

## CS543 - Homework 4

Group 1: Kanya Kreprasertkul  
NetID: kk1003

Siva Harshini Dev Bonam  
sdb202

Yifan Liao  
yl1463

### Q1. Edge and weight list

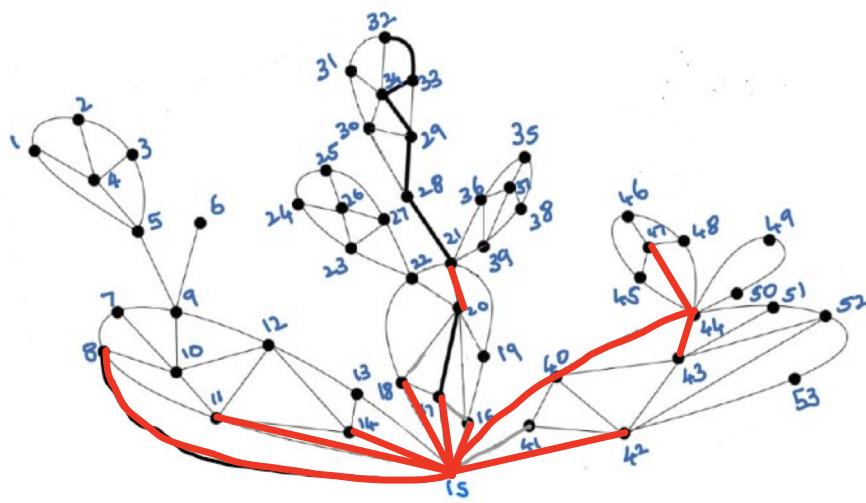
```
print(rdd_edge_weight_list.collect())  
[(1, 2, 0.1111111111111111), (1, 4, 0.0833333333333333), (1, 5, 0.0833333333333333), (2, 3, 0.1111111111111111), (2, 4, 0.0833333333333333),  
(3, 4, 0.0833333333333333), (3, 5, 0.0833333333333333), (4, 5, 0.0625), (5, 9, 0.05), (6, 9, 0.2), (7, 8, 0.0833333333333333), (7, 9,  
0.06666666666666667), (7, 10, 0.06666666666666667), (8, 10, 0.05), (8, 11, 0.05), (8, 15, 0.025), (9, 10, 0.04), (9, 12, 0.04), (10, 11, 0.  
04), (10, 12, 0.04), (11, 12, 0.04), (11, 14, 0.05), (11, 15, 0.02), (12, 13, 0.0666666666666667), (12, 14, 0.05), (13, 14, 0.0833333333333333)  
(13, 15, 0.0333333333333333), (14, 15, 0.025), (15, 16, 0.025), (15, 17, 0.025), (15, 18, 0.025), (15, 40, 0.02), (15, 41, 0.03333333  
(15, 42, 0.0166666666666666), (16, 17, 0.0625), (16, 19, 0.0833333333333333), (16, 20, 0.04166666666666664), (17, 18, 0.062  
5), (17, 20, 0.04166666666666664), (18, 20, 0.04166666666666664), (18, 22, 0.05), (19, 20, 0.0555555555555555), (19, 21, 0.0555555555555555  
555), (20, 21, 0.0277777777777776), (20, 22, 0.0333333333333333), (21, 22, 0.0333333333333333), (21, 28, 0.0555555555555555), (21, 36,  
0.04166666666666664), (21, 39, 0.04166666666666664), (22, 23, 0.05), (22, 27, 0.05), (23, 24, 0.0833333333333333), (23, 26, 0.0625), (23,  
27, 0.0625), (24, 25, 0.1111111111111111), (24, 26, 0.0833333333333333), (25, 26, 0.0833333333333333), (25, 27, 0.0833333333333333), (26,  
27, 0.0625), (28, 29, 0.0833333333333333), (28, 30, 0.0833333333333333), (29, 30, 0.0625), (29, 33, 0.0833333333333333), (29, 34, 0.05),  
(30, 31, 0.0833333333333333), (30, 34, 0.05), (31, 32, 0.1111111111111111), (31, 34, 0.0666666666666667), (32, 33, 0.1111111111111111), (3  
2, 34, 0.0666666666666667), (33, 34, 0.0666666666666667), (35, 36, 0.0833333333333333), (35, 37, 0.0833333333333333), (35, 38, 0.1111111  
1111111111), (36, 37, 0.0625), (36, 39, 0.0625), (37, 38, 0.0833333333333333), (37, 39, 0.0625), (38, 39, 0.0833333333333333), (40, 41, 0.  
0666666666666667), (40, 42, 0.0333333333333333), (40, 43, 0.04), (40, 44, 0.025), (41, 42, 0.0555555555555555), (42, 43, 0.0333333333333333  
33), (42, 52, 0.04166666666666664), (42, 53, 0.0833333333333333), (43, 44, 0.025), (43, 51, 0.0666666666666667), (43, 52, 0.05), (44, 45,  
0.04166666666666664), (44, 47, 0.03125), (44, 48, 0.04166666666666664), (44, 49, 0.0625), (44, 50, 0.0625), (44, 51, 0.0416666666666666  
4), (45, 46, 0.1111111111111111), (45, 47, 0.0833333333333333), (46, 47, 0.0833333333333333), (46, 48, 0.1111111111111111), (47, 48, 0.083  
33333333333333), (49, 50, 0.25), (51, 52, 0.0833333333333333), (52, 53, 0.125)]
```

### Q3. The corresponding intermediate solution

1) 1st step

```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[:12]

[(15, 42, 0.01666666666666666),
 (11, 15, 0.02),
 (15, 40, 0.02),
 (8, 15, 0.025),
 (14, 15, 0.025),
 (15, 16, 0.025),
 (15, 17, 0.025),
 (15, 18, 0.025),
 (40, 44, 0.025),
 (43, 44, 0.025),
 (20, 21, 0.02777777777777776),
 (44, 47, 0.03125)]
```



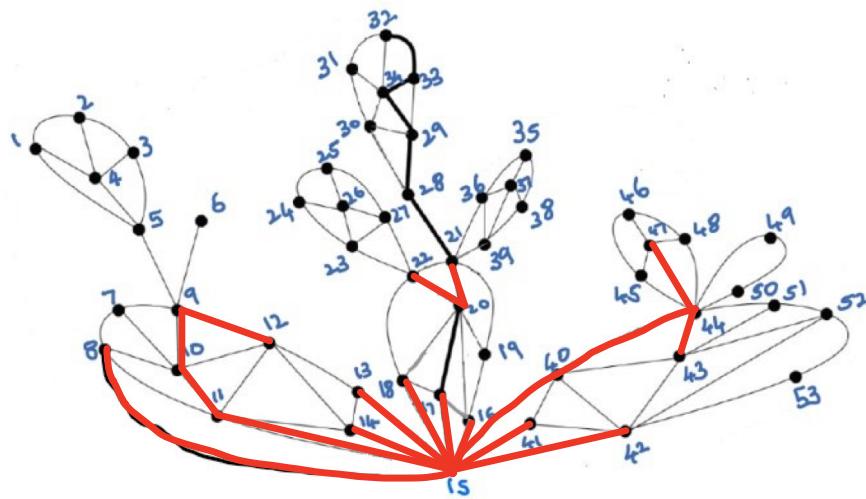
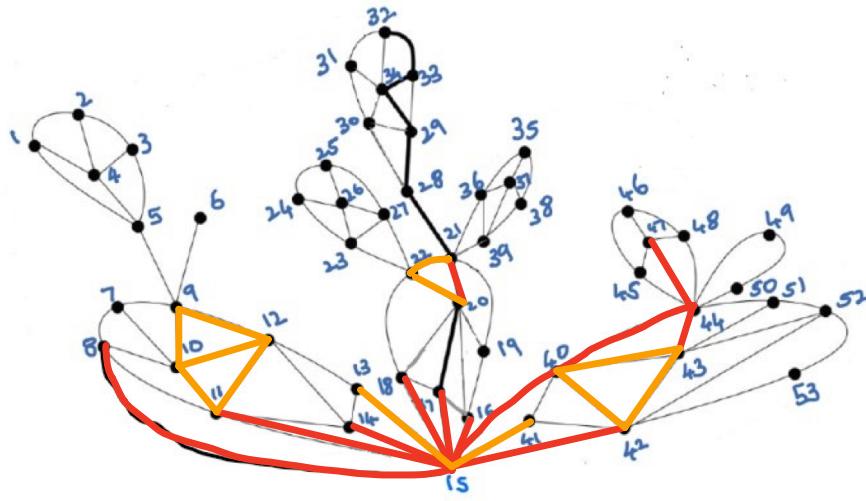
Note:

- 1) The red edges are the final edges from each step.
- 2) The yellow edges are the new 12 edges.

## 2) 2nd step

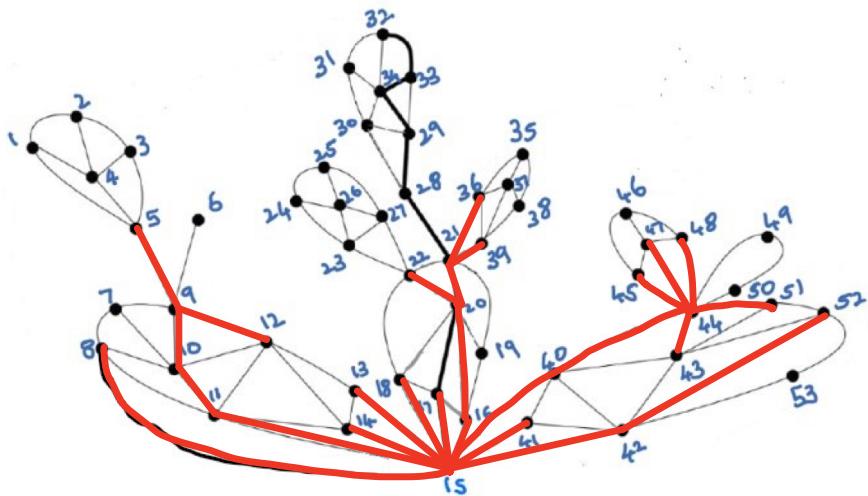
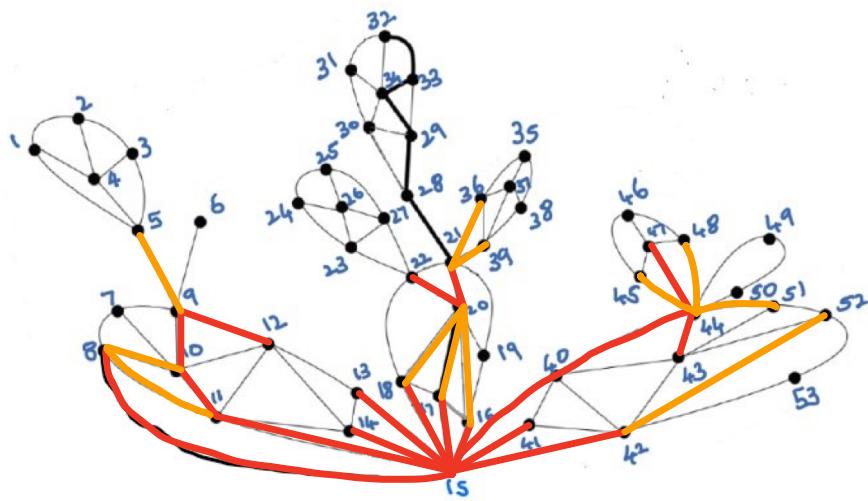
```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[12:24]

[(13, 15, 0.0333333333333333),
(15, 41, 0.0333333333333333),
(20, 22, 0.0333333333333333),
(21, 22, 0.0333333333333333),
(40, 42, 0.0333333333333333),
(42, 43, 0.0333333333333333),
(9, 10, 0.04),
(9, 12, 0.04),
(10, 11, 0.04),
(10, 12, 0.04),
(11, 12, 0.04),
(40, 43, 0.04)]
```



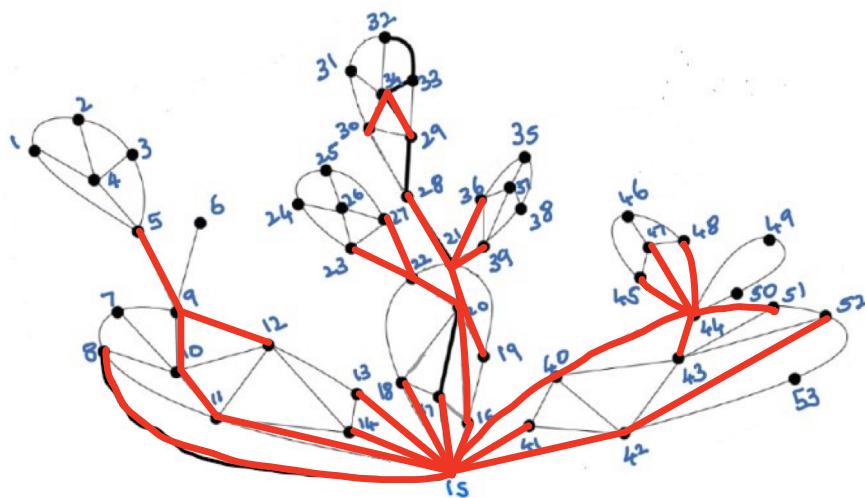
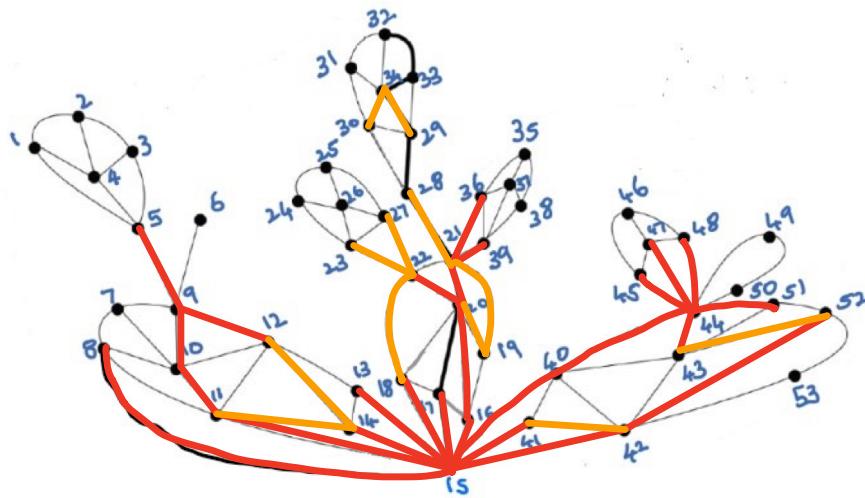
### 3) 3rd step

```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[24:36]
[(16, 20, 0.04166666666666664),
(17, 20, 0.04166666666666664),
(18, 20, 0.04166666666666664),
(21, 36, 0.04166666666666664),
(21, 39, 0.04166666666666664),
(42, 52, 0.04166666666666664),
(44, 45, 0.04166666666666664),
(44, 48, 0.04166666666666664),
(44, 51, 0.04166666666666664),
(5, 9, 0.05),
(8, 10, 0.05),
(8, 11, 0.05)]
```



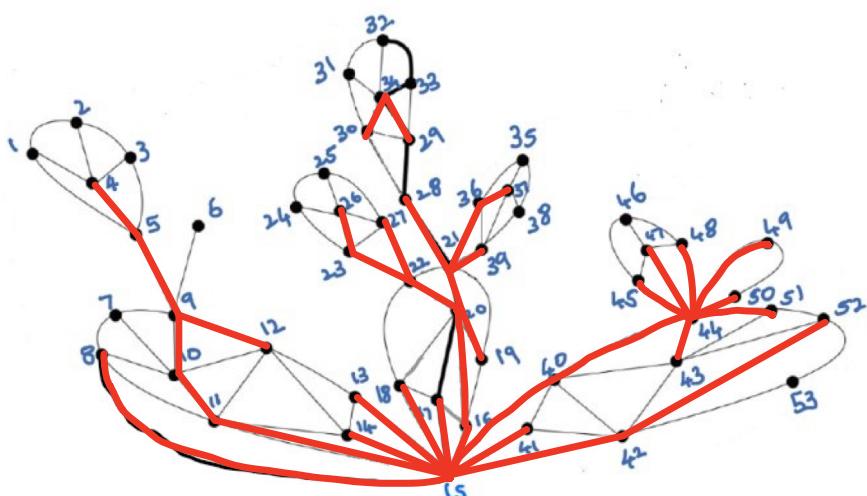
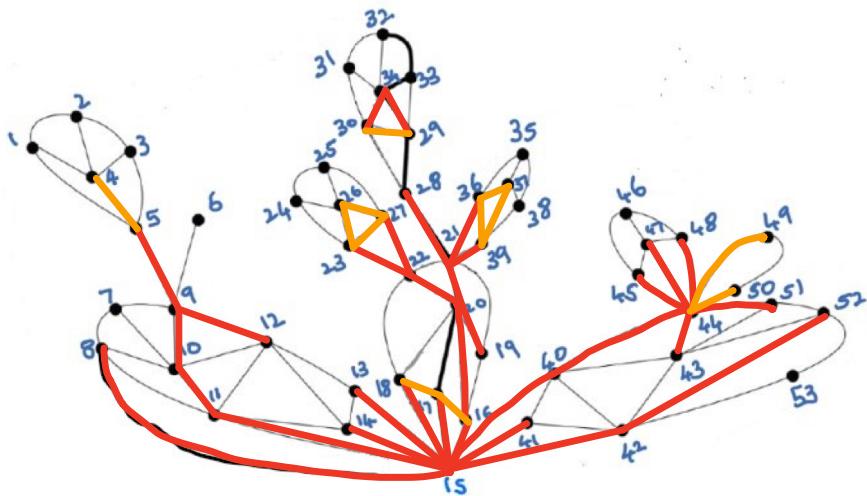
#### 4) 4th step

```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[36:48]  
[(11, 14, 0.05),  
(12, 14, 0.05),  
(18, 22, 0.05),  
(22, 23, 0.05),  
(22, 27, 0.05),  
(29, 34, 0.05),  
(30, 34, 0.05),  
(43, 52, 0.05),  
(19, 20, 0.05555555555555555),  
(19, 21, 0.05555555555555555),  
(21, 28, 0.05555555555555555),  
(41, 42, 0.05555555555555555)]
```



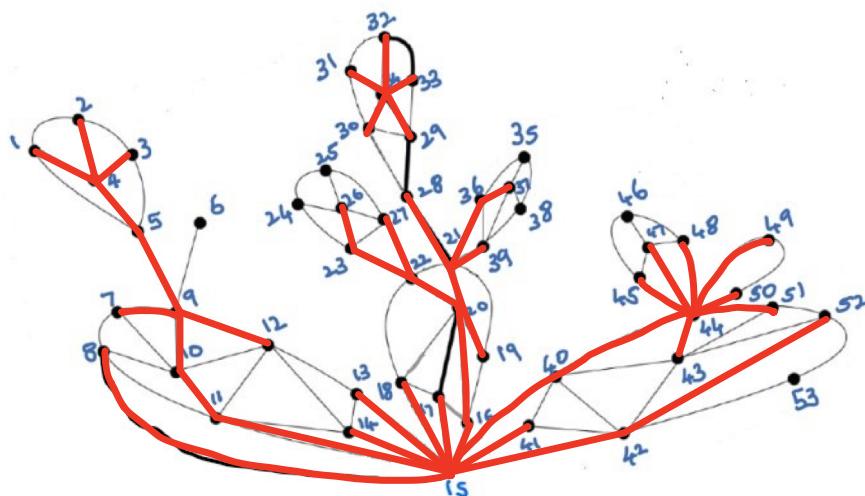
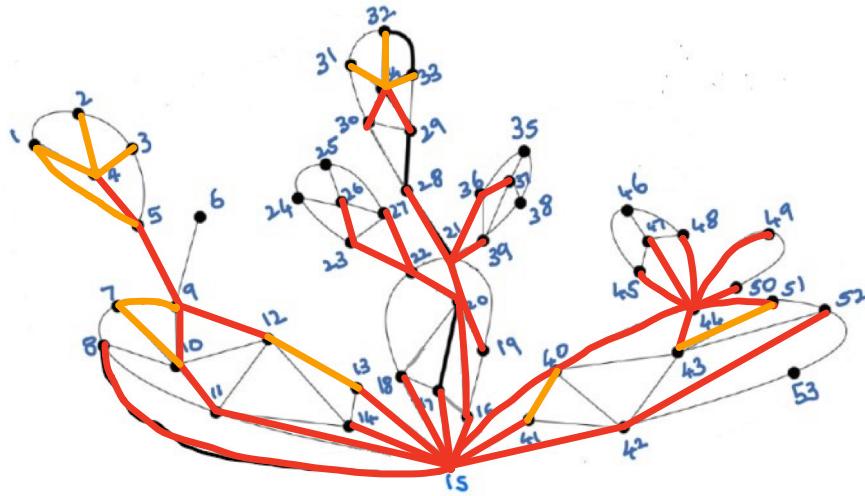
## 5) 5th step

```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[48:60]  
[(4, 5, 0.0625),  
(16, 17, 0.0625),  
(17, 18, 0.0625),  
(23, 26, 0.0625),  
(23, 27, 0.0625),  
(26, 27, 0.0625),  
(29, 30, 0.0625),  
(36, 37, 0.0625),  
(36, 39, 0.0625),  
(37, 39, 0.0625),  
(44, 49, 0.0625),  
(44, 50, 0.0625)]
```



## 6) 6th step

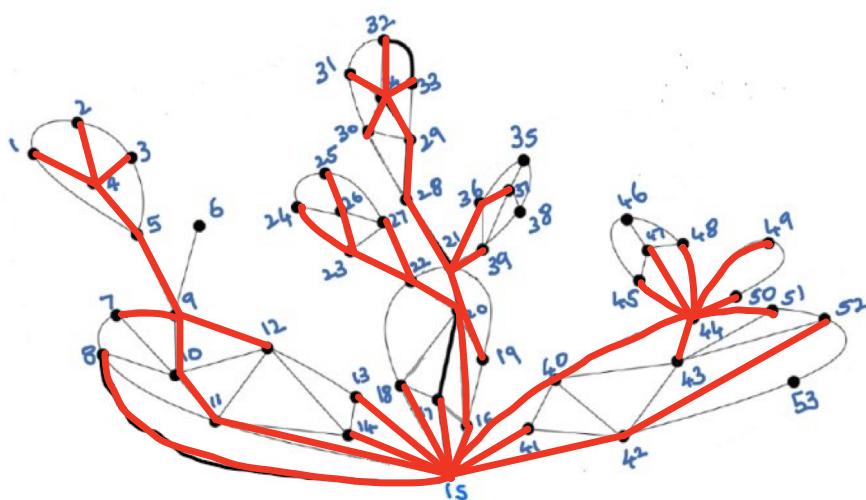
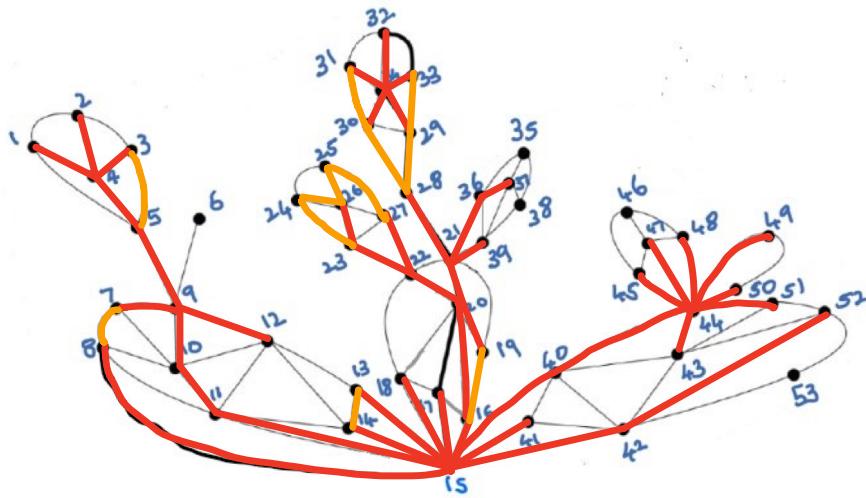
```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[60:72]  
[(7, 9, 0.06666666666666667),  
(7, 10, 0.06666666666666667),  
(12, 13, 0.06666666666666667),  
(31, 34, 0.06666666666666667),  
(32, 34, 0.06666666666666667),  
(33, 34, 0.06666666666666667),  
(40, 41, 0.06666666666666667),  
(43, 51, 0.06666666666666667),  
(1, 4, 0.0833333333333333),  
(1, 5, 0.0833333333333333),  
(2, 4, 0.0833333333333333),  
(3, 4, 0.0833333333333333)]
```



## 7) 7th step

```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[72:84]
```

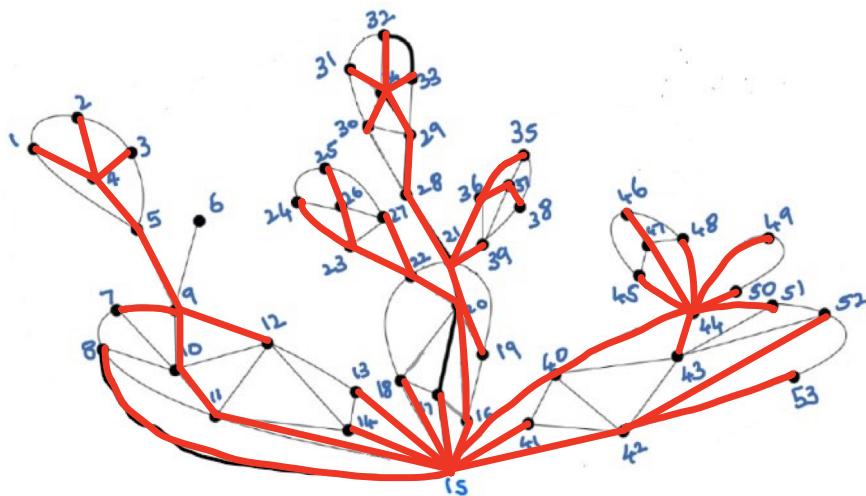
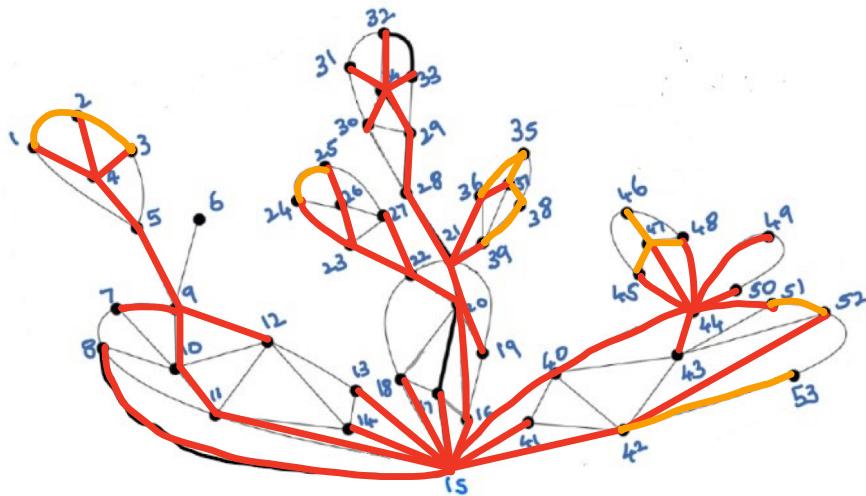
```
[(3, 5, 0.0833333333333333),
(7, 8, 0.0833333333333333),
(13, 14, 0.0833333333333333),
(16, 19, 0.0833333333333333),
(23, 24, 0.0833333333333333),
(24, 26, 0.0833333333333333),
(25, 26, 0.0833333333333333),
(25, 27, 0.0833333333333333),
(28, 29, 0.0833333333333333),
(28, 30, 0.0833333333333333),
(29, 33, 0.0833333333333333),
(30, 31, 0.0833333333333333)]
```



## 8) 8th step

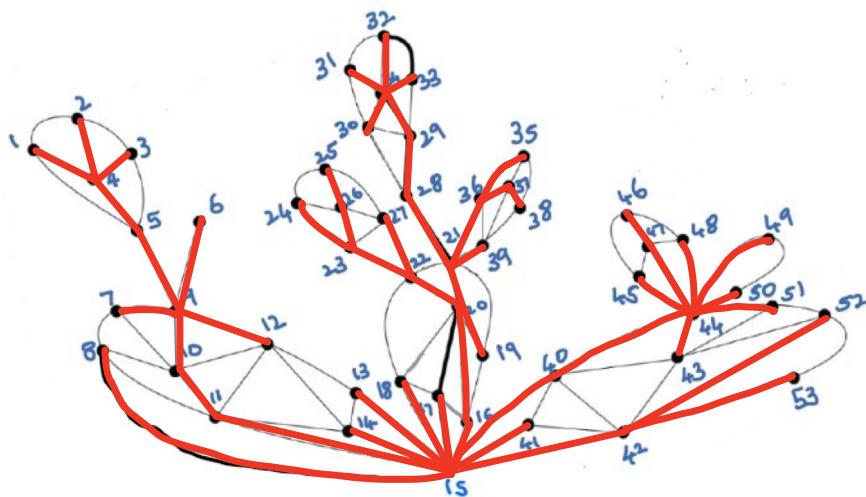
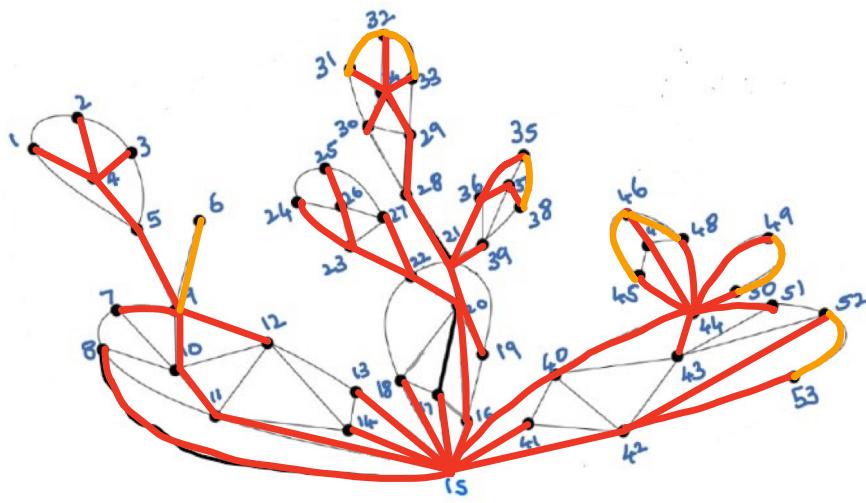
```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[84:96]
```

```
[(35, 36, 0.0833333333333333),  
(35, 37, 0.0833333333333333),  
(37, 38, 0.0833333333333333),  
(38, 39, 0.0833333333333333),  
(42, 53, 0.0833333333333333),  
(45, 47, 0.0833333333333333),  
(46, 47, 0.0833333333333333),  
(47, 48, 0.0833333333333333),  
(51, 52, 0.0833333333333333),  
(1, 2, 0.1111111111111111),  
(2, 3, 0.1111111111111111),  
(24, 25, 0.1111111111111111)]
```

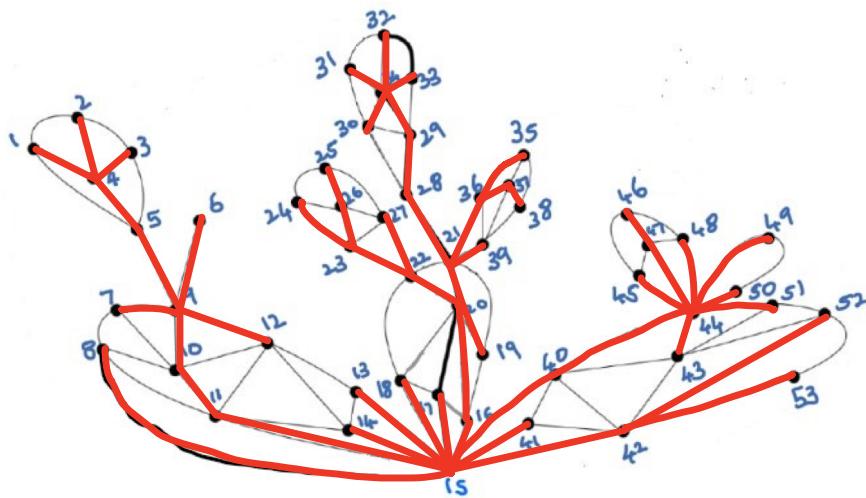


## 9) 9th step

```
sorted(rdd_edge_weight_list.collect(), key = lambda tup: tup[2])[96:108]  
[(31, 32, 0.1111111111111111),  
(32, 33, 0.1111111111111111),  
(35, 38, 0.1111111111111111),  
(45, 46, 0.1111111111111111),  
(46, 48, 0.1111111111111111),  
(52, 53, 0.125),  
(6, 9, 0.2),  
(49, 50, 0.25)]
```



#### Q4. Minimum spanning tree



**Q5. The number of I/O's of this algorithm if the block size B = 3**

The number of nodes,  $V = 53$

The number of edges,  $E = N = 104$

The number of edges that can be processed once in the RAM,  $M = 12$

The size of RAM block  $B = 3$

Implies, the log base,  $b = M / B = 12 / 3 = 4$

Assuming  $P = 1$ , the number of blocks that can be transferred simultaneously between main memory and disk.

If  $I(E)$  denote the number of I/O's to compute the MST on  $E$  edges then

$$I(E) = O(\text{sort}(E) * \log_b(V/M))$$

The number of I/Os for sort( $N$ ) =  $O((N / PB) * \log_b(N / PB)) = (104 / 3) * \log_4(104 / 3) = 88.668 \sim 89$

The total  $I(E) = O(\text{sort}(E) * \log_b(V/M)) = 89 * 1.07 = 95$

**Therefore, the number of I/O's of this algorithm if the block size B = 3 is 95**