**CS146 HW1**

**Problem 1:**

Ans: As you can see in the union() in

public void union(int p, int q) {

if (connected(p, q)) return;

for (int i = 0; i < id.length; i++)

if (id[i] == id[p]) id[i] = id[q];

count--;

}

The value of id[p] changes to id[q] and if a > b with id[a] = id[b] will not be updated which is equal to id[q].

Examples: union(0,1) => 1 1 2 3 4 5 …

Union(0,2) => 2 1 2 3 4 5…

**Problem 2:**

Ans: If we set id[root(p)] = q instead of id[root(q)] the resulting algorithm would be correct. If we do this we increase the tree height. No there are no performance consequences.

**Problem 3:**

Ans: Coded (Program works for all inputs)

**Problem 4:**

Ans:

**Quick Union**

**smallUF.txt took : 0.154ms**

**mediumUF.txt took: 0.241ms**

**largeUF.txt took a lot of time to compute**

**QuickUnionPathCompressionUF**

**smallUF.txt took: 0.136ms**

**mediumUF.txt took: 0.24ms**

**largeUF.txt: 5.126**

Quick Union will have O(nm) in worst case and Quick Union Path compression will have O(n+mlogn).Quick Union path compression takes o(n) linear time for initializing with n. find and connected and union takes total running time is amortized) logn and count is o(1).Quick Union takes o(n) for union, find and connected and count takes o(1).

QuickUnionPathCompression is faster because it employs a technique called path compression which flattens the structure of tree every node point to the root where find is used. The resulting tree is faster and speeds up operations. It first finds the root then updates it parents.find(p) is called then it is pointed to the root.

Example: 5 – 4 – 5-3-1-5-23-5-1-

When find() is called for 3 then 3 points to 5 hence the path of root is reduced.

**Problem 5: Coded (passed all test cases)**

**Problem 6: Coded (passed all test cases)**

**Problem 7: Coded (passed all test cases)**