Let me analyze the path compression logic in your `findUltimateParent` method carefully.

The current implementation is:
```java
public int findUltimateParent (int node) {
    if (this.parent[node] != node) {
        // Path Compression
        this.parent[node] = this.findUltimateParent(this.parent[node]);
    }
    return node;
}
```

There's actually a bug here. After performing path compression, you're still returning the original `node` instead of the compressed path. Here's what's happening:

1. You correctly update `parent[node]` to point directly to the ultimate parent
2. But then you return `node` instead of the updated parent value

The corrected version should be:
```java
public int findUltimateParent (int node) {
    if (this.parent[node] != node) {
        this.parent[node] = this.findUltimateParent(this.parent[node]);
        return this.parent[node];  // Return the compressed path
    }
    return node;
}
```

Or more concisely:
```java
public int findUltimateParent (int node) {
    if (this.parent[node] != node) {
        return this.parent[node] = this.findUltimateParent(this.parent[node]);
    }
    return node;
}
```

The issue you're seeing in your results is because:
1. The path compression is actually happening (the parent pointers are being updated)
2. But the method returns the wrong value (the original node instead of its ultimate parent)
3. This causes incorrect union operations later

Your suggested fix of `node = this.findUltimateParent(this.parent[node])` wouldn't work because:
1. It wouldn't update the parent array (no path compression)
2. It would just modify the local `node` variable

The correct approach is to both:
1. Update the parent array (path compression)
2. Return the ultimate parent (not the original node)

This is why you're seeing incorrect results - nodes aren't being properly connected in the DSU structure because the `find` operation isn't returning their true ultimate parents.