Assignment-1(Disjoint Sets)

Name- Prashant Singh

Course-Mtech(CS)

Enrollment Number: SAU/CS/Mtech(CS)/2024/04

Question-1 Consider the set of natural numbers <= 20 and find a set of all the even numbers in the given set. ● ● Input: Natural Numbers Operations: ○ Union(2, 4)

○ Union(2, 6)

○ .

○ .

○ .

○ Union(2, 20)

○ Find-Set(2)

○ Find-Set(12)

○ Find-Set(5)

○ Find-Set(17)

Code:

```
#include <stdio.h>

int parent[21];
int rank[21];


int find_set(int x) {
    if (parent[x] != x) {
        parent[x] = find_set(parent[x]);
    }
    return parent[x];
}


void union_sets(int x, int y) {
    int rootX = find_set(x);
    int rootY = find_set(y);

    if (rootX != rootY) {
        if (rank[rootX] > rank[rootY]) {
            parent[rootY] = rootX;
        } else if (rank[rootX] < rank[rootY]) {
            parent[rootX] = rootY;
        } else {
            parent[rootY] = rootX;
            rank[rootX]++;
        }
    }
}


void intial_set() {
    for (int i = 1; i < 21; i++) {
        parent[i] = i;
        rank[i] = 0;
    }
}


void print_set() {
    int representative[21] = {0};
    int root_of_even = find_set(2);

    printf("All sets:\n");
    for (int i = 1; i < 21; i++) {
        int rep = find_set(i);
        if (representative[rep] == 0) {
            representative[rep] = 1;
            printf("Representative %d has following elements: ", rep);
            for (int j = 1; j < 21; j++) {
                if (find_set(j) == rep) {
                    printf("%d ", j);
                }
            }
            printf("\n");
        }
    }


    printf("\nEven numbers in the set: ");
    for (int i = 2; i < 21; i += 2) {
        if (find_set(i) == root_of_even) {
            printf("%d ", i);
        }
    }
    printf("\n");
}

int main() {
    intial_set();
 //print all Sets
    print_set();
    printf("\n");

 //Making Set of all even Numbers
    for (int i = 2; i <= 20; i += 2) {
        union_sets(2, i);
    }

//Even Number Set
    print_set();
    printf("\n");

//Operations Asked in the Assignment
    printf("find_set-Set(2): %d\n", find_set(2));
    printf("find_set-Set(12): %d\n", find_set(12));
    printf("find_set-Set(5): %d\n", find_set(5));
    printf("find_set-Set(17): %d\n", find_set(17));

    return 0;
```

}

```
All sets:
Representative 1 has following elements: 1
Representative 2 has following elements: 2
Representative 3 has following elements: 3
Representative 4 has following elements: 4
Representative 5 has following elements: 5
Representative 6 has following elements: 6
Representative 7 has following elements: 7
Representative 8 has following elements: 8
Representative 9 has following elements: 9
Representative 10 has following elements: 10
Representative 11 has following elements: 11
Representative 12 has following elements: 12
Representative 13 has following elements: 13
Representative 14 has following elements: 14
Representative 15 has following elements: 15
Representative 16 has following elements: 16
Representative 17 has following elements: 17
Representative 18 has following elements: 18
Representative 19 has following elements: 19
Representative 20 has following elements: 20

Even numbers in the set: 2

All sets:
Representative 1 has following elements: 1
Representative 2 has following elements: 2 4 6 8 10 12 14 16 18 20
Representative 3 has following elements: 3
Representative 5 has following elements: 5
Representative 7 has following elements: 7
Representative 9 has following elements: 9
Representative 11 has following elements: 11
Representative 13 has following elements: 13
Representative 15 has following elements: 15
Representative 17 has following elements: 17
Representative 19 has following elements: 19
```

```
Even numbers in the set: 2 4 6 8 10 12 14 16 18 20

Find-Set(2): 2
Find-Set(12): 2
Find-Set(5): 5
Find-Set(17): 17
```

Question-2

Implement the connected component application (discussed in the class). Graphically show its time complexity. For instance, if there are n operations in all, then take random number of operations of each kind(i.e. MAKE_SET, FIND_SET and UNION), calculate the time complexity and plot. (Also, for the plot you need to capture results for n=50, 100, ... 400 with a scale difference of 50 or so.)

Code

```
#include <stdlib.h>
#include <time.h>

#define MAX 401

int parent[MAX];
int rank[MAX];


void intial_set(int n) {
    for (int i = 1; i <= n; i++) {
        parent[i] = i;
        rank[i] = 0;
    }
}


int find_set(int x) {
    if (parent[x] != x) {
        parent[x] = find_set(parent[x]);
    }
    return parent[x];
}


void union_sets(int x, int y) {
    int rootX = find_set(x);
    int rootY = find_set(y);

    if (rootX != rootY) {
        if (rank[rootX] > rank[rootY]) {
            parent[rootY] = rootX;
        } else if (rank[rootX] < rank[rootY]) {
            parent[rootX] = rootY;
        } else {
            parent[rootY] = rootX;
            rank[rootX]++;
        }
    }
}

void perform_operations(int n, int* make_set_count, int* find_set_set_count, int* union_count) {
    srand(time(NULL));

    for (int i = 0; i < n; i++) {
        int op = rand() % 3;
        int x = rand() % n + 1;
        int y = rand() % n + 1;

        if (op == 0) {
            (*make_set_count)++;
        } else if (op == 1) {
            find_set(x);
            (*find_set_set_count)++;
        } else if (op == 2) {
            union_sets(x, y);
            (*union_count)++;
        }
    }
}

int main() {
    int sizes[] = {50, 100, 150, 200, 250, 300, 350, 400};
    int num_sizes = sizeof(sizes) / sizeof(sizes[0]);


    for (int i = 0; i < num_sizes; i++) {
        int size = sizes[i];
        int make_set_count = 0;
```

```
        int find_set_set_count = 0;
        int union_count = 0;

        intial_set(size);

        clock_t start = clock();

        //For Randomly Taking Operations
        perform_operations(size, &make_set_count, &find_set_set_count, &union_count);
        clock_t end = clock();

        double time_taken = (double)(end - start) / CLOCKS_PER_SEC * 1000;

        printf("Size n=%d, Time Taken%fms, Make_Set Count%d, find_set_set Count%d,Union Count%d\n", size, time_taken, make_set_count, find_set_set_count, union_count);
    }

    return 0;
}
```

```
Size n=50, Time Taken0.051000ms, Make_Set Count13, find_set_set Count20,Union Count17
Size n=100, Time Taken0.021000ms, Make_Set Count26, find_set_set Count38,Union Count36
Size n=150, Time Taken0.027000ms, Make_Set Count44, find_set_set Count56,Union Count50
Size n=200, Time Taken0.035000ms, Make_Set Count57, find_set_set Count78,Union Count65
Size n=250, Time Taken0.042000ms, Make_Set Count73, find_set_set Count102,Union Count75
Size n=300, Time Taken0.050000ms, Make_Set Count88, find_set_set Count117,Union Count95
Size n=350, Time Taken0.057000ms, Make_Set Count105, find_set_set Count134,Union Count111
Size n=400, Time Taken0.064000ms, Make_Set Count121, find_set_set Count149,Union Count130
```

Time Complexity Plot