WASHINGTON STATE UNIVERSITY

School of Electrical and Computer Engineering

CptS223: Advanced Data Structures C/C++

Spring 2018

<u>Homework #8</u>

<u>Due: Monday 04/09/2018 @ 11:59pm</u>

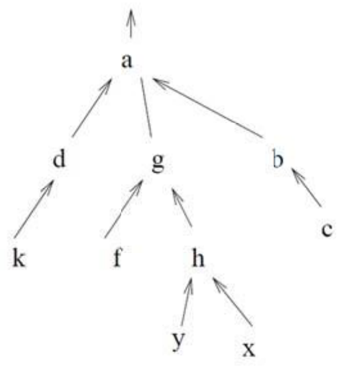
Rules/Instructions: This is an individual assignment and all work must be your own. Be sure to show your work when appropriate. This assignment must be turned in by the due date via Blackboard in *PDF* format. You may use any editor you like (or even print it out, *legibly* write in answers, and scan it in), but convert it to *PDF* for submission.

A maximum of 24 hours delay is permitted at the cost of 10% penalty. Beyond this 24-hour delay, not submissions will be graded. That is, assignments submitted after 24 hours after the above due date will not get any credits.

Grading: *50 points for each problem (i.e., Problem 1 and Problem 2).*

Feel free to use the empty space provided in this homework for filling up your answers in submitting this homework.

1. We are given two programs *A* and *B* that use two different implementations of the union-find data structure. *Program A* applies path compression when it performs each *find()* operation; whereas *Program B* does not apply path compression for its *find()* operations. Both programs start off with the same initial union-find data structure shown below. Note that this union-find data structure contains only one tree in its forest and element '*a*' is its root. Also, note that character labels are assigned for each element which is okay as long as you work with the tree representation (and not the vector implementation) for this example.



Assume that both programs perform the same sequence of *find()* operations in the following specified order from left to right:

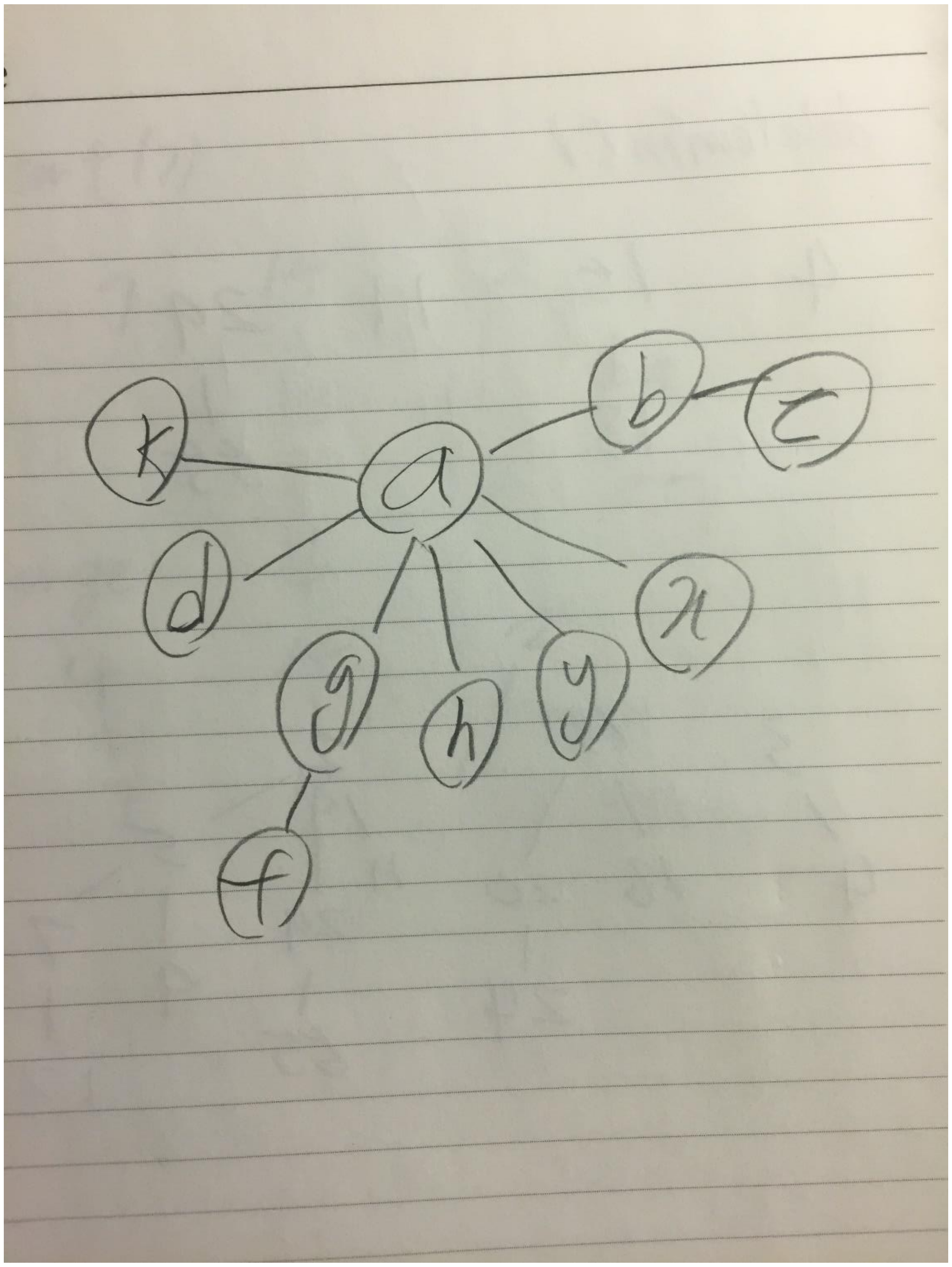
find(y), find(x), find(k), find(y), find(x), find(h)

Calculate the number of steps each of the above *find()* operations takes to climb from the element being searched to the root node. For example, the number of such steps if one were to perform a *find(d)* on the above shown initial tree will be *1* (under both programs). Give your answer by filling the number of steps for each *find()* operation in the table below:

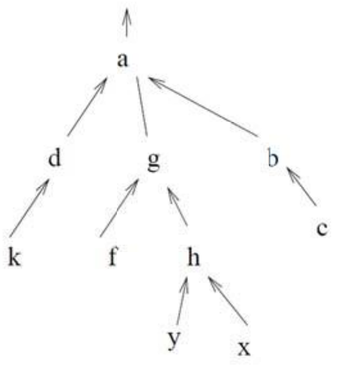
	Program A (w/ path compression)	**Program B (w/o path compression)**
1. find(y)	**3**	**3**
2. find(x)	**2**	**3**
3. find(k)	**2**	**2**
4. find(y)	**1**	**3**
5. find(x)	**1**	**3**
6. find(h)	**1**	**2**
Total	**10**	**16**

Also in the empty space provided below, show the final trees resulting from both programs, after the last find operation (i.e., *6. find(h)*).

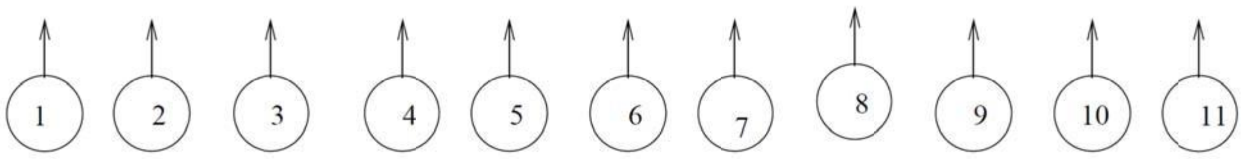
Program A's output union-find tree:


k
a
b
c
d
g
h
y
x
f

Program B’s output union-find tree:



2. Assume a union-find data structure with the following initial setting:

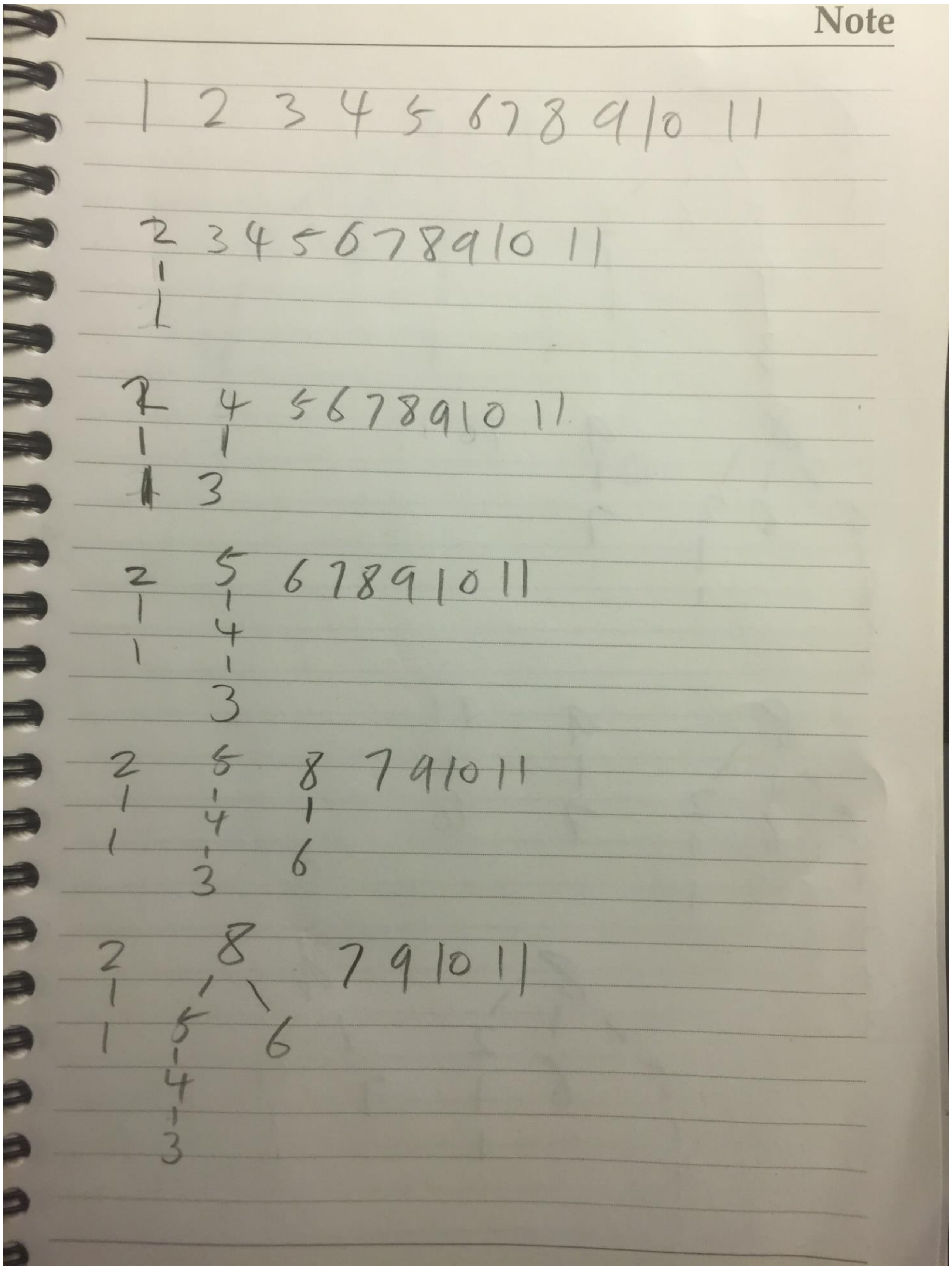


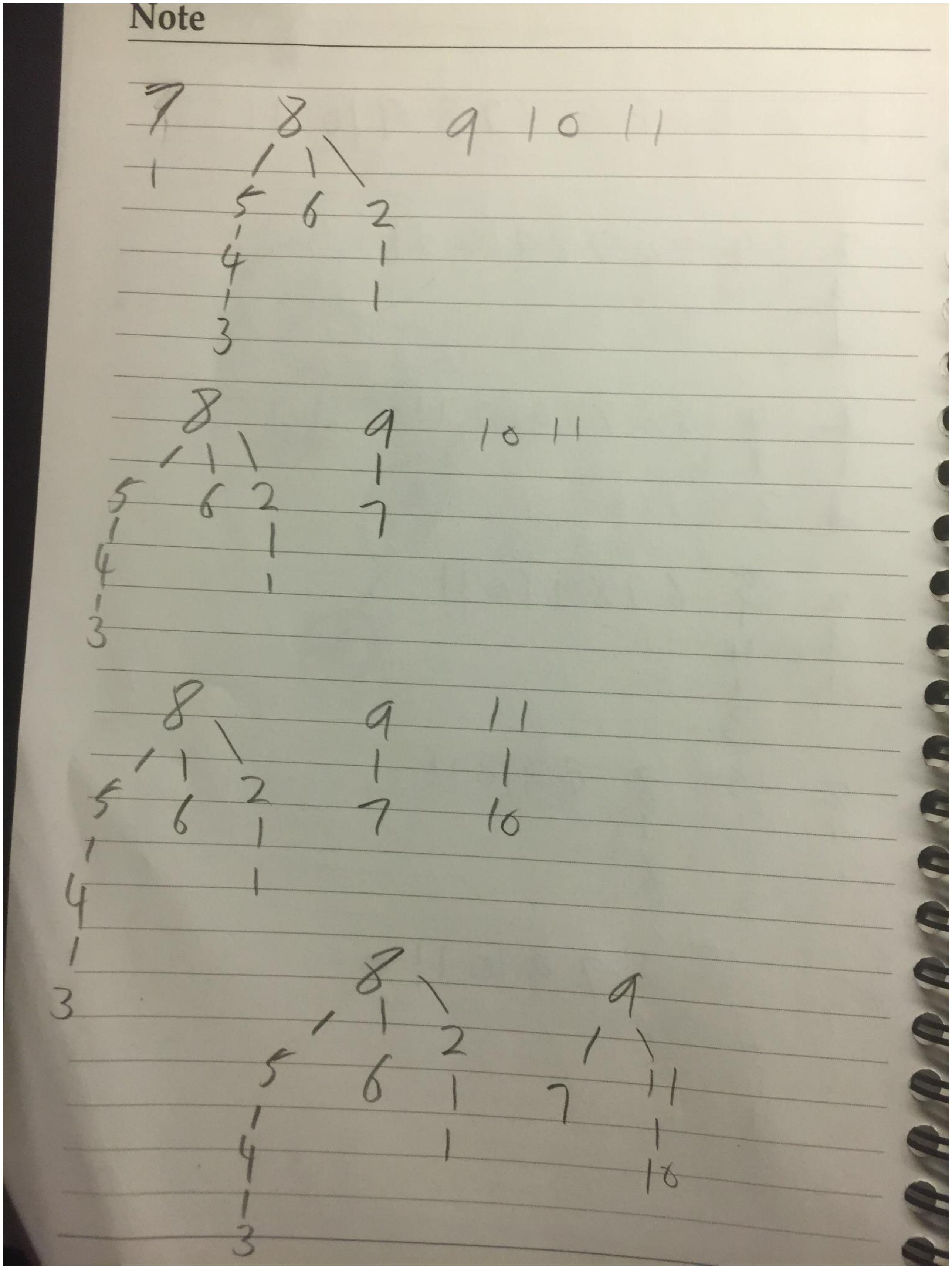
Assume that the following operations (in that order from left to right) are applied on the initial setting.

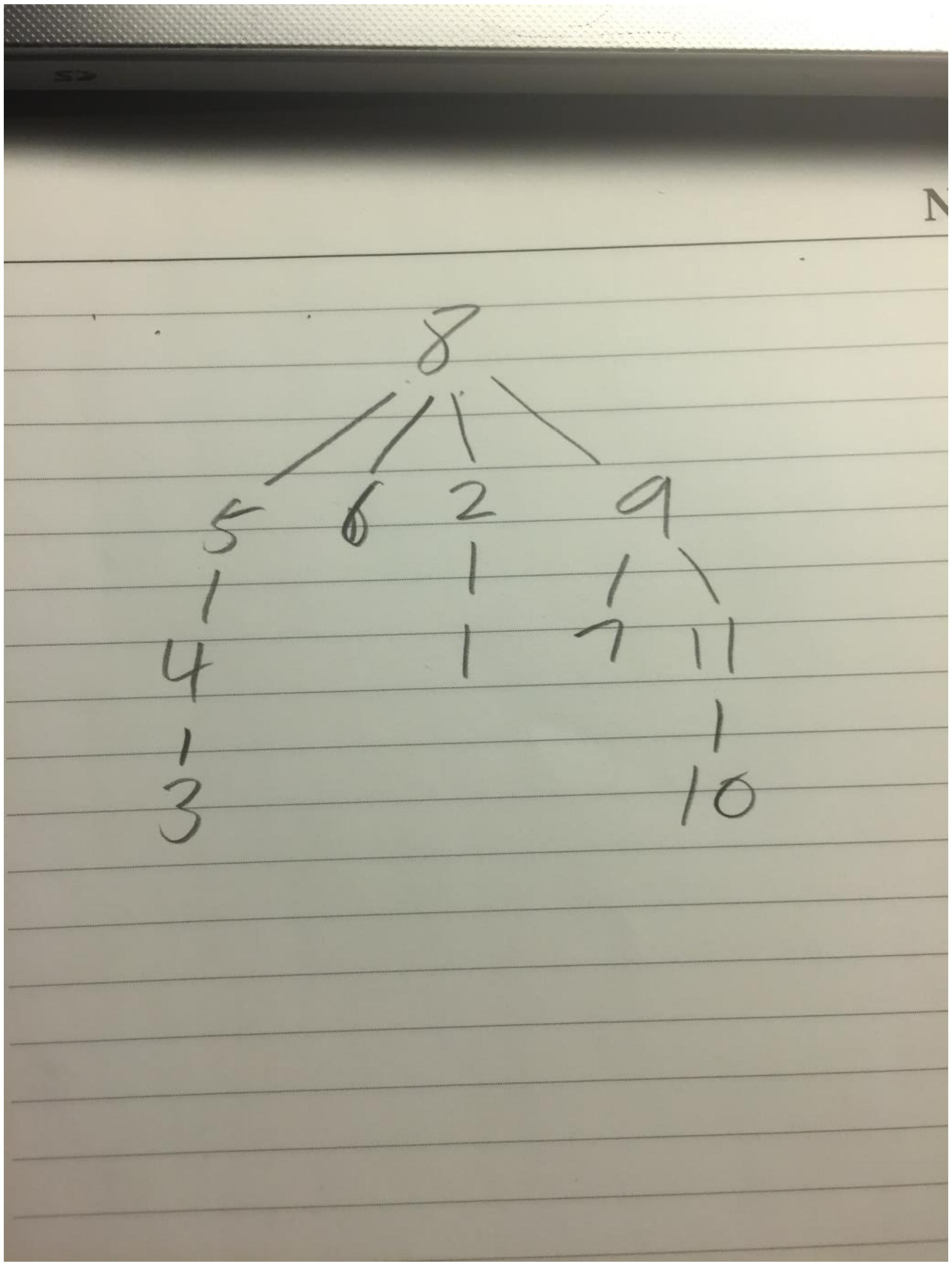
union(1, 2), union(3, 4), union(4, 5), union(6, 8), union(5, 8), union(1, 6), union(7, 9), union(10, 11), union(11, 9), union(1, 11).

Show the sequence of union-find data structures that result from applying the above-mentioned sequence of operations. Answer this question for each of the three following parts separately:

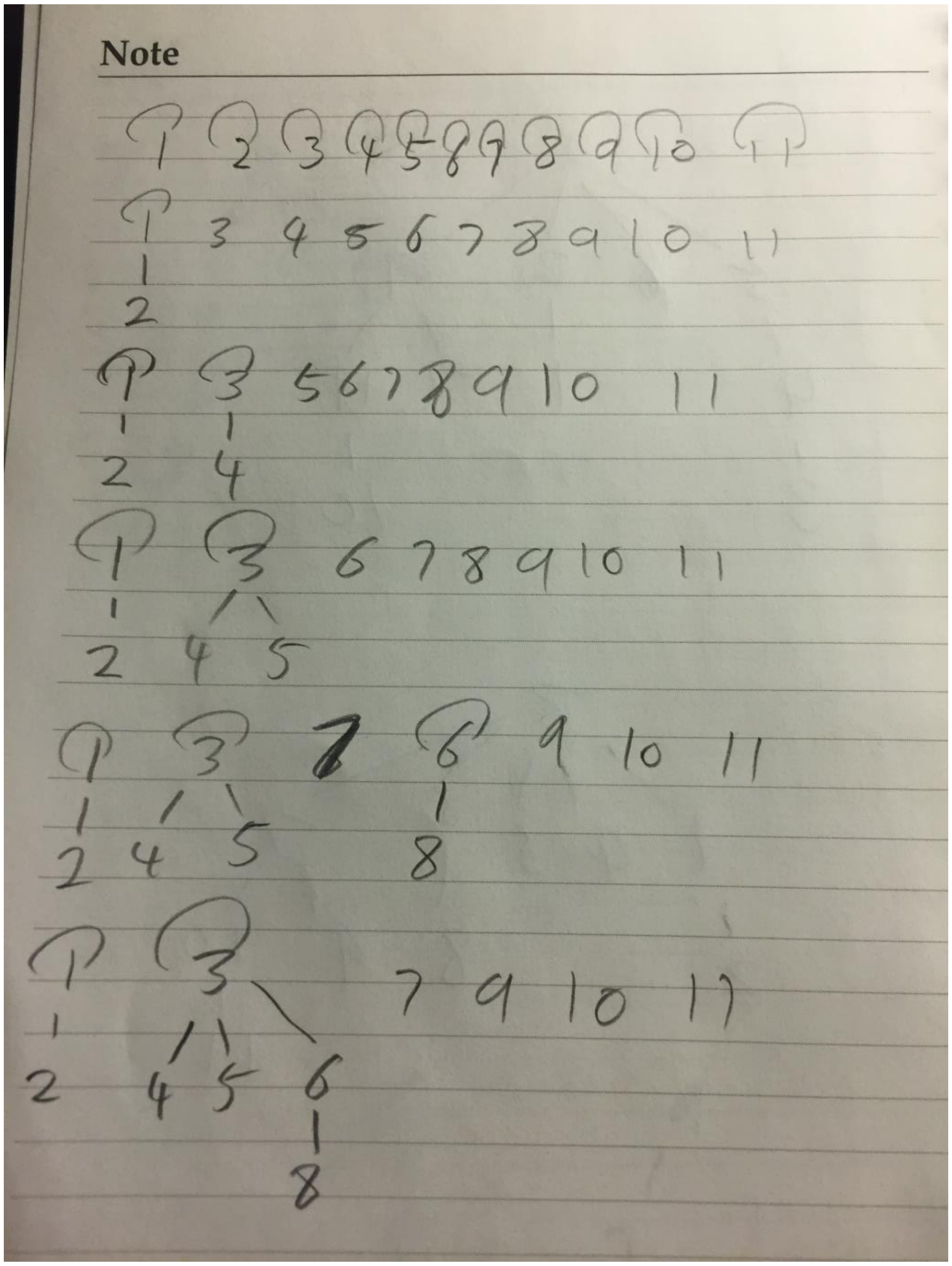
(a) The *unions* are performed by height (same as union-by-rank) and *finds* are simple.

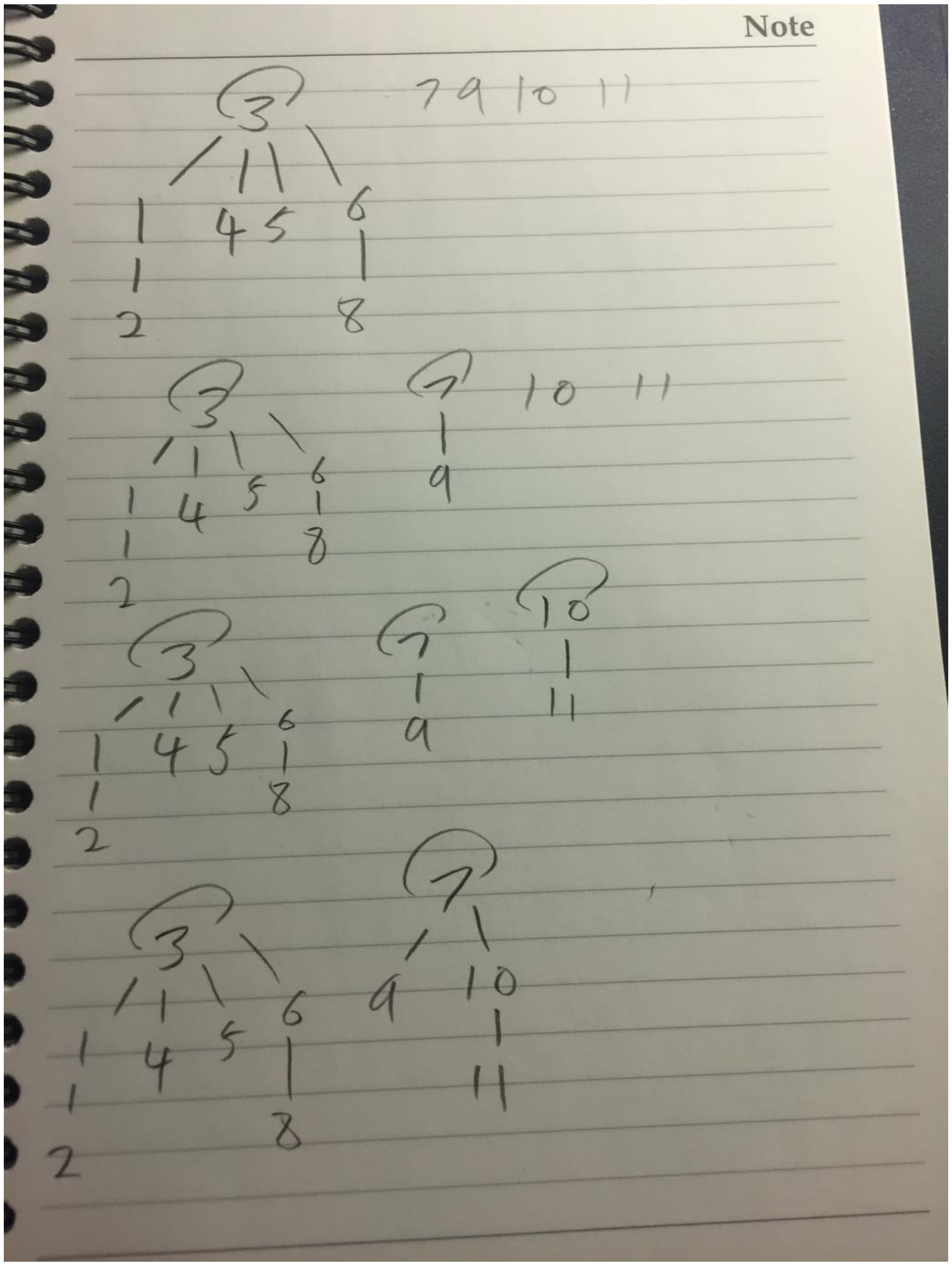

Note
1 2 3 4 5 6 7 8 9 10 11
2 3 4 5 6 7 8 9 10 11
1
2 4 5 6 7 8 9 10 11
1 3
2 5 6 7 8 9 10 11
1 4
3
2 5 8 7 9 10 11
1 4 6
3
2 8 7 9 10 11
1 5 6
4
3

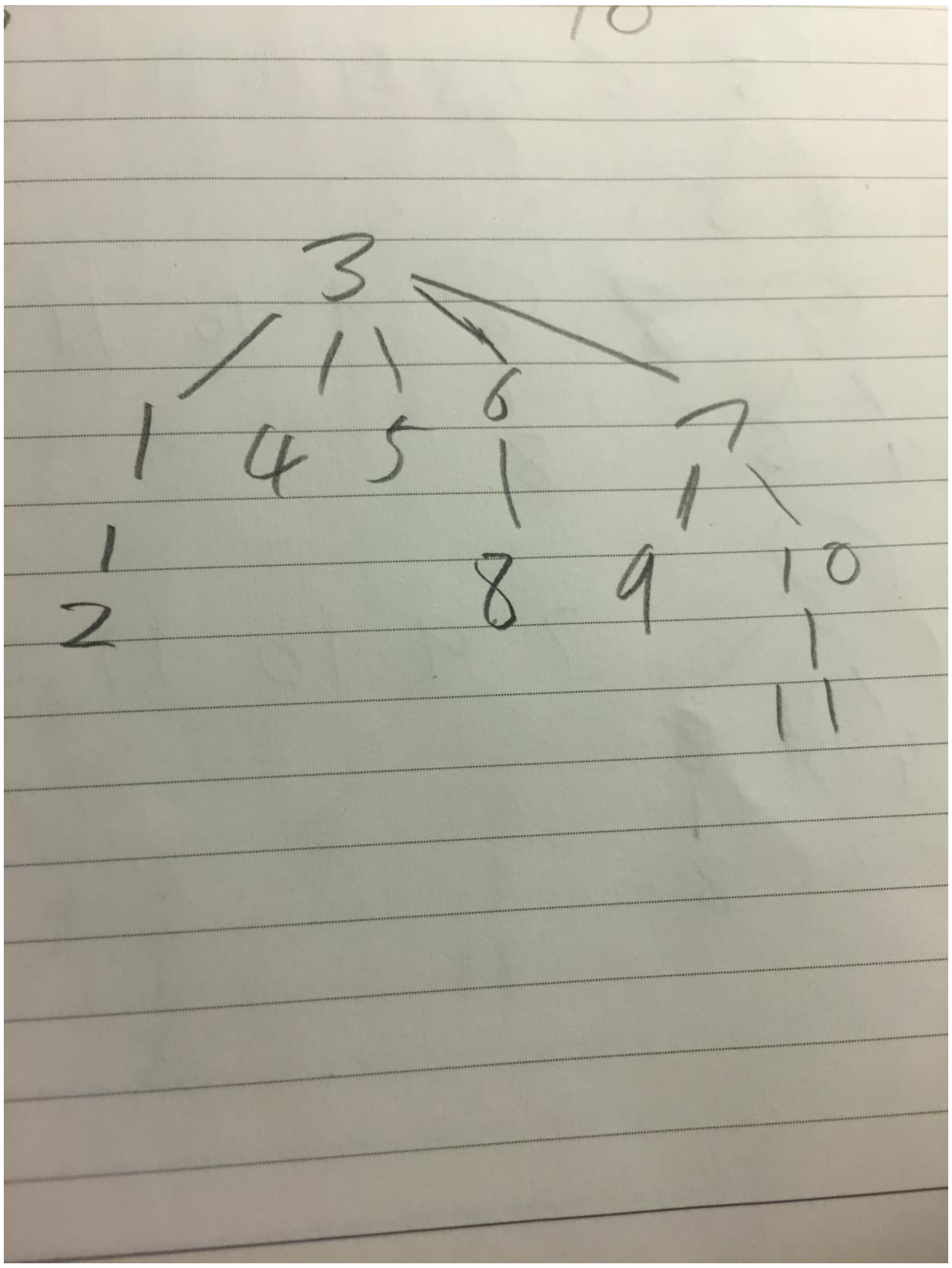

Note



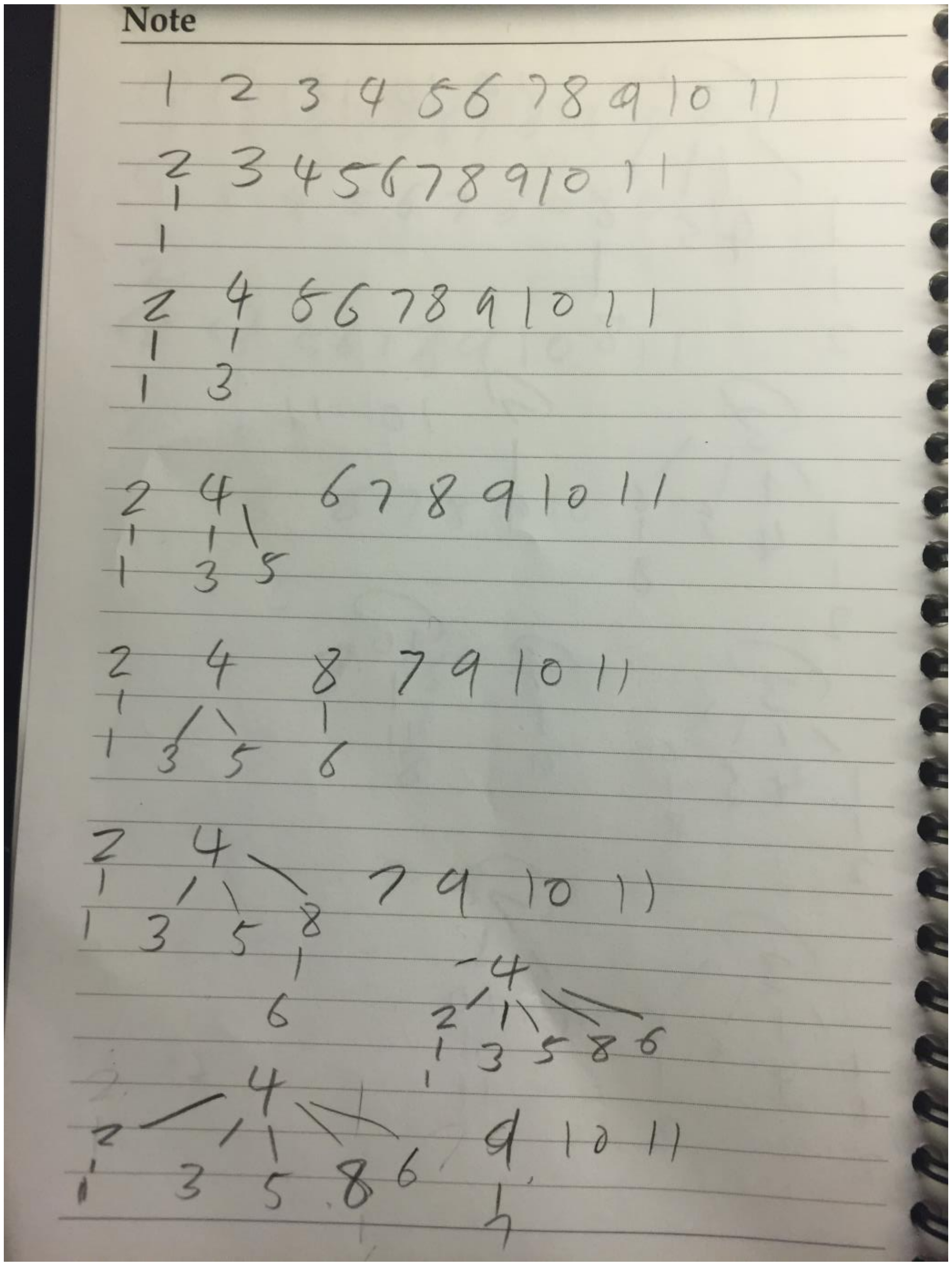
(b) The *unions* are performed by size and *finds* are simple.

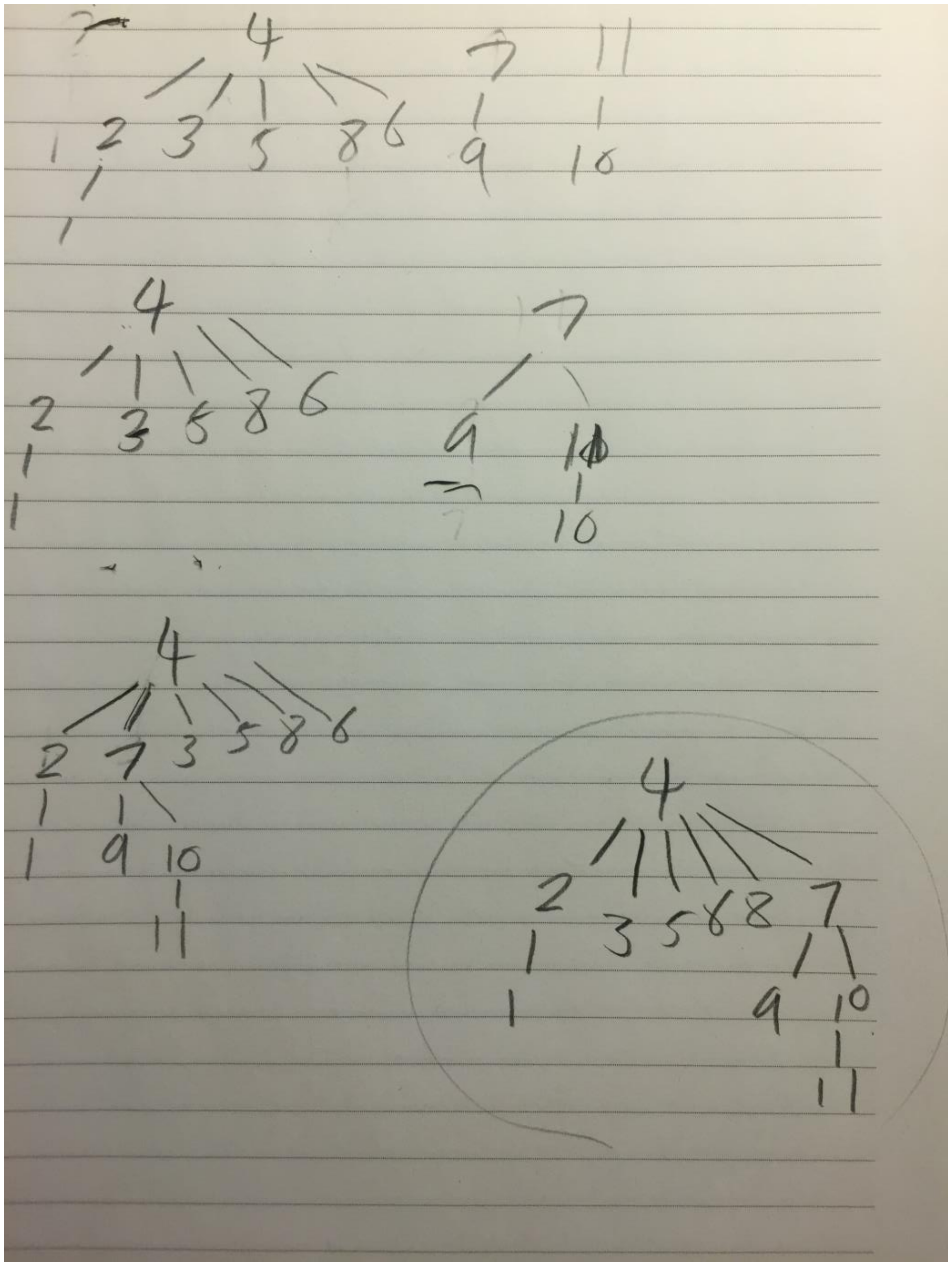

Note


Note
7 9 10 11
10 11



(c) The *unions* are performed by height and *finds* use path compression.


Note
1 2 3 4 5 6 7 8 9 10 11
2 3 4 5 6 7 8 9 10 11
2 4 5 6 7 8 9 10 11
2 4 6 7 8 9 10 11
2 4 8 7 9 10 11
2 4 7 9 10 11
9 10 11



Since 11 was called, the final one should have 10 and 11 directly attached to 4.

```
4 - 7 – 9
  - 10
  - 11
```

Note: There could be more than one correct answer sequence for each part. You just need to give one.