CRACK A HACK

THE VALUE OF FRIENDSHIP

Course: ALGORITHMIC PROBLEM SOLVING

Course code: 17ECSE309

by:

Veeresh Karikai

USN: 01FE15BCS222

1. Introduction

What is a disjoint-set?

A disjoint-set is a data structure that keeps track of a set of elements partitioned into a number of disjoint (non-overlapping) subsets. In other words, disjoint set is a group of sets where no item can be in more than one set. It is also called a union–find data structure as it supports union find operations on subsets.

Find: It determines in which subset a particular element is in and returns the representative of that particular set. An item from this set typically acts as a representative of the set

Union: It merges two different subsets into a single subset and representative of one set becomes representative of other.

The disjoint-set also supports one other important operation called Make set which creates a set containing only a given element in it.

2. Example

For a set of elements S = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Here you have 10 elements (N = 10). We can use an array **Arr** to manage the connectivity of elements. Arr[ ] indexed by elements of set, having size of N (as N elements in set) and can be used to manage the above operations.

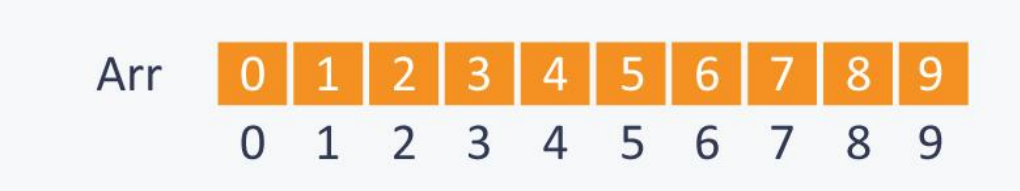
Assumption: A and B objects are connected only if Arr[A] = Arr [B].

Now how we will implement above operations :

Find (A, B) - check if Arr[A] is equal to Arr[B] or not. Union (A, B) - Connect A to B and merge the components having A and B by changing all the elements ,whose value is equal to Arr[A], to Arr[B].

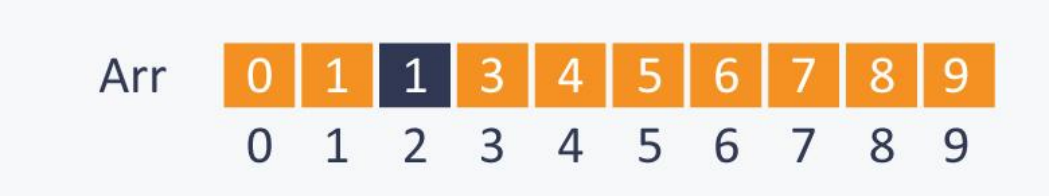
Initially there are 10 subsets and each subset has single element in it.

When each subset contains only single element, the array Arr is:

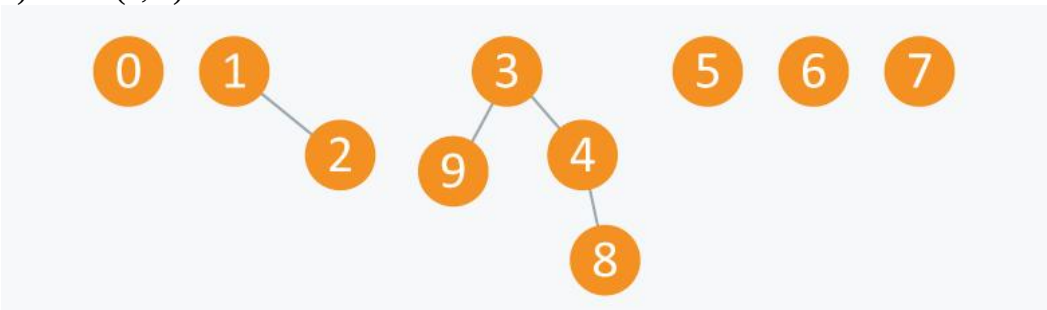


After performing Operations: 1) Union(2, 1)

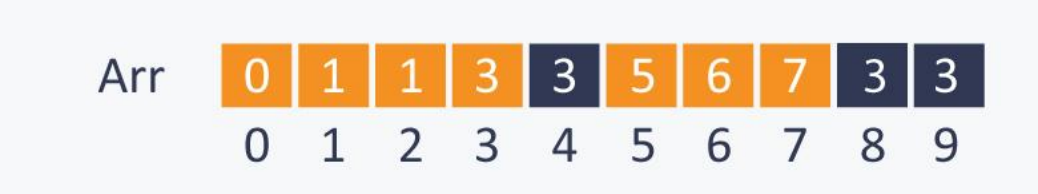
Arr will be:



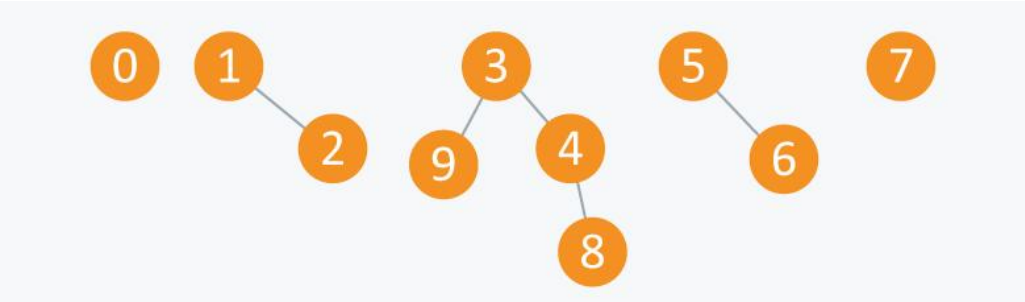
2) Union(4, 3)
3) Union(8, 4)
4) Union(9, 3)



Arr will be:



5) Union(6, 5)



Arr will be:

After performing some operations of Union(A ,B), you can see that now there are 5 subsets. First has elements {3, 4, 8, 9}, second has {1, 2}, third has {5, 6}, fourth has {0} and fifth has {7}. All these subsets are said to be Connected Components.

One can also relate these elements with nodes of a graph. The elements in one subset can be considered as the nodes of the graph which are connected to each other directly or indirectly, therefore each subset can be considered as **connected component.**

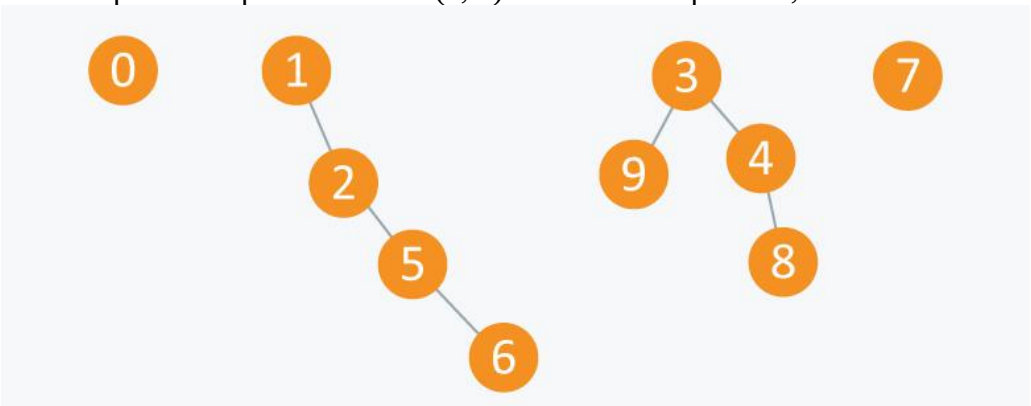
From this, we can infer that Union-Find data structure is useful in Graphs for performing various operations like connecting nodes, finding connected components etc.

Let's perform some Find(A, B) operations. 1) Find (0, 7) - as 0 and 7 are disconnected ,this will gives false result.
2) Find (8, 9) -though 8 and 9 are not connected directly ,but there exist a path connecting 8 and 9, so it will give us true result.

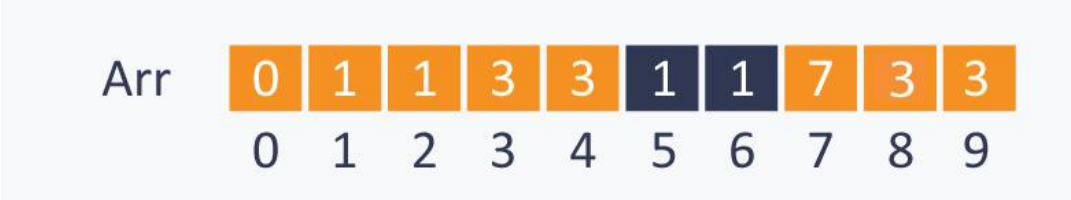
When we see above operations in terms of components, then :

Union(A, B) - Replace components containing two objects A and B with their union. Find(A, B) - check if two objects A and B are in same component or not.

So if we perform operation Union(5, 2) on above components, then it will be :



Now the Arr will be:



3. Algorithm

```
//initialization of array elements

void initialize( int Arr[ ], int N)
{
   for(int i = 0;i<N;i++)
            Arr[ i ] = i ;
}

//returns true,if A and B are connected, else it will return false.

bool find( int Arr[ ], int A, int B)
{
            if(Arr[ A ] == Arr[ B ])
                        return true;
            else
                        return false;
}

///change all entries from Arr[ A ] to Arr[ B ].

void union(int Arr[ ], int N, int A, int B)
{
            int TEMP = Arr[ A ];
            for(int i = 0; i < N;i++)
            {
                        if(Arr[ i ] == TEMP)
                        Arr[ i ] = Arr[ B ];
            }
}
```

4. Code

```
#include <bits/stdc++.h>
using namespace std;
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <bits/stdc++.h>
#include<vector>

int find(vector<int> &arr, int i){

        if (arr[i]!=i)
      arr[i]=find(arr, arr[i]);
   return  arr[i];
}

void unionn(vector<int> &arr, int x, int y){

        int x1=find(arr,x);
   int y1=find(arr,y);
   arr[x1]=y1;
}

int main(){
   int t;
   cin>>t;
   int n,m,x,y;
   for(int a0 = 0; a0 < t; a0++){
      cin>>n>>m;
                vector<int> arr(n+1);
      vector<long> count(n+1);
      long long total=0,abc=0;
      long loop=0;

      for(int i=1;i<=n;i++){
         arr[i]=i;
      }
      for(int a1 = 0; a1 < m; a1++){
         cin>>x>>y;
                        if(find(arr,x)!=find(arr,y))
                           unionn(arr,x,y);
```

```
            else
                loop++;
        }
        for(int i=1;i<=n;i++)
            find(arr,i);

        for(int i=1;i<=n;i++){
            count[arr[i]]++;
            count[i]--;
        }

        long long c,c1,c2;
        sort(count.begin(),count.begin()+n+1);
        for(int i=n;i>=1;i--){
            if(count[i]<=0)
                break;
            long temp=0;
            c=count[i];
            c1=abc*c;
            c2=(c*(c+1)*(c+2))/3;
            total=total+c1+c2;
            abc=abc+(c*(c+1));
        }
        total=total+(abc*loop);
        cout<<total<<endl;
    }
    return 0;
}
```

5. References

1. http://www.techiedelight.com/disjoint-set-data-structure-union-find-algorithm/
2. https://www.hackerearth.com/practice/notes/disjoint-set-union-union-find/
3. https://github.com/prakashbh/day-today-codes/blob/master/10-union-find-basic.c/
4. https://www.hackerrank.com/challenges/value-of-friendship/forum