T.A. Help Session: Union Find

1. a. Show the resulting up Up-Tree data structure after the following unions:
   
   union(1,5), union(3,7), union(5,6), union(1,4), union(6,2), union(5,3)

b. Fill in the resulting up-tree array implementation with

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What is the worst-case runtime of a union operation? 
What is the worst-case runtime of a find operation? 

1. c. Now show the resulting up-tree data structure after the following finds, utilizing path compression:

   find(7)
   find(2)
2. Pseudocode

a. Write pseudocode for a basic union operation:
   ```java
   int[] up;
   public void union(int x, int y)
   ```

b. Write pseudocode for a union-by-weight operation:
   ```java
   int[] up;
   int[] weight;
   public void unionByWeight(int x, int y)
   ```

c. Write pseudocode for a basic find operation:
   ```java
   int[] up;
   public int find(int x)
   ```

d. Write pseudocode for a find operation, implementing path compression:
   ```java
   int[] up;
   public int findWithPathCompression(int x)
   ```