[프로젝트] 그래프 정제 및 삼각형 수 세기

학번 : 20181612

이름 : 박병규

전체적인 구현 과정

Task1 -> Task 2 -> Triangle listing on MapReduce

```
1  package project2;
2  //
3  import org.apache.hadoop.conf.Configured;
17
18
19 public class TriCnt extends Configured implements Tool {
20
21     public static void main(String[] args) throws Exception {
22         ToolRunner.run(new TriCnt(), args);
23     }
24
25     public int run(String[] args) throws Exception {
26
27         String inputpath = args[0];
28         String tmppath = inputpath + ".tmp";
29         String tmppath2 = inputpath + "2.tmp";
30         String tmppath3 = inputpath + "3.tmp";
31         String tmppath4 = inputpath + "4.tmp";
32         String outpath = inputpath + ".out";
33         //task1
34         runStep0(inputpath, tmppath);
35
36         //task2
37         runStep1(tmppath, tmppath2);
38         runStep1_2(tmppath2, tmppath3);
39
40         //triangle
41         runStep2_1(tmppath3, tmppath4);
42         runStep2_2(tmppath3, tmppath4, outpath);
43
44         return 0;
45     }
```

과제를 진행함에 있어서 처음에는 해결이 잘 안되는 부분은 Text 타입을 섞어 과제를 진행하였으나 wiki-topcats의 데이터 셋을 적용하면 무척 시간이 걸리고 용량을 많이 잡아먹는 현상이 생겨 이를 예방하기 위해 Text타입은 사용하지 않았고 Input/Ouput Format을 SequenceFile를 사용하며 진행을 했다.

Task 1. 구현 방법 및 결과 설명

Mapper

```java
package project2;

import java.io.IOException;


public class TriStep0Mapper extends Mapper<Object, Text, IntPairWritable, IntWritable> {

    IntPairWritable ok = new IntPairWritable();
    IntWritable ov = new IntWritable();

    @Override
    protected void map(Object key, Text value, Mapper<Object, Text, IntPairWritable, IntWritable>.Context context)
            throws IOException, InterruptedException {

        StringTokenizer st = new StringTokenizer(value.toString());

        int u = Integer.parseInt(st.nextToken());
        int v = Integer.parseInt(st.nextToken());

        if (u < v) {
            ok.set(u, v);
            ov.set(-1);
            context.write(ok, ov);
        }
        else if (u > v) {
            ok.set(v, u);
            ov.set(-1);
            context.write(ok, ov);
        }
    }
}
```

Reducer

```java
package project2;

import java.io.IOException;

public class TriStep0Reducer extends Reducer<IntPairWritable, IntWritable, IntWritable, IntWritable>{

    IntWritable ok = new IntWritable();
    IntWritable ov = new IntWritable();
    String str_key;
    String[] arr;
    @Override
    protected void reduce(IntPairWritable key, Iterable<IntWritable> values,
            Reducer<IntPairWritable, IntWritable, IntWritable, IntWritable>.Context context) throws IOException, InterruptedException {

        ok.set(key.u);
        ov.set(key.v);
        context.write(ok, ov);
    }
}

```

u 와 v 사이의 크기관계를 이용하여 순서를 정해줌으로써 (u,v), (v,u)를 모두 포함하는 중복되는 경우를 제거해 주었다.

```
PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt.tmp> Get-ChildItem .₩* -Include part-r-00000  -Recurse | Get-Content | Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
 3246


PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt.tmp> Get-ChildItem .₩* -Include part-r-00001_ -Recurse | Get-Content | Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
 3270


PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt.tmp> Get-ChildItem .₩* -Include part-r-00002_ -Recurse | Get-Content | Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
 3186
```

```
PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources> Get-ChildItem .₩* -Include testset.txt  -Recurse | Get-Content | Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
10000
```

Task 1을 진행한 결과

9,702(3246+3270+3186) < 10000임을 알 수 있다.

여기서 테스트를 진행했던 testset.txt.는 wiki-tocats의 상위 10000줄을 가지고 온 것이다.

Task1 실행결과 중복이 제거되어 edge 수가 준 것을 확인 할 수 있다.

Task 2 구현 방법 및 결과 설명

1 단계

Mapper

```java
package project2;

import java.io.IOException;

public class Task2step1Mapper extends Mapper<IntWritable, IntWritable, IntWritable, IntPairWritable> {

    IntWritable ok = new IntWritable();
    IntPairWritable ov = new IntPairWritable();

    @Override
    protected void map(IntWritable key, IntWritable value, Mapper<IntWritable, IntWritable, IntWritable, IntPairWritable>.Context context)
            throws IOException, InterruptedException {

        //StringTokenizer st = new StringTokenizer(value.toString());

        int u = key.get();
        int v = value.get();
        ok.set(u);
        ov.set(u,v);
        context.write(ok, ov);
        ok.set(v);
        ov.set(u,v);
        context.write(ok, ov);
    }
}
```

Reducer

```java
package project2;

import java.io.IOException;

public class Task2step1Reducer extends Reducer<IntWritable, IntPairWritable, IntPairWritable, IntPairWritable>{
    IntPairWritable ok = new IntPairWritable();
    IntPairWritable ov = new IntPairWritable();
    String str_v;
    @Override
    protected void reduce(IntWritable key, Iterable<IntPairWritable> values,
            Reducer<IntWritable, IntPairWritable, IntPairWritable, IntPairWritable>.Context context) throws IOException, InterruptedException {

        ArrayList<Integer[]> list = new ArrayList<Integer[]>();
        int k = key.get();
        int leng = 0;
        for(IntPairWritable v : values) {
            int a = v.u;
            int b = v.v;
            list.add(new Integer[] {a,b});
            leng++;
        }

        for(Integer[] val : list) {
            if (k == val[0]) {
                ok.set(val[0], val[1]);
                ov.set(leng, -1);
                context.write(ok, ov);
            }
            else  {
                ok.set(val[0], val[1]);
                ov.set(-1, leng);
                context.write(ok, ov);
            }
        }
    }

}
```

Mapper에서 각각의 u, v 값을 key로 설정하고 u,v를 묶은 IntPairWritable값을 value로 해서 reducer로 전달해 주었다. Reducer에서는 각각의 키 값이 갖는 degree값을 갖도록 하였다. 하

직 엣지에 포함되는 두개의 값에 대해서는 degree값을 알지 못하므로 Mapreduce를 한번 더 진행하였다.

2 단계

Mapper

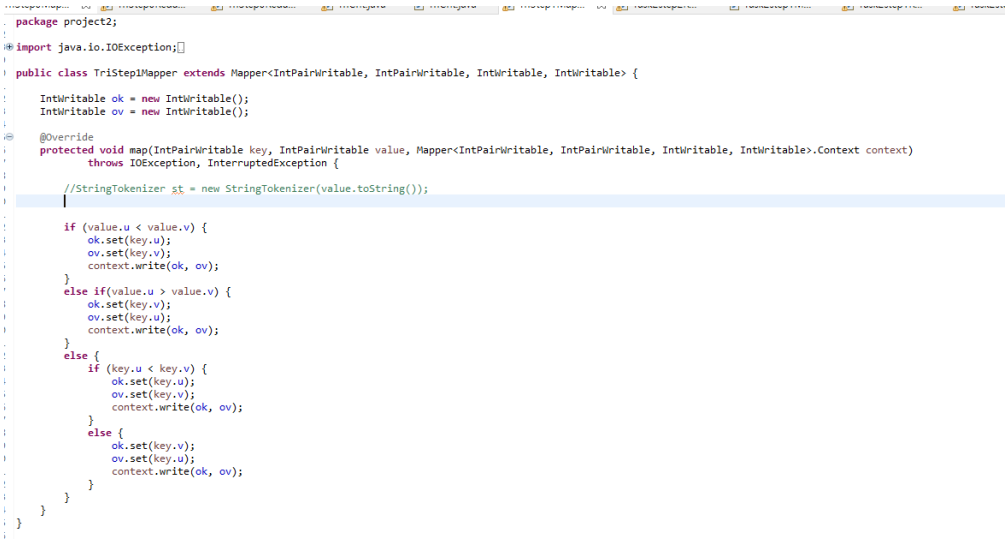
```
package project2;

import java.io.IOException;

public class Task2step2Mapper extends Mapper<IntPairWritable, IntPairWritable,IntPairWritable, IntPairWritable> {

    IntPairWritable ok = new IntPairWritable();
    IntPairWritable ov = new IntPairWritable();

    @Override
    protected void map(IntPairWritable key, IntPairWritable value, Mapper<IntPairWritable, IntPairWritable, IntPairWritable, IntPairWritable>.Context context)
            throws IOException, InterruptedException {

        // st = 0 67    0 67 347 x
        int k1 = key.u;
        int k2 = key.v;
        int v1 = value.u;
        int v2 = value.v;

        ok.set(k1, k2);
        ov.set(v1, v2);

        context.write(ok, ov);

    }
}
```

Reducer

```
package project2;

import java.io.IOException;

public class Task2step2Reducer extends Reducer<IntPairWritable, IntPairWritable, IntPairWritable, IntPairWritable>{
    IntPairWritable ok = new IntPairWritable();
    IntPairWritable ov = new IntPairWritable();


    @Override
    protected void reduce(IntPairWritable key, Iterable<IntPairWritable> values,
            Reducer<IntPairWritable, IntPairWritable, IntPairWritable, IntPairWritable>.Context context) throws IOException, InterruptedException {
        //0 102 x 60 102 347 x

//      String kk = key.toString();

        int cnt1 = 0;
        int cnt2 = 0;
        for (IntPairWritable a : values) {
            if (a.u == -1) {
                cnt2 = a.v;
            }
            else {
                cnt1 = a.u;
            }
        }
        ov.set(cnt1, cnt2);
        context.write(key, ov);

    }
}
```

2단계의 Reducer 에서는 1 단계에서 보내준 값을 토대로 degree 정보가 있으면 나머지 degree 값을 IntPair로 묶고 value값으로 전달해 주었고 각각의 u,v값도 IntPair로 묶어 key값으로 전달해 주었다. degree값 비교를 통해 자리를 옮겨 전달을 하려 했지만 값이 누락되는 경우가 생겨 다음 Mapper에서 degree 비교를 통해 데이터를 처리해주기로 하였다.

Mapper



```java
package project2;

import java.io.IOException;

public class TriStep1Mapper extends Mapper<IntPairWritable, IntPairWritable, IntWritable, IntWritable> {

    IntWritable ok = new IntWritable();
    IntWritable ov = new IntWritable();

    @Override
    protected void map(IntPairWritable key, IntPairWritable value, Mapper<IntPairWritable, IntPairWritable, IntWritable, IntWritable>.Context context)
            throws IOException, InterruptedException {

        //StringTokenizer st = new StringTokenizer(value.toString());

        if (value.u < value.v) {
            ok.set(key.u);
            ov.set(key.v);
            context.write(ok, ov);
        }
        else if(value.u > value.v) {
            ok.set(key.v);
            ov.set(key.u);
            context.write(ok, ov);
        }
        else {
            if (key.u < key.v) {
                ok.set(key.u);
                ov.set(key.v);
                context.write(ok, ov);
            }
            else {
                ok.set(key.v);
                ov.set(key.u);
                context.write(ok, ov);
            }
        }
    }
}
```

이 과정에서 가져온 u, v 값과 이에 해당하는 degree를 통해 degree(u) < degree(v) 이거나, degree(u) == degree(v) 이면서 id(u) < id(v) 인 경우는 그대로 두고 그 밖의 경우에는 u와 v의 순서를 바꿔주었다.

Task 3. 삼각형 MapReduce 알고리즘에 1) Task 1의 결과 그래프와 2) Task 2의 결과 그래프를 각각 입력했을 때, 성능 비교하기

- 중간중간 발생하는 출력 값 비교하기(wiki-topcats 상단 10000줄로 진행)

```
1	2
1	598775
3	1100919
3	1101828
3	1101924
3	1101945
5	1101550
5	1101709
5	1102159
7	1102709
8	1101610
8	1103797
10	1101491
11	285865
12	279696
13	279122
14	279607
15	279696
17	279607
18	279696
18	1214910
19	279122
20	279607
21	270087
21	279696
22	279122
22	280241
24	279714
26	46
```

중복값을 제거하는 taks1을 실행하고 나온 출력값이다. 보고서 제일 상단부분에서 설명 했듯이 라인 수를 카운팅 했을 때 중복값이 제거되어 라인 수가 줄어드는 것을 확인 할 수 있다.

Task2 의 1단계를 실행하고 출력되는 모습이다.

```
0	10772	1	-1
3	1101827	9	-1
3	1101709	9	-1
3	1101724	9	-1
3	1101808	9	-1
3	1102561	9	-1
3	1100919	9	-1
3	1101945	9	-1
3	1101924	9	-1
3	1101828	9	-1
6	1102234	1	-1
```

1단계이므로 한쪽의 키 값에 대해서는 degree값이 채워져 있고 남은 한쪽은 -1로 채워준 모습을 볼 수 있다.

Task2 의 2단계를 실행하고 출력되는 모습이다.

```
5	1101709	6	7
5	1102159	6	2
7	1102709	1	3
8	1101610	7	1
8	1103797	7	1
10	1101491	4	1
11	285865	1	1
12	279696	3	9
13	279122	4	143
```

두개의 u, v 값에 대해 각각의 degree값이 채워져 있고 이제 다음 Mapper에서 degree가 낮은 쪽을 왼쪽에 둬서 연산을 하게되면 보다 더 효율적이게 삼각형을 추출할 수 있을 것이다.

Wedge를 찾아서 결합한 모습을 볼 수 있다.

279122	279696	13
279122	279607	13
270087	279122	13
270087	279696	13
270087	279607	13
279607	279696	13
46	279696	16
270087	279122	19
270087	279607	19
279122	279607	19

최종 찾은 삼각형

104	107	109
104	107	1174302
104	107	1178487
105	106	1178487
105	109	1178487
108	109	1178487
110	113	1185127
110	113	1184263
110	279122	113
126	127	273720
126	127	272352
126	127	1061389

Wiki-topcats.txt 데이터 셋으로 삼각형을 찾고 개수를 세어본 결과

52106893이라는 값이 나왔고 이는 stanford edu 사이트와 값이 일치함을 확인 할 수 있다.

```
pbk5485@kmubigdata-cluster-m:~$ hdfs dfs -cat /wiki-topcats.txt.out* | wc -l
cat: `/wiki-topcats.txt.out': Is a directory
0
pbk5485@kmubigdata-cluster-m:~$ hdfs dfs -cat /wiki-topcats.txt.out/* | wc -l
52106893
pbk5485@kmubigdata-cluster-m:~$
```

Dataset statistics	
Nodes	1791489
Edges	28511807
Nodes in largest WCC	1791489 (1.000)
Edges in largest WCC	28511807 (1.000)
Nodes in largest SCC	1791489 (1.000)
Edges in largest SCC	28511807 (1.000)
Average clustering coefficient	0.2746
Number of triangles	52106893
Fraction of closed triangles	0.00165
Diameter (longest shortest path)	9
90-percentile effective diameter	3.8

- 두 경우에 대해서, 실행시간 및 성능 비교

Task1 wedges 수

```
PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt2.tmp> Get-ChildItem .₩* -Include part-r-00000  -Recurse | Get-Content |
Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
59725


PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt2.tmp> Get-ChildItem .₩* -Include part-r-00001  -Recurse | Get-Content |
Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
56586


PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt2.tmp> Get-ChildItem .₩* -Include part-r-00002  -Recurse | Get-Content |
Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
64359
```

Task2 wedges 수

```
PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt4.tmp> Get-ChildItem .₩* -Include part-r-00000  -Recurse | Get-Content |
Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
 2316


PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt4.tmp> Get-ChildItem .₩* -Include part-r-00001  -Recurse | Get-Content |
Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
 2412


PS C:₩Users₩pbk54₩Desktop₩3학년1학기₩빅최기₩triangle_mr_orig₩src₩test₩resources₩testset.txt4.tmp> Get-ChildItem .₩* -Include part-r-00002  -Recurse | Get-Content |
Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
 2408
```

Task1에서 wedge가 더 많이 나타남을 눈으로 확인 할 수 있다.

먼저 Task 1의 결과 그래프에 wiki-topcats 데이터셋을 넣고 코드를 돌려보게 되면 내 노트북과 나의 인내심이 버티지 못하였다. 코드를 돌려놓고 잠에 들어도 봤지만 좋은 결과를 얻을 수는 없었다. 꽤나 많은 시간이 필요했고 컴퓨터 용량과 인내심도 많이 필요했다.

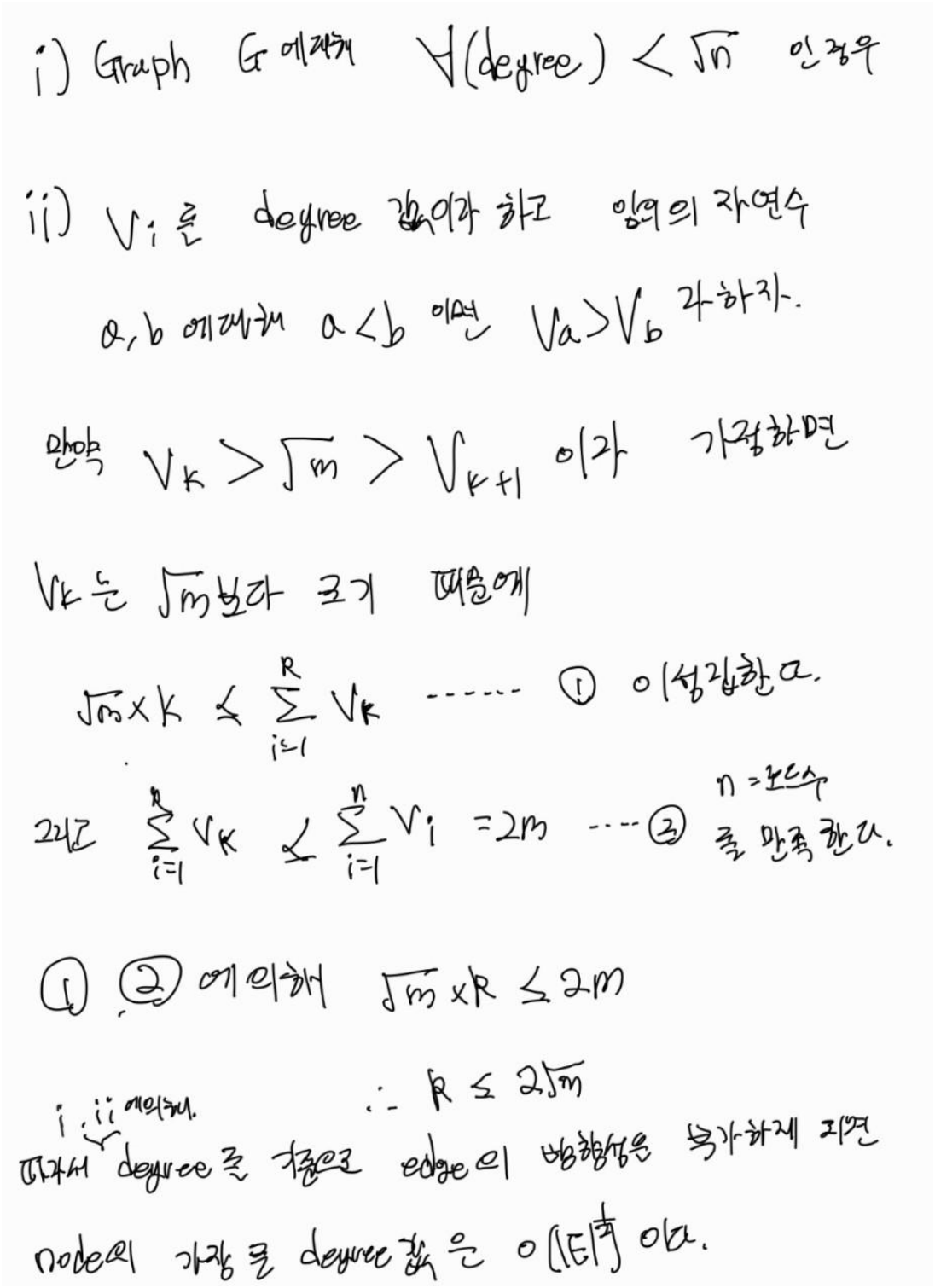
하지만 Task 2의 결과를 보게 되면 대략 24분정도 시간이 걸린 것을 확인할 수 있다.

Show 20 entries Search:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Resource Manager Host	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed	Elapsed Time
2022.06.20 07:48:27 UTC	2022.06.20 07:48:40 UTC	2022.06.20 08:03:52 UTC	job_1655710541920_0005	final-0.1.jar	pbk5485	kmubigdata-cluster-m	default	SUCCEEDED	55	55	5	5	00hrs, 15mins, 12sec
2022.06.20 07:45:46 UTC	2022.06.20 07:45:59 UTC	2022.06.20 07:48:23 UTC	job_1655710541920_0004	final-0.1.jar	pbk5485	kmubigdata-cluster-m	default	SUCCEEDED	5	5	5	5	00hrs, 02mins, 24sec
2022.06.20 07:42:44 UTC	2022.06.20 07:42:57 UTC	2022.06.20 07:45:42 UTC	job_1655710541920_0003	final-0.1.jar	pbk5485	kmubigdata-cluster-m	default	SUCCEEDED	10	10	5	5	00hrs, 02mins, 45sec
2022.06.20 07:40:35 UTC	2022.06.20 07:40:48 UTC	2022.06.20 07:42:41 UTC	job_1655710541920_0002	final-0.1.jar	pbk5485	kmubigdata-cluster-m	default	SUCCEEDED	5	5	5	5	00hrs, 01mins, 53sec
2022.06.20 07:38:14 UTC	2022.06.20 07:38:26 UTC	2022.06.20 07:40:31 UTC	job_1655710541920_0001	final-0.1.jar	pbk5485	kmubigdata-cluster-m	default	SUCCEEDED	4	4	5	5	00hrs, 02mins, 04sec

Task 1이 Task 2 결과가 출력 되지 않을 만큼 오래 걸리는 이유는 직관적으로 용량과 연산 횟수 때문임을 직관적으로 알 수 있다.

왜 Task 2 를 해야지 연산이 줄어드는 지에 대해서는 아래 수식을 통해 증명을 해봤다.



즉 task2를 거치고 나서는 최대 wedge의 수가 2*m**1/2이 이고 2개의 점을 뽑게 되면 대략 4m정도의 크기가 나온다 하지만 task2를 거치지 않게 된다면 최대 degree수가 m이 되고 m**2의 경우의 수 만큼 연산을 진행해야 하는 사항도 나올 수 있다. 따라서 최악이 경우 task2가 task1 보다 m/4배 더 빠르다고 볼 수 있다. Task2를 실행하고도 24분이라는 시간이 걸렸으니 Task1만으로 실행시켰을 때는 에러가 날 확률이 높고 노트북 성능이 못 받쳐준 것 같다.