-알고리즘 과제03-

과목명	알고리즘
교수명	최지웅
학 과	컴퓨터학부
학 번	20203123
이 름	이중곤
제출일	2023.11.25

목차

1. 알고리즘 구현
   - 1.1. Algorithm 4.3 Dijkstra's Algorithm 구현
2. 교재 입력 데이터 테스트
3. 자작 입력 데이터 생성
4. 자작 데이터를 이용한 알고리즘 손계산
5. 자작 데이터 테스트

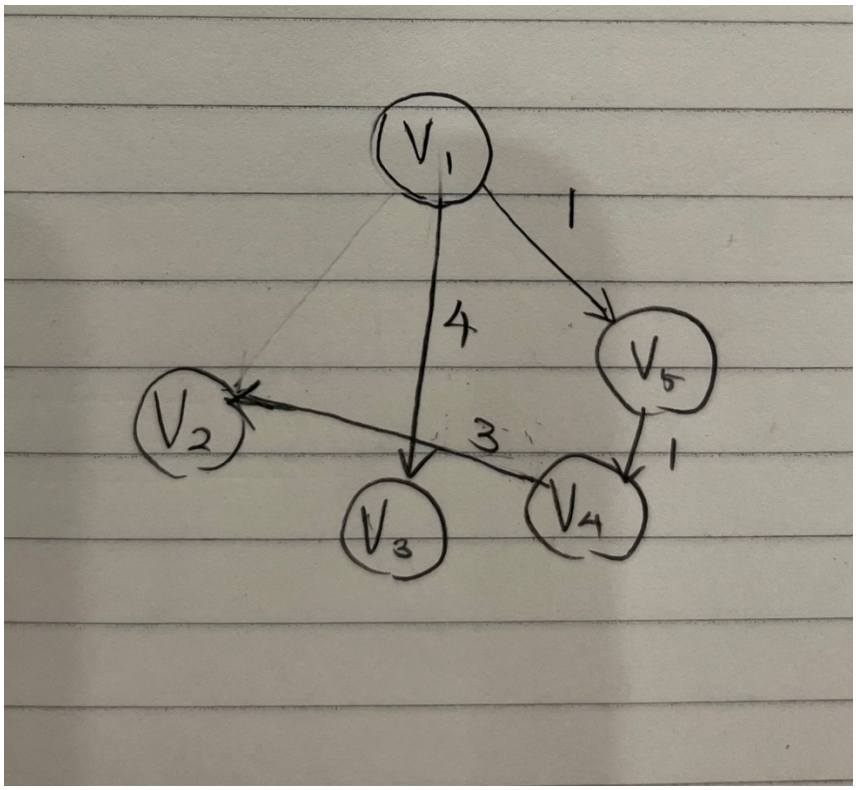
1.1 Algorithm 4.3 Dijkstra's Algorithm 구현 (Java)

```java
import java.util.ArrayList;
import java.util.List;

public class Dijkstra {

    // 교재 입력 데이터
    private static final int[][] W1 = {
            {0, 7, 4, 6, 1},
            {999, 0, 999, 999, 999},
            {999, 2, 0, 5, 999},
            {999, 3, 999, 0, 999},
            {999, 999, 999, 1, 0}
    };

    // 자작 데이터
    private static final int[][] W2 = {
            {0, 2, 8, 9, 999},
            {999, 0, 4, 4, 999},
            {999, 999, 0, 5, 2},
            {999, 999, 999, 0, 1},
            {999, 999, 999, 999, 0},
    };

    public static void main(String[] args) {
        List<int[]> F = new ArrayList<>();

        dijkstra(5, W2, 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] + 1);
            }
            System.out.println();
        }
    }

    private static void dijkstra(int n, int[][] W, List<int[]> F) {
        int[] edge;
        int vnear = 0;
        int[] touch = new int[n];
        int[] length = new int[n];

        // 모든 정점에 대해 v1을 v1 -> 정점의 최단 경로의 마지막 정점으로 초기화하고,
        // 해당 경로의 길이를 v1 -> 정점에 대한 가중치로 초기화
        for (int i = 1; i < n; i++) {
            touch[i] = 0;
            length[i] = W[0][i];
        }
```

```
    // n - 1개의 정점이 고려될 때까지 반복
    for (int i = 1; i < n; i++) {
      int min = 999;
      // 각 정점에 대해 최단 경로가 있는지 확인
      for (int j = 1; j < n; j++) {
        if (0 <= length[j] && length[j] < min) {
          min = length[j];
          vnear = j;
        }
      }

      // touch[vnear] 정점에서 vnear 정점으로 가는 간선 edge
      edge = new int[]{touch[vnear], vnear, W[touch[vnear]][vnear]};
      // edge를 해집합 F에 추가
      F.add(edge);

      // vnear 정점을 거쳐 가는 경로가 더 짧으면 최단 경로를 갱신
      for (int k = 1; k < n; k++) {
        if (length[vnear] + W[vnear][k] < length[k]) {
          length[k] = length[vnear] + W[vnear][k];
          touch[k] = vnear;
        }
      }

      // vnear 정점을 다시 고려하지 않도록 length[vnear]를 -1로 설정
      length[vnear] = -1;
    }
  }
}
```

2. 교재 입력 데이터 테스트

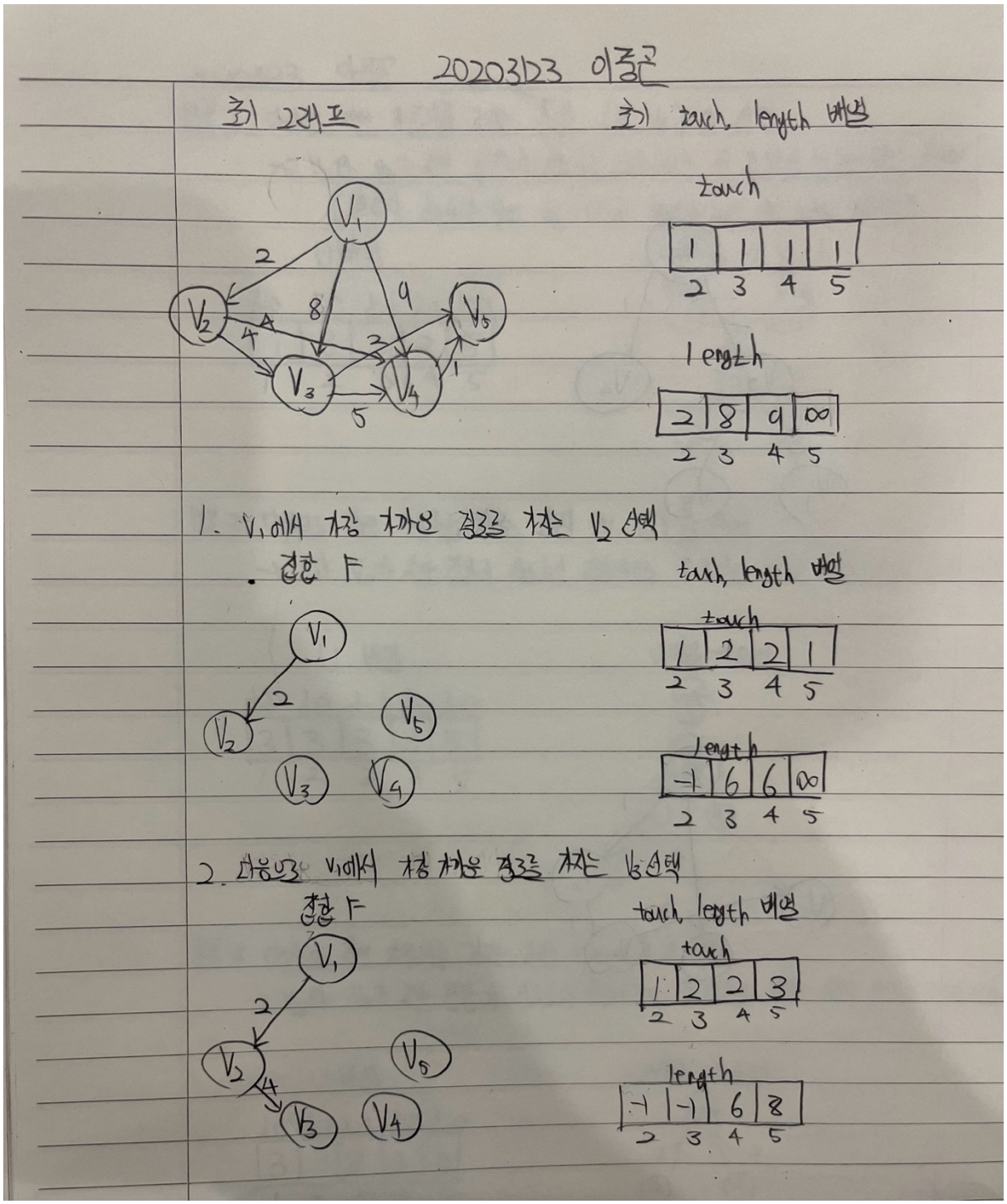
```
v1 v5 1
v5 v4 1
v1 v3 4
v4 v2 3
```



3. 자작 입력 데이터 생성 (Java)

```
private static final int[][] W2 = {
        {0, 2, 8, 9, 999},
        {999, 0, 4, 4, 999},
        {999, 999, 0, 5, 2},
        {999, 999, 999, 0, 1},
        {999, 999, 999, 999, 0},
};
```

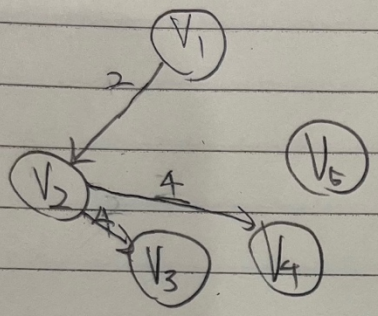
4. 자작 데이터를 이용한 Dijkstra's Alogrithm 손계산



2020/3/23 이중곤

3. 다음은 V_1에서 최소 거리로 경로를 찾는 V_4 선택

집합 F



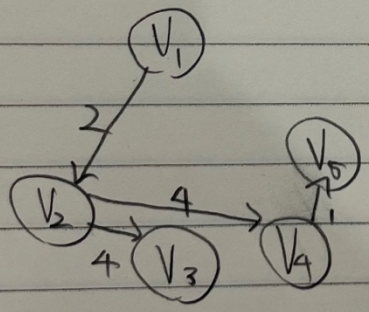
touch, length 배열

touch			
1	2	2	4
2	3	4	5

length			
-1	-1	-1	7
2	3	4	5

4. 다음은 V_1에서 최소 거리로 경로를 찾는 V_5 선택

집합 F



touch, length 배열

touch			
1	2	2	4
2	3	4	5

length			
-1	-1	-1	-1
2	3	4	5

5. V_1을 제외한 총 4개에 대한 최단 경로를 모두 찾았으므로 반복종료

5. 자작 데이터 테스트

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