FIT3155: Week 5 Tutorial - Answer Sheet

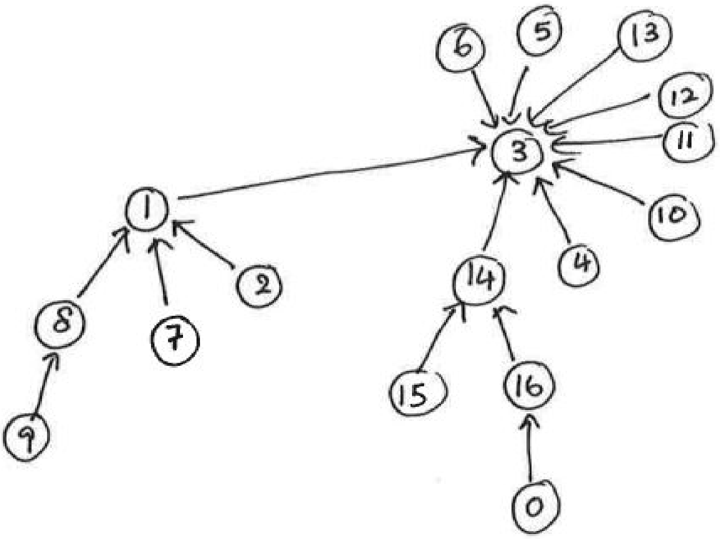
(Scribe: Dinithi Sumanaweera)

Question 1

Consider a disjoint set data structure involving 17 elements labeled {0….16}. Upon the given sequence of operations in the tutorial sheet,

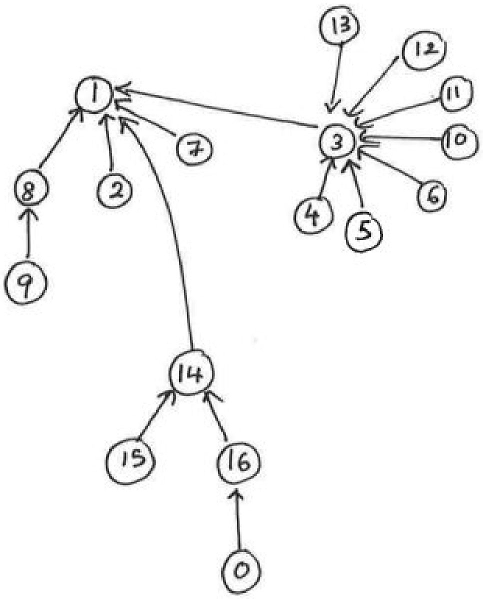
(a) Union-by-size without path compression

The final resultant structure



(b) Union-by-height without path compression

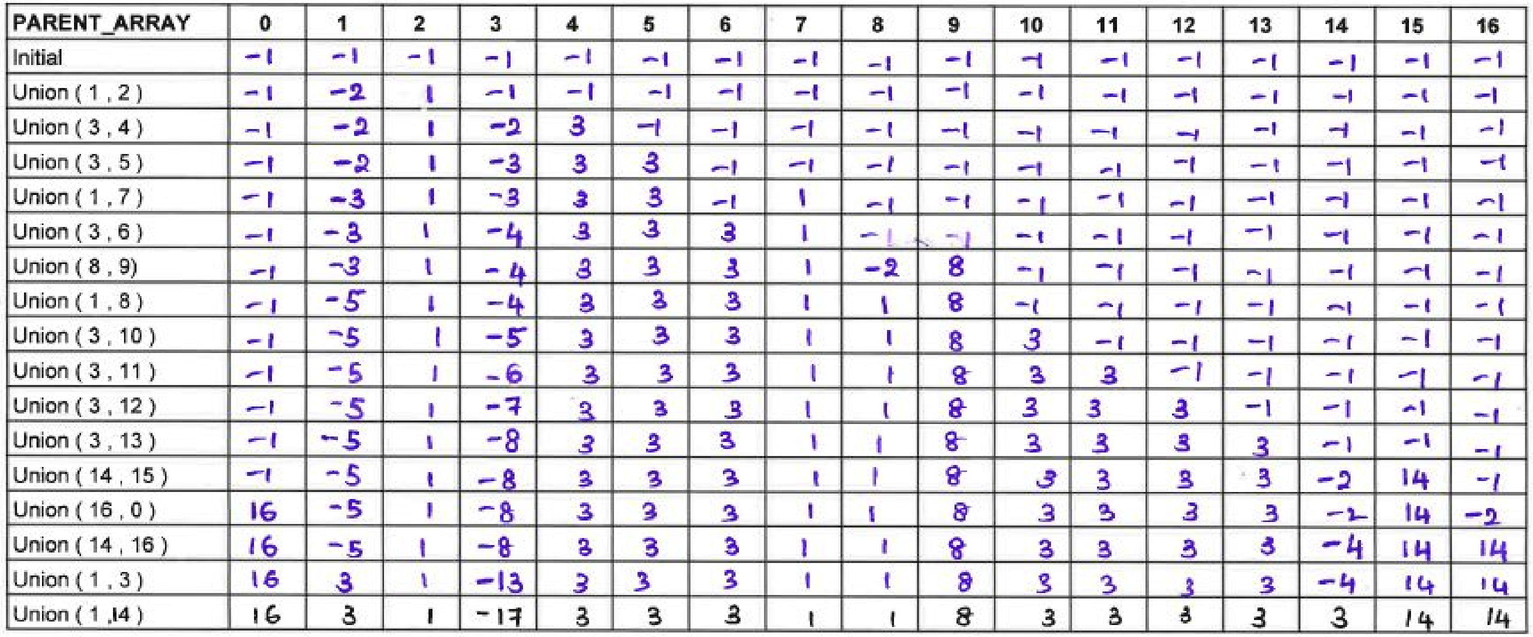
The final resultant structure



(c) Union-by-height with path compression: SELF STUDY EXERCISE

Step-by-step parent array updates for (a) and (b)

(a) Union-by-size without path compression



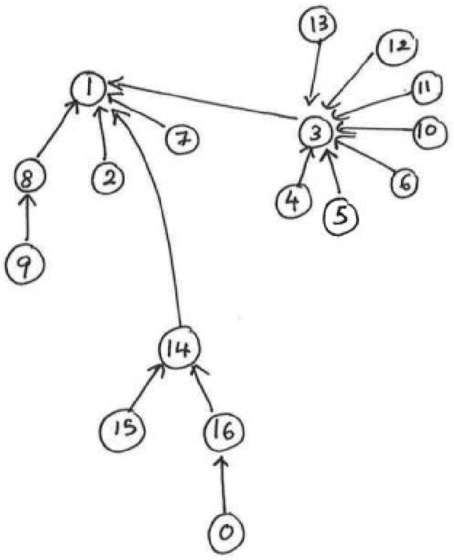
PARENT_ARRAY	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Initial	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (1 , 2)	-1	-2	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (3 , 4)	-1	-2	1	-2	3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (3 , 5)	-1	-2	1	-3	3	3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (1 , 7)	-1	-3	1	-3	3	3	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (3 , 6)	-1	-3	1	-4	3	3	3	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (8 , 9)	-1	-3	1	-4	3	3	3	1	-2	8	-1	-1	-1	-1	-1	-1	-1
Union (1 , 8)	-1	-5	1	-4	3	3	3	1	1	8	-1	-1	-1	-1	-1	-1	-1
Union (3 , 10)	-1	-5	1	-5	3	3	3	1	1	8	3	-1	-1	-1	-1	-1	-1
Union (3 , 11)	-1	-5	1	-6	3	3	3	1	1	8	3	3	-1	-1	-1	-1	-1
Union (3 , 12)	-1	-5	1	-7	3	3	3	1	1	8	3	3	3	-1	-1	-1	-1
Union (3 , 13)	-1	-5	1	-8	3	3	3	1	1	8	3	3	3	3	-1	-1	-1
Union (14 , 15)	-1	-5	1	-8	3	3	3	1	1	8	3	3	3	3	-2	14	-1
Union (16 , 0)	16	-5	1	-8	3	3	3	1	1	8	3	3	3	3	-2	14	-2
Union (14 , 16)	16	-5	1	-8	3	3	3	1	1	8	3	3	3	3	-4	14	14
Union (1 , 3)	16	3	1	-13	3	3	3	1	1	8	3	3	3	3	-4	14	14
Union (1 ,14)	16	3	1	-17	3	3	3	1	1	8	3	3	3	3	3	14	14

(b) **Union-by-height without path compression**

PARENT_ARRAY	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Initial	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (1 , 2)	-1	-2	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (3 , 4)	-1	-2	1	-2	3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (3 , 5)	-1	-2	1	-2	3	3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (1 , 7)	-1	-2	1	-2	3	3	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (3 , 6)	-1	-2	1	-2	3	3	3	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Union (8 , 9)	-1	-2	1	-2	3	3	3	1	-2	8	-1	-1	-1	-1	-1	-1	-1
Union (1 , 8)	-1	-3	1	-2	3	3	3	1	1	8	-1	-1	-1	-1	-1	-1	-1
Union (3 , 10)	-1	-3	1	-2	3	3	3	1	1	8	3	-1	-1	-1	-1	-1	-1
Union (3 , 11)	-1	-3	1	-2	3	3	3	1	1	8	3	3	-1	-1	-1	-1	-1
Union (3 , 12)	-1	-3	1	-2	3	3	3	1	1	8	3	3	3	-1	-1	-1	-1
Union (3 , 13)	-1	-3	1	-2	3	3	3	1	1	8	3	3	3	3	-1	-1	-1
Union (14 , 15)	-1	-3	1	-2	3	3	3	1	1	8	3	3	3	3	-2	14	-1
Union (16 , 0)	16	-3	1	-2	3	3	3	1	1	8	3	3	3	3	-2	14	-2
Union (14 , 16)	16	-3	1	-2	3	3	3	1	1	8	3	3	3	3	-3	14	14
Union (1 , 3)	16	-3	1	1	3	3	3	1	1	8	3	3	3	3	-3	14	14
Union (1 ,14)	16	-4	1	1	3	3	3	1	1	8	3	3	3	3	1	14	14

Question 1 - ADDITIONAL NOTE for union by height with path compression

Suppose you have a new element 17 as a single node, and the set structure you have is obtained by a union-by-rank (union by height with path compression)



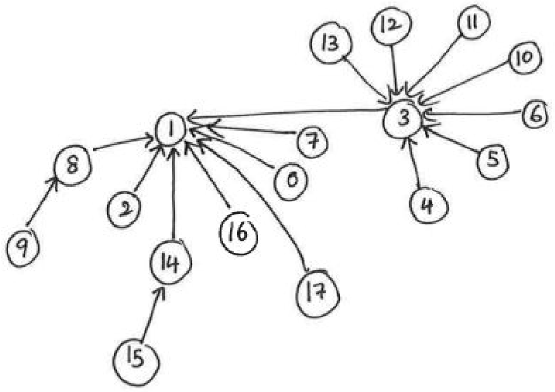
The corresponding parent array is:

```
Node:  0  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
     [16 -4 1 1 3 3 3 1 1 8  3  3  3  3  1 14 14 -1]
```

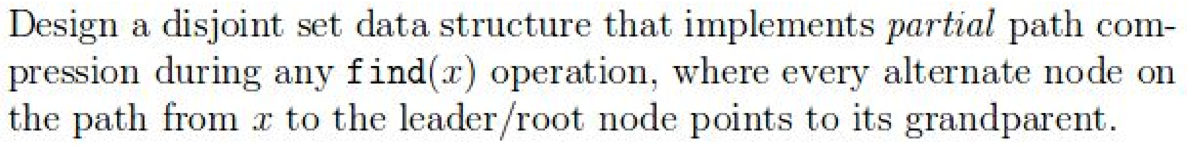
Now consider **Union(0,17) with path compression.** This involves **find(0)** and **find(17)**.
find(17): returns the leader (root) of the set as itself.
find(0): involves going through node 16 and 14 to reach the leader node (root) of the set: node 1, thus at each return call in recursive function, the parent array[0], parent array[16], parent array[14] is set to 1 (path compression). parent array[17] is set to 1 to fulfill union operation.

```
Node:  0  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
      [1 -4 1 1 3 3 3 1 1 8  3  3  3  3  1 14  1  1]
```

The resultant set structure after **Union(0,17)** is:



Question 4



Design a disjoint set data structure that implements *partial* path compression during any find(x) operation, where every alternate node on the path from x to the leader/root node points to its grandparent.

A possible solution for partial path compression

```
find(a,c){
   if(parent[a]<0){
     return <a,a>
   }else{
     <root_a,grandparent_a> = find(parent[a],c+1)
     parent_a = parent[a]
     if (c%2==0){
          parent[a] = root_a
     }else{
          parent[a] = grandparent_a
     }
     return <root_a, parent_a>
   }
}
```