|  |  |  |
| --- | --- | --- |
| Logo  Description automatically generated | National University  of Computer and Emerging Sciences  Chiniot-Faisalabad Campus | Icon  Description automatically generated |

**Discrete Project**

**Semester Project**

**Fall 2021**

|  |  |
| --- | --- |
| **Maximum Marks:** | **Due Date:** 2 Jan 2021 |

**Submitted By**

|  |  |
| --- | --- |
| Name: | Muhammad Awais  Muhammad Ahmed |
| Student ID: | 20F-0166  20F-0282 |
| Section: | BSCS-3C |

**Submitted To**

Sir Tayyab Javed

**Submission Date**

2 Jan 2021

# QUESTION#1:

#include<iostream>

#include<string>

#define MAX 20

using namespace std;

char stk[20];

int top = -1;

void convert(int, int\*\* array, int colomn);

// Push function here, inserts value in stack and increments stack top by 1

void push(char oper)

{

if (top == MAX - 1)

{

cout << "FULL" << endl;

}

else

{

top++;

stk[top] = oper;

}

}

// Function to remove an item from stack. It decreases top by 1

char pop()

{

char ch;

if (top == -1)

{

cout << "Empty" << endl;

}

else

{

ch = stk[top];

stk[top] = '\0';

top--;

return(ch);

}

return 0;

}

int priority(char alpha)

{

if (alpha == '+' || alpha == '?')

{

return(1);

}

if (alpha == '\*' || alpha == '/')

{

return(1);

}

if (alpha == '!')

{

return(2);

}

return 0;

}

string In\_po(string infix)

{

int i = 0;

string postfix = "";

while (infix[i] != '\0')

{

if (infix[i] >= 'a' && infix[i] <= 'z' || infix[i] >= 'A' && infix[i] <= 'Z')

{

postfix.insert(postfix.end(), infix[i]);

i++;

}

else if (infix[i] == '(' || infix[i] == '{' || infix[i] == '[')

{

push(infix[i]);

i++;

}

else if (infix[i] == ')' || infix[i] == '}' || infix[i] == ']')

{

if (infix[i] == ')')

{

while (stk[top] != '(')

{

postfix.insert(postfix.end(), pop());

}

pop();

i++;

}

if (infix[i] == ']')

{

while (stk[top] != '[')

{

postfix.insert(postfix.end(), pop());

}

pop();

i++;

}

if (infix[i] == '}')

{

while (stk[top] != '{')

{

postfix.insert(postfix.end(), pop());

}

pop();

i++;

}

}

else

{

if (top == -1)

{

push(infix[i]);

i++;

}

else if (priority(infix[i]) <= priority(stk[top])) {

postfix.insert(postfix.end(), pop());

while (priority(stk[top]) == priority(infix[i])) {

postfix.insert(postfix.end(), pop());

if (top < 0) {

break;

}

}

push(infix[i]);

i++;

}

else if (priority(infix[i]) > priority(stk[top])) {

push(infix[i]);

i++;

}

}

}

while (top != -1)

{

postfix.insert(postfix.end(), pop());

}

// cout << "The converted postfix string is : " << postfix; //it will print postfix conversion

return postfix;

}

int main()

{

// int cont;

string infix, postfix;

cout << "and &, or ~ ,imp > ,bicond <>" << endl;

cout << "\nEnter the infix expression : "; //enter the expression

cin >> infix;

postfix = In\_po(infix);

int n = 0, count = 0;

char repeatedchar[10] = { '\0' };

cout << postfix << endl;

bool flag = false;

for (int i = 0; postfix[i] != '\0'; i++)

{

count++;

if ((postfix[i] > 64 && postfix[i] < 91) || (postfix[i] > 96 && postfix[i] < 123))

{

for (int j = 0; repeatedchar[j] != '\0'; j++)

{

if (repeatedchar[j] == postfix[i])

{

flag = true;

break;

}

else

flag = false;

}

if (flag == false)

{

for (int k = 0; k < 10; k++)

{

if (repeatedchar[k] == '\0')

{

repeatedchar[k] = postfix[i];

break;

}

}

}

else

continue;

}

}

for (int i = 0; repeatedchar[i] != '\0'; i++)

{

n++;

}

int rows = pow(2, n);

int\* result = new int[rows];

int\* proper = new int[rows];

int\*\* array = new int\* [rows];

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

{

array[i] = new int[n];

}

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

{

result[i] = 0;

proper[i] = 0;

for (int j = 0; j < n; j++)

{

array[i][j] = 0;

}

}

convert(rows - 1, array, n);

int counter = 0;

int\* a1 = new int[rows];

int\* a2 = new int[rows];

int num1 = -1, num2 = -1, num3 = -1;

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

{

if (postfix[i] > 96 && postfix[i] < 123)

{

for (int j = 0; j < n; j++)

{

if (postfix[i] == repeatedchar[j])

{

if (num1 == -1)

{

num1 = j;

break;

}

else if (num2 == -1)

{

num2 = j;

break;

}

else

{

num3 = j;

break;

}

}

}

}

else

{

if (postfix[i] == '!') // NOT

{

if (num1 != -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num2] == 0)

{

result[j] = 1;

}

else

{

result[j] = 0;

}

}

}

else if (num3 != -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num3] == 0)

{

result[j] = 1;

}

else if (array[j][num3] == 1)

{

result[j] = 0;

}

}

}

else if (num1 == -1 && num2 == -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == 0)

{

result[j] = 1;

}

else if (result[j] == 1)

{

result[j] = 0;

}

}

}

else if (num1 == -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num2] == 0)

{

result[j] = 1;

}

else if (array[j][num2] == 1)

{

result[j] = 0;

}

}

}

else if (num1 != -1 && num2 == -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num1] == 0)

{

result[j] = 1;

}

else if (array[j][num1] == 1)

{

result[j] = 0;

}

}

}

else if (num1 != -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num2] == 0)

{

result[j] = 1;

}

else if (array[j][num2] == 1)

{

result[j] = 0;

}

}

}

if (num3 != -1)

{

num3 = -1;

}

else if (num3 == -1 && num2 == -1 && num1 != -1)

{

num1 = -1;

}

else if (num3 == -1 && num2 != -1 && num1 != -1)

{

num2 = -1;

}

else if (num3 == -1 && num2 != -1 && num1 == -1)

{

num2 = -1;

}

}

if (postfix[i] == '~') //OR

{

if (num1 != -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = array[j][num1] || array[j][num2];

}

}

else

{

if (num1 != -1 && num2 != -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = array[j][num3] || array[j][num2];

}

}

else if (num1 == -1 && num2 != -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = result[j] || array[j][num2];

}

}

else if (num1 == -1 && num2 == -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = result[j] || array[j][num3];

}

}

else if (num2 == -1 && num3 == -1 && num1 != -1)

{

for (int j = 0; j < rows; j++)

{

cout << result[j] << endl;

}

cout << endl;

for (int j = 0; j < rows; j++)

{

result[j] = result[j] || array[j][num1];

}

}

}

if (num3 != -1)

{

num2 = -1, num3 = -1;

}

else

{

num1 = -1; num2 = -1;

}

}

else if (postfix[i] == '&') //AND

{

if (num1 != -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = array[j][num1] && array[j][num2];

}

}

else

{

if (num1 != -1 && num2 != -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = array[j][num3] && array[j][num2];

}

}

else if (num1 == -1 && num2 != -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = result[j] && array[j][num2];

}

}

else if (num1 == -1 && num2 == -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = result[j] && array[j][num3];

}

}

else if (num2 == -1 && num3 == -1 && num1 != -1)

{

for (int j = 0; j < rows; j++)

{

result[j] = result[j] && array[j][num1];

}

}

}

if (num3 != -1)

{

num2 = -1, num3 = -1;

}

else

{

num1 = -1; num2 = -1;

}

}

else if (postfix[i] == '>') // Implication

{

if (num1 != -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num1] == 0)

{

array[j][num1] = 1;

}

else if (array[j][num1] == 1)

{

array[j][num1] = 0;

}

}

for (int j = 0; j < rows; j++)

{

result[j] = array[j][num1] || array[j][num2];

}

}

else

{

if (num1 != -1 && num2 == -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == 0)

{

result[j] = 1;

}

else if (result[j] == 1)

{

result[j] = 0;

}

}

for (int j = 0; j < rows; j++)

{

result[j] = result[j] || array[j][num1];

}

}

if (num1 != -1 && num2 != -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num2] == 0)

{

array[j][num2] = 1;

}

else if (array[j][num2] == 1)

{

array[j][num2] = 0;

}

}

for (int j = 0; j < rows; j++)

{

result[j] = array[j][num3] || array[j][num2];

}

}

else if (num1 == -1 && num2 != -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == 0)

{

result[j] = 1;

}

else if (result[j] == 1)

{

result[j] = 0;

}

}

for (int j = 0; j < rows; j++)

{

result[j] = result[j] || array[j][num2];

}

}

else if (num1 == -1 && num2 == -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == 0)

{

result[j] = 1;

}

else if (result[j] == 1)

{

result[j] = 0;

}

}

for (int j = 0; j < rows; j++)

{

result[j] = result[j] || array[j][num3];

}

}

else if (num2 == -1 && num3 == -1 && num1 != -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == 0)

{

result[j] = 1;

}

else if (result[j] == 1)

{

result[j] = 0;

}

}

for (int j = 0; j < rows; j++)

{

result[j] = result[j] || array[j][num1];

}

}

}

if (num3 != -1)

{

num2 = -1, num3 = -1;

}

else

{

num1 = -1; num2 = -1;

}

}

else if (postfix[i] == '<') // By Condition

{

if (num1 != -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num1] == array[j][num2])

result[j] = 1;

else

result[j] = 0;

}

}

else

{

if (num1 != -1 && num2 == -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == array[j][num1])

{

result[j] = 1;

}

else

{

result[j] = 0;

}

}

}

else if (num1 != -1 && num2 != -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

if (array[j][num3] == array[j][num2])

result[j] = 1;

else

result[j] = 0;

}

}

else if (num1 == -1 && num2 != -1 && num3 == -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == array[j][num2])

result[j] = 1;

else

result[j] = 0;

}

}

else if (num1 == -1 && num2 == -1 && num3 != -1)

{

for (int j = 0; j < rows; j++)

{

if (result[j] == array[j][num3])

result[j] = 1;

else

result[j] = 0;

}

}

}

if (num3 != -1)

{

num2 = -1, num3 = -1;

}

else

{

num1 = -1; num2 = -1;

}

}

}

}

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

{

cout << result[i] << endl;

}

}

void convert(int n, int\*\* array, int colomn)

{

int no = n;

int a = n;

while (no >= 0)

{

n = no;

long long bin = 0;

int rem, i = 1;

while (n != 0 && a > -1) {

rem = n % 2;

n /= 2;

bin += rem \* i;

i \*= 10;

}

int div = 0;

int b = colomn - 1;

while (bin != 1 && bin != 0)

{

div = bin % 10;

bin = bin / 10;

array[a][b] = div;

b--;

if (bin == 0 || bin == 1)

{

array[a][b] = bin;

}

}

if (no == 1)

{

array[1][colomn - 1] = 1;

}

a--;

no--;

}

}

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

# QUESTION#2:

## Relations:

#include <iostream>

using namespace std;

void input(int\* array1, int size1, int\* array2, int size2, int\* array3, int size3)

{

cout << "Enter elements in set 1"<<endl;

for (int i = 0; i < size1; i++) {

cout << "tEnter element " << i + 1 << " of first set >> ";

int input;

cin >> input;

while (cin.fail())

{

cin.clear();

cin.ignore();

cout << "Invalid input\n";

cout << "Enter Again: ";

cin >> input;

array1[i] = input;

}

array1[i] = input;

}

system("cls");

cout << "Enter elements in set 2 "<<endl;

for (int i = 0; i < size2; i++) {

int input;

cin >> input;

while (cin.fail())

{

cin.clear();

cin.ignore();

cout << "Invalid input\n";

cout << "Enter Again: ";

cin >> input;

array2[i] = input;

}

array2[i] = input;

}

system("cls");

cout << "Enter elements in set 3 "<<endl;

for (int i = 0; i < size3; i++) {

int input;

cin >> input;

while (cin.fail())

{

cin.clear();

cin.ignore();

cout << "Invalid input\n";

cout << "Enter Again: ";

cin >> input;

array3[i] = input;

}

array3[i] = input;

}

system("cls");

}

int totalRelations(int size1, int size2) {

int x = pow(2, size1 \* size2);

return x;

}

int ReflexiveRelations(int size) {

int x = pow(2, size \* (size - 1));

return x;

}

int ReflexiveSymmetricRelations(int size) {

int x = pow(2, (size \* (size - 1)) / 2);

return x;

}

int AntiSymmetricRelations(int size) {

int x = pow(2, size) \* pow(3, (size \* (size - 1)) / 2);

return x;

}

int SymmetricAntisymmetricRelations(int size) {

int x = 2 \* size;

return x;

}

int SymmetricRelations(int size) {

int x = pow(2, (size \* (size + 1)) / 2);

return x;

}

void CartesianProduct(int\* arr1, int size1, int\* arr2, int size2) {

int count = 0;

cout << "Cartesian Product is : "<<endl;

cout << "{";

for (int i = 0; i < size1; i++)

{

for (int j = 0; j < size2; j++)

{

cout << "(" << arr1[i] << "," << arr2[j] << ")";

count++;

}

}

cout << "}" << endl;

cout << endl;

}

void setDisplay(int\* arr, int size) {

cout << "{ ";

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

cout << arr[i] << " , ";

}

cout << " }" << endl;

}

int SingleSetRelationCount(int size) {

int x = pow(2, size);

return x;

}

int main()

{

int\* arr1, \* arr2, \* arr3;

int size1, size2, size3;

cout << "Enter size of set 1 "<<endl;

cin >> size1;

while (size1<3)

{

cout << "size cannot be less than three " << endl;

cin >> size1;

}

cout << "Enter size of set 2 " << endl;

cin >> size2;

while (size2 < 3)

{

cout << "size cannot be less than three " << endl;

cin >> size2;

}

cout << "Enter size of set 3 " << endl;

cin >> size3;

while (size3 < 3)

{

cout << "size cannot be less than three " << endl;

cin >> size3;

}

system("cls");

arr1 = new int[size1];

arr2 = new int[size2];

arr3 = new int[size3];

input(arr1, size1, arr2, size2, arr3, size3);

cout << "SET 1 >> ";

setDisplay(arr1, size1);

cout << "SET 2 >> ";

setDisplay(arr2, size2);

cout << "SET 3 >> ";

setDisplay(arr3, size3);

cout << "<< SET1 & SET2 >> " << endl;

CartesianProduct(arr1, size1, arr2, size2);

cout << "<< SET2 & SET3 >> " << endl;

CartesianProduct(arr2, size2, arr3, size3);

cout << "<< SET1 & SET3 >> " << endl;

CartesianProduct(arr1, size1, arr3, size3);

cout << endl;

cout << "<< Total Relations between set1 and set2 >> " << totalRelations(size1, size2);

cout << "<< Total Relations between set2 and set3 >> " << totalRelations(size2, size3);

cout << "<< Total Relations between set1 and set3 >> " << totalRelations(size1, size3);

cout << endl;

cout << "<< Total Relations of set1 >> " << SingleSetRelationCount(size1);

cout << "<< Total Relations of set2 >> " << SingleSetRelationCount(size2);

cout << "<< Total Relations of set3 >> " << SingleSetRelationCount(size3);

int Sum = SingleSetRelationCount(size1) + SingleSetRelationCount(size2) + SingleSetRelationCount(size3);

cout << "<< Sum of all the relations is >> " << Sum;

cout << endl;

cout << "<< Total Reflexive Relations of set1 >> " << ReflexiveRelations(size1);

cout << "<< Total Reflexive Relations of set2 >> " << ReflexiveRelations(size2);

cout << "<< Total Reflexive Relations of set3 >> " << ReflexiveRelations(size3);

cout << endl;

cout << "<< Total Symmetric Relations of set1 >> " << SymmetricRelations(size1);

cout << "<< Total Symmetric Relations of set2 >> " << SymmetricRelations(size2);

cout << "<< Total Symmetric Relations of set3 >> " << SymmetricRelations(size3);

cout << endl;

cout << "<< Total Anti-Symmetric Relations of set1 >> " << AntiSymmetricRelations(size1);

cout << "<< Total Anti-Symmetric Relations of set2 >> " << AntiSymmetricRelations(size2);

cout << "<< Total Anti-Symmetric Relations of set3 >> " << AntiSymmetricRelations(size3);

cout << endl;

cout << "<< Both Reflexive and Symmetric Relations of set1 >> " << ReflexiveSymmetricRelations(size1);

cout << "<< Both Reflexive and Symmetric Relations of set2 >> " << ReflexiveSymmetricRelations(size2);

cout << "<< Both Reflexive and Symmetric Relations of set3 >> " << ReflexiveSymmetricRelations(size3);

cout << endl;

cout << "<< Both Anti-Symmetric and Symmetric Relations of set1 >> " << SymmetricAntisymmetricRelations(size1);

cout << "<< Both Anti-Symmetric and Symmetric Relations of set2 >> " << SymmetricAntisymmetricRelations(size2);

cout << "<< Both Anti-Symmetric and Symmetric Relations of set3 >> " << SymmetricAntisymmetricRelations(size3);

cout << endl;

}

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

## Euler Path (UNDIRECTED GRAPH):

#include<iostream>

using namespace std;

int edges = 0;

int Matrix[9][9];

void matrixdisp(int vertices)

{

cout << "Adjacency Matrix" << endl;

for (int i = 1; i <= vertices; i++)

{

cout << "\t";

for (int j = 1; j <= vertices; j++)

{

cout << Matrix[i][j] << " ";

}

cout << endl;

}

}

void AdjacencyList(int vertices)

{

char charact = 'A';

cout << "Adjacency List" << endl;

for (int i = 1; i <= vertices; i++)

{

cout << "\t" << charact << " => ";

for (int j = 1; j <= vertices; j++)

{

if (Matrix[i][j] != NULL)

if (j == 1)

cout << "A ";

else if (j == 2)

cout << "B ";

else if (j == 3)

cout << "C ";

else if (j == 4)

cout << "D ";

else if (j == 5)

cout << "E ";

else if (j == 6)

cout << "F ";

else if (j == 7)

cout << "G ";

else if (j == 8)

cout << "H ";

else if (j == 9)

cout << "I ";

}

cout << endl;

charact++;

}

}

void edge(int a, int b)

{

Matrix[a][b] = 1;

Matrix[b][a] = 1;

edges++;

}

void getdegree(int vertices)

{

int degree[9] = { 0 };

bool found = false;

int countOdd = 0;

char ch = 'A';

cout << "Adjacency List " << endl;

for (int i = 1; i <= vertices; i++)

{

cout << "\t" << ch << "--------->";

for (int j = 1; j <= vertices; j++)

{

if (Matrix[i][j] != NULL)

if (j == 1) {

cout << "A ";

degree[i - 1]++;

}

else if (j == 2) {

cout << "B ";

degree[i - 1]++;

}

else if (j == 3) {

cout << "C ";

degree[i - 1]++;

}

else if (j == 4) {

cout << "D ";

degree[i - 1]++;

}

else if (j == 5) {

cout << "E ";

degree[i - 1]++;

}

else if (j == 6) {

cout << "F ";

degree[i - 1]++;

}

else if (j == 7) {

cout << "G ";

degree[i - 1]++;

}

else if (j == 8) {

cout << "H ";

degree[i - 1]++;

}

else if (j == 9) {

cout << "I ";

degree[i - 1]++;

}

}

ch++;

cout << endl;

if (degree[i - 1] % 2 == 0) {

continue;

}

else {

cout << "Euler Circuit is not possible ";

}

}

int vertexWithOddCount[9] = { 0 };

//vertices with odd count

for (int i = 1; i <= 9; i++) {

if (degree[i - 1] % 2 == 0) {

continue;

}

else {

countOdd++;

vertexWithOddCount[i - 1] = i;

}

}

//odd degrees

if (countOdd <= 2) {

cout << "Euler Path Possible " << endl;

found = true;

}

else {

cout << "Euler Path not Possible " << endl;

}

//odd vertex found

if (found)

{

if (countOdd == 0)

{

char character = 'A';

cout << "Euler Circuit" << endl;

for (int i = 1; i <= 9; i++)

{

cout << " " << character << " -> ";

for (int j = 1; j <= 9; j++)

{

if (Matrix[i][j] != NULL) {

if (j == 1)

{

cout << "A ";

}

else if (j == 2)

{

cout << "B ";

}

else if (j == 3)

{

cout << "C ";

}

else if (j == 4)

{

cout << "D ";

}

else if (j == 5)

{

cout << "E ";

}

else if (j == 6)

{

cout << "F ";

}

else if (j == 7) {

cout << "G ";

}

else if (j == 8) {

cout << "H ";

}

else if (j == 9) {

cout << "I ";

}

Matrix[i][j] = NULL;

Matrix[j][i] = NULL;

}

break;

}

character++;

}

cout << "Euler Path" << endl;

character = 'A';

for (int i = 1; i <= 9; i++)

{

cout << " " << character << " -> ";

for (int j = 1; j <= 9; j++)

{

if (Matrix[i][j] != NULL) {

if (j == 1)

{

cout << "A ";

}

else if (j == 2)

{

cout << "B ";

}

else if (j == 3)

{

cout << "C ";

}

else if (j == 4)

{

cout << "D ";

}

else if (j == 5)

{

cout << "E ";

}

else if (j == 6)

{

cout << "F ";

}

else if (j == 7) {

cout << "G ";

}

else if (j == 8) {

cout << "H ";

}

else if (j == 9) {

cout << "I ";

}

Matrix[i][j] = NULL;

Matrix[j][i] = NULL;

}

break;

}

character++;

}

}

}

}

int main()

{

int vertices = 9;

edge(1, 2);

edge(1, 9);

edge(2, 3);

edge(3, 4);

edge(4, 5);

edge(5, 6);

edge(6, 4);

edge(7, 4);

edge(7, 3);

edge(7, 8);

edge(7, 6);

edge(8, 7);

edge(8, 2);

edge(8, 3);

edge(8, 9);

edge(9, 6);

edge(9, 1);

edge(9, 2);

edge(9, 8);

cout << "total number of edges are : " << edges << endl;

matrixdisp(vertices);

getdegree(vertices);

}

Text

Description automatically generated

## Report:

## Sets:

# Music Playlist:

Our Spotify or music playlist are the most widely and commonly used application of sets.

In playlist we collect a set of music according to our mood or taste.in that sequence we are

Making a collection of well-defined collections of objects which are distinct

# Library Portion:

Are another example of sets. Same type of books is available in same portion or rack.

In this way same set of books are available in a portion

# Universe:

In this whole word there are millions of galaxies in which there are different planet system exists

Which are separated form each other by some distance. Therefore, the universe act as a set which contains collection of same kind of planets, orbits and galaxies in different sets

## Relations:

# Family system:

In family system one family lives in one house and the house contains one family and one person has one passport, and the passport can only be used by one person it cannot be used by more than one person. One person has one identity number, and the identity number is also unique to a person and is different form every other individual

# Education:

In educational institutes, there exists many one-to-one relationships, For Example, one student can check out a book form library and the library book can be checked out by one student at a time and Each student must complete one work sheet and the worksheet can only be completed by one student at a time

# Marketing:

Many companies use one to one marketing to directly target potential customers and users they collect information about consumers and customers based on their previous purchases or interactions. Think about the last time you ordered something online and then clicked to another browser just to see an advertisement for the very same item, That’s one-to-one relation in marketing

## Graph:

# Google Maps:

Most of us use maps in our life almost daily to go to some place or to plan a route with the least traffic and stop signs and the best application used for this process is google maps and the google maps use graph for building routes and transportation system where intersections of two or more roads are considered to be a vertex and the road connecting two vertices is considered as an edge, thus their navigation system is based on the algo to calculate the shortest path between two vertex or nodes

# Facebook App:

Is an example of applications of graph in real life. It is a revolution in large scale data provision for example, a single user in Facebook can be represented as a node or vertex while their connections with others can be represented as an edge between the nodes

# Communication Tower:

Are the application of graphs. Consider the tower as node or vertex and the electrical wires

As edges. The planning of devising communication however is only possible by graph

# Airplane:

As we all know that there are no routes in the air so how airplanes know where on in which direction

They want to go. For that purpose, they use radars, the methodology behind the radars are graphs

Graphs determine their paths

## Minimal Spanning Tree:

# Network Design:

Minimal spanning tree methodology is used in network designing. Which helps in find the shortest path

Which contain the least traffic and noise.

# Face Tracking:

In real life face tracking. Like locating human face and objects in a video stream or a Live stream

Is done by minimal spanning tree algo

# Network Routing:

Routing is a process of selecting path along which the data can be transferred from source to the destination specified. Spanning tree are used in this data Structure. The computer network lines are connected to all the other cities (nodes or vertices). Therefore, spanning tree plays its role in network routing system