Assignment # 4

Subject: Advance Analysis of Algorithm

Submitted to: Dr. Saqib

Submitted by: Phool Fatima

Roll #: 2433

Reg #: 2022-ag-2433

Anatomy of Weighted Quick-Union

Public class WeightedQuickUnionUF {

This line indicate the weighted QuickUnionUF class. In support classic union-find operations, along with a count operation that return the total number of sets

Private int [] parent;

parent[i] = parent of i

Private int[] size;

size[i] = number of elements in subtree rooted at i;

Private int count;

To count number of components

Public WeightedQuickUnionUF(int n)

{

count = n;

Parent = new int [n];

Size = new int [n];

for (int i=0; i<n; i++){

parent[i] = i;

size[i] = 1; }

}

Initializes an empty union-find data structure with n elements through n-1. Initially, each element is in its own set, n is number of elements.

```
public int count() {
    return count; }
```

Returns the number of sets, between 1 and n.

```
public int find(int p) {
    validate (p);
    while (p != parent(p))
        p = parent (p);
    return p;
    }
```

It return the canonical element of the set containing element p.

Parameter p an element throws illegal argument exception unless 0 <= p < n.

@Deprecated
public boolean connected(int p, int q)
{
return find(p) = find(q);
}

p one element
q the other element
It return true if p and q are in the same set. If not it return false.
Throws illegal arguments exception unless both 0<=p<n and 0<=q<n
Deprecated => Replace with two calls to (find(int))

Private void validates(int p){
Int n= parent length;
if (p<0 || p>=n) {
Throw new IllegalArgumentException
("index" ++" is not between
0 and" + (n-1)); }
}

It is used to validate that p is a valid index.
void does not return value int=> IS

a data type used a loop to check validation of p..

```
Public void union (int p, int q) {
    int root p = find (p);
    int root Q = find (q);
    if (root p == root Q) return;
```

Merges the set containing element p with the set containing element q.

p => one element.

q <= the other element.

Throws illegal argument exception unless both the 0 <= pcn and 0 <= qcn

```
    if (size [root p] < size [root Q]) {
        parent [root p] = root Q;
        Size [root Q ]+ = size [root P];
    }
    else
        {
    Parent [root Q] = root P;
        Size [root P]+ = Size [root Q];
    }
    }
        count -- ;
    }
```

In this roots are being compared to each other smaller roots are being compared to larger and larger with smaller root point to larger one.

```
Public static void main(String[]args)
{
    int n = stdIn.readInt();
Weighted QuickUnion Uf uf = new
Weighted QuickUnion (int
        while (!StdIn.isEmpty()){
        int p= stdIn.readInt();
        int q= StdIn.readInt();
       if (uf.find(p)== uf.find(q))
       continue;
       uf.union (p, q);
      stdout.println (p+ " " +q);
    }
      stdout. println (uf.count()+ "comp");
  }
  }
```

Reads an integer and a sequence of pairs of integer between 0 and n-1 from standard input, where each integer in the pair represents some

elements. If the elements are in different set merge the two sets and print the pair to standard output.