Practical 8

# Student Details

* Name : Tushar Harsora
* Roll : 19BCE509

# Aim

The Aim of this Practical is to implement the simple union find structure. The union and find set is done in O(1) time. The union operation uses the path compression and the find also uses the path compression so amortized complexiety is O(1)

# Code

* Set\_union.cpp

#include <iostream>

#include <vector>

#include <array>

using namespace std;

template<int size>

struct dsu {

array<int, size> par;

dsu() {

for (int i = 0; i < size; i++)

par[i] = i;

}

int find(int root) {

if (root == par[root])

return root;

return par[root] = find(par[root]);

}

void union\_dsu(int x, int y) {

int xroot = find(x);

int yroot = find(y);

if (xroot != yroot) {

par[xroot] = yroot;

}

}

void list\_set(int root) {

int target = find(root);

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

if (find(i) == target) {

cout << i << "\t";

}

}

cout << endl;

}

friend ostream& operator<<(ostream& stream, dsu<size>& d) {

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

stream << i << " = " << d.find(i) << "\t";

}

return stream;

}

};

int main() {

dsu<10> d;

d.union\_dsu(0, 2);

d.union\_dsu(2, 5);

d.union\_dsu(5, 7);

d.union\_dsu(8, 9);

d.list\_set(0);

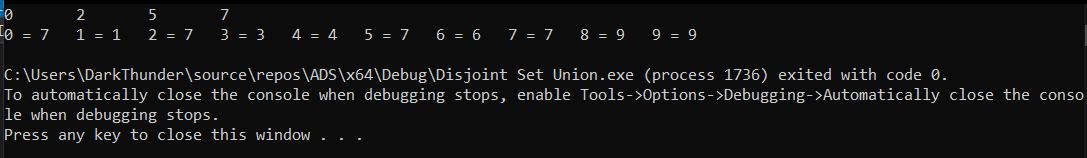
cout << d << endl;

}

# Inputs

This Program Doesn’t take any inputs.

# Output



# Conclusion

Here we learned about the union and find operations of the disjoint set union data structure. And their time complexity and implementation