3, 5},
        {2, 3, 2},
        {2, 4, 6},
        {2, 5, 4},
        {3, 4, 1},
        {4, 5, 7}
  };
  private static ArrayList<int[]> F = new ArrayList<>();

  public static void main(String[] args) {
    kruskal(N, M, set_of_edges2, F);

    for (int[] row : F) {
      for (int col = 0; col < row.length; col++) {
        if (col == 2) {
          System.out.printf("%d", row[col]);
          continue;
        }
        System.out.printf("v%d ", row[col]);
      }
      System.out.println();
    }
  }

  private static void kruskal(int n, int m, int[][] setOfEdges, ArrayList<int[]> F) {
    int i, j; // index
    int p, q; // set_pointer
    int[] e; // edge

    // sort the m edges in E by weight in nondecreasing order
```

```
    sort(m, setOfEdges);

    // initialize disjoint sets
    DisjointSet ds = new DisjointSet(n);

    while (F.size() < n - 1) {
      // edges with least weight not yet considered
      e = setOfEdges[0];
      // indicies of vertices connected by e
      i = e[0] - 1;
      j = e[1] - 1;

      p = ds.find(i);
      q = ds.find(j);

      if (!ds.equal(p, q)) {
        ds.merge(p, q);
        // add e to F
        F.add(e);
      }

      // remove e from E
      setOfEdges = Arrays.copyOfRange(setOfEdges, 1, setOfEdges.length);
    }
  }

  private static void sort(int m, int[][] setOfEdges) {
    int i, j;
    int[] e;

    for (i = 0; i < m - 1; i++) {
      for (j = i + 1; j < m; j++) {
        int weight1 = setOfEdges[i][2];
        int weight2 = setOfEdges[j][2];

        if (weight1 > weight2) {
          e = setOfEdges[i];
          setOfEdges[i] = setOfEdges[j];
          setOfEdges[j] = e;
        }
      }
    }
  }
}
```

1.2 Appendix C Disjoint Set Data Structure 구현 (Java)

```
public class DisjointSet {

  private final int SIZE;
  private Node[] universe;

  public DisjointSet(int n) {
    SIZE = n;
```

```
        universe = new Node[SIZE];
        initial(SIZE);
    }

    static class Node {

        int parent;
        int depth;

        Node(int parent, int depth) {
            this.parent = parent;
            this.depth = depth;
        }
    }

    void makeSet(int i) {
        universe[i] = new Node(i, 0);
    }

    int find(int i) {
        int j = i;

        while (universe[j].parent != j) {
            j = universe[j].parent;
        }

        return j;
    }

    void merge(int p, int q) {
        if (universe[p].depth == universe[q].depth) {
            universe[p].depth += 1;
            universe[q].parent = p;
        } else if (universe[p].depth < universe[q].depth) {
            universe[p].parent = q;
        } else {
            universe[q].parent = p;
        }
    }

    boolean equal(int p, int q) {
        if (p == q) {
            return true;
        } else {
            return false;
        }
    }

    void initial(int n) {
        for (int i = 0; i < n; i++) {
            makeSet(i);
        }
    }
}
```

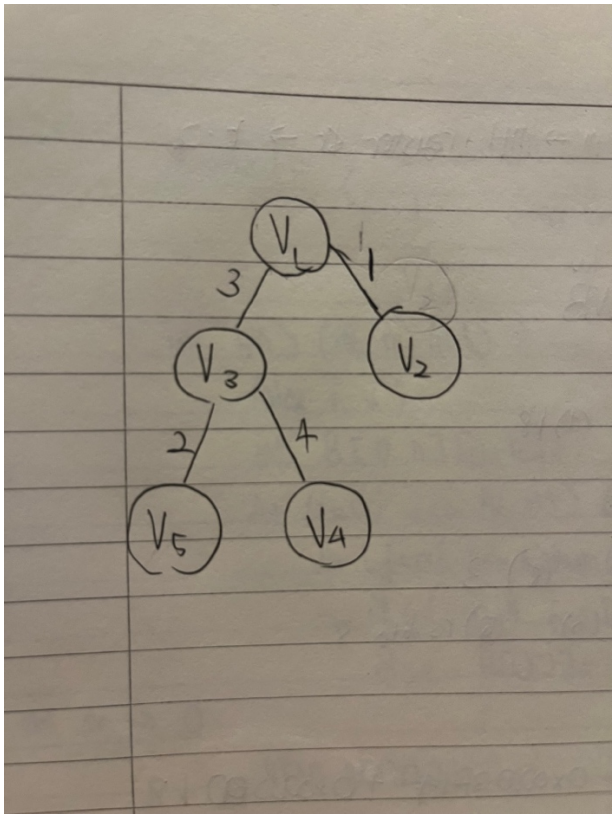
2. 교재 입력 데이터 테스트

v1 v2 1

v3 v5 2

v1 v3 3

v3 v4 4

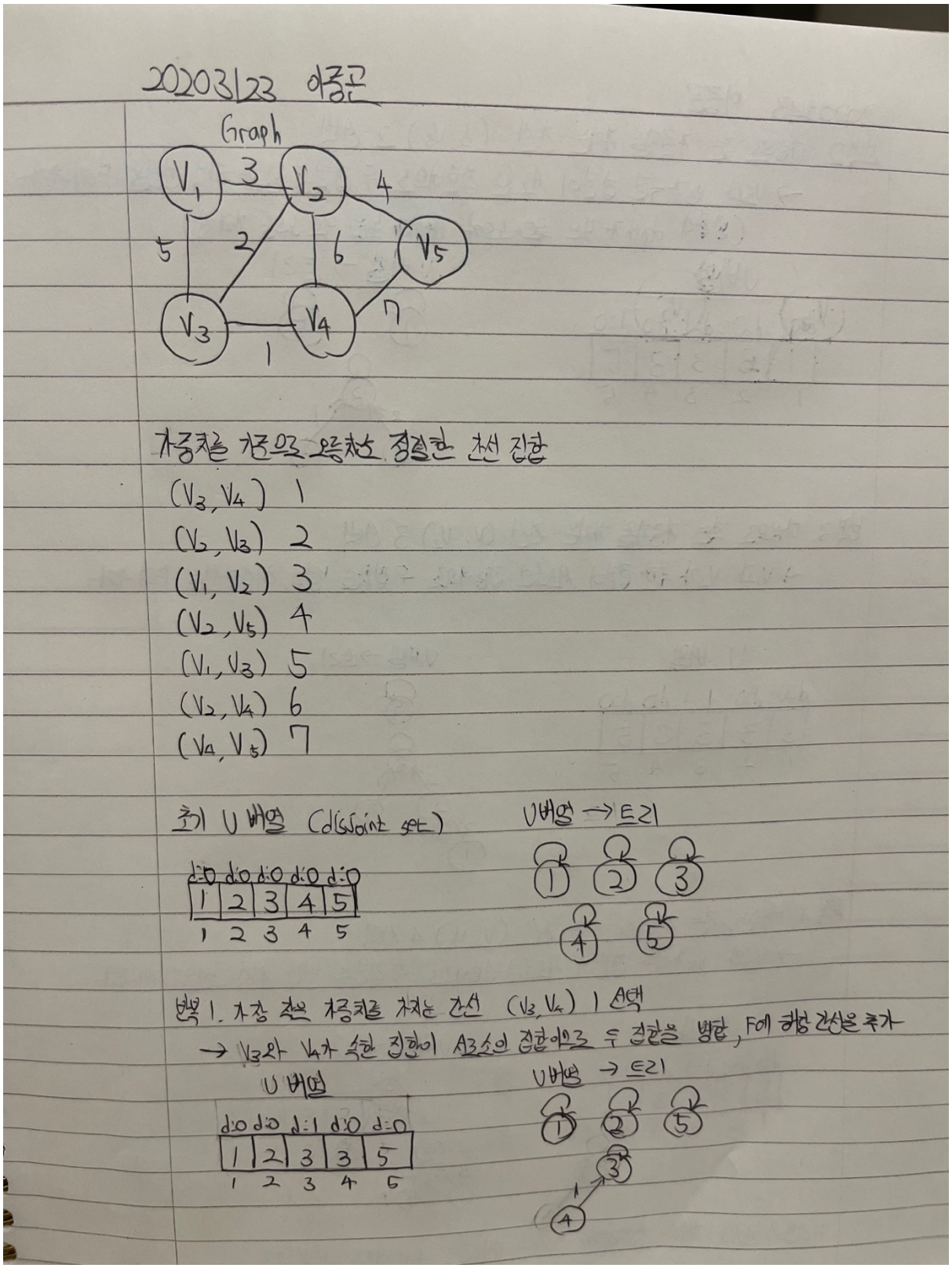


3. 자작 입력 데이터 생성 (Java)

```
private static int[][] set_of_edges2 = {
        {1, 2, 3},
        {1, 3, 5},
        {2, 3, 2},
        {2, 4, 6},
        {2, 5, 4},
        {3, 4, 1},
        {4, 5, 7}
```

};

4-1. 자작 데이터를 이용한 Kruskal 알고리즘 손계산



20200323 이중곤

반복 2. 다음으로 작은 가중치를 가지는 간선 (V_2, V_3) 2 선택

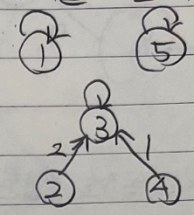
→ V_2와 V_3가 속한 집합이 서로소인 집합이므로 두 집합을 병합, 해당 간선을 T에 추가

(병합시 depth가 낮은 쪽 노드가 depth가 높은 쪽 노드를 처리)

U배열

d:0	d:0	d:1	d:0	d:0
1	3	3	3	5
1	2	3	4	5

U 배열 → 트리



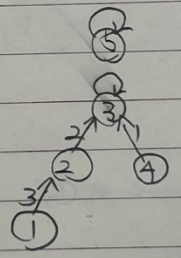
반복 3. 다음으로 작은 가중치를 가지는 간선 (V_1, V_2) 3 선택

→ V_1과 V_2가 속한 집합이 서로소인 집합이므로 두 집합을 병합, 해당간선을 T에 추가

U 배열

d:0	d:0	d:1	d:0	d:0
3	3	3	3	5
1	2	3	4	5

U 배열 → 트리



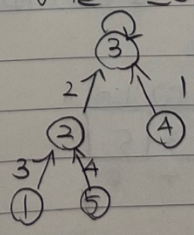
반복 4. 다음으로 작은 가중치를 가지는 간선 (V_2, V_5) 4 선택

→ V_2와 V_5가 속한 집합이 서로소인 집합이므로 두 집합을 병합, 해당 간선을 T에 추가

U 배열

d:0	d:0	d:1	d:0	d:0
3	3	3	3	3
1	2	3	4	5

U 배열 → 트리



T 집합의 간선의 개수가 정점의 개수 5보다 하나 적은 4개 이므로 반복 종료

5. 자작 데이터 테스트

```
v3 v4 1
v2 v3 2
v1 v2 3
v2 v5 4
```