ACT 4.3

Programación de estructura de datos

Elías Uriel Velázquez Rojas
A01639716

Lo primero que hicimos en nuestro archivo principal fue incluir librerías y crear dos estructuras, las cuales nombramos src y Edge, la primera tiene dos valores enteros, que son src que es el valor, y repe que son las veces que este se repite en el archivo, en Edge es similar pero se agrega el atributo dest, que es para que el grafo pueda ser no dirigido que pueda ir y volver

```
struct src {
    int src, repe;
};

struct Edge {
    int src, dest, repe;
};
```

Después creamos la clase grafo con sus atributos en publico un vector, creado de un vector de tipo src llamado adjList, y luego se crea su constructor, con los atributos un vector de tipo Edge y uno entero, se usa un for hasta Edge, y se declaran ejes, y se usa el push back, para poder hacer el grafo no dirigido.

```
class Graph
{
public:
    vector<vector<src>> adjList;
    Graph(vector<Edge> const& edges, int N)
    {

        adjList.resize(N);

        for (auto& edge : edges)
        {
            src eje = { edge.src, edge.repe };
            src eje2 = { edge.dest, edge.repe };
            adjList[edge.src].push_back(eje2);
            adjList[edge.dest].push_back(eje);
        }
    }
};
```

Esta función se usa para regresar el primer valor de la ip que se encuentra en el txt, mediante el uso de la librería string

```
int ip2int(string ip) { //Regresa primer valor entero de IP
    int rst = 0;
    int a, b, c, d;
    char t = '.';
    stringstream ss(ip);
    ss >> a >> t >> b >> t >> c >> t >> d;
    rst = (a);
    return rst;
}
```

Esta función la usamos para crear el grafo, que recibe los contadores dividiendo las ip, a su vez que un grafo y un vector referenciados bool y de tipo src una variable v.

```
void BFS(Graph const& graph, src v, vector<bool>& discovered,int &cont1_99, int &cont100_199, int &cont200_299,int &cont300_399, int &cont400_499, int &cont500_599, int &cont
{
```

Se crea un queue de tipo src , se declara un discovered como true y en q se pushea el valor de v, después se hace un ciclo comprobando que q sea diferente a vacio, después se hace una comprobación para sumar al contador el valor de las repeticiones del v.

```
queue<src> q;

discovered[v.src] = true;
q.push(v);
while (!q.empty())
{
    v = q.front();
    q.pop();
    if (v.src >= 1 and v.src<100) {
        cont1_99 = cont1_99 + v.repe;
    }
    if (v.src >= 100 and v.src < 199) {
        cont100_199 = cont100_199 + v.repe;
    }
    if (v.src >= 200 and v.src < 299) {
        cont200_299 = cont200_299 + v.repe;
    }
    if (v.src >= 300 and v.src < 399) {
        cont300_399 = cont300_399 + v.repe;
    }
```

Luego se hace un ciclo con un for , que se usa para ver si el Edge u su valor de tipo src ya ha sido descubierto si es diferente a descubierto se declara como true, y después se pushea a q el valor de u.

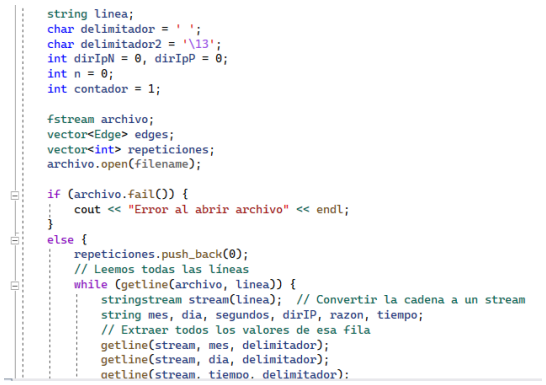
```
    if (v.src >= 500 and v.src < 599) {
        cont500_599 = cont500_599 + v.repe;
    }
    if (v.src >= 600 and v.src < 699) {
        cont600_699 = cont600_699 + v.repe;
    }
    if (v.src >=700 and v.src < 799) {
        cont700_799 = cont700_799 + v.repe;
    }
    if (v.src >= 800 and v.src < 899) {
        cont800_899 = cont800_899 + v.repe;
    }
    if (v.src >= 900 and v.src < 999) {
        cont900_999 = cont900_999 + v.repe;
    }

    // do for every edge (v -> u)
    for (src u : graph.adjList[v.src])
        if (!discovered[u.src])
        {
            discovered[u.src] = true;
            q.push(u);
        }
}
```

Esta función se usa para recibir los valores del txt que recibe los contadores referenciados, y un string .

```
void llenarArbol(string filename,int &cont1_99, int &cont100_199, int &cont200_299, int& cont300_399, int& cont400_499, int& cont500_599, int& cont600_699, int& cont700_799,
```

Dentro de la funcion se definen los delimitadores de tipo char, y las variables para guardar las direcciones de tipo entero, a su vez de los contadores, se cream dos vectores de tipo Edge y tipo entero, para las repeticiones y los edge, después se abre el archivo llamdo filename con un ciclo para obtener cada linea, dividiendo por los el txt, como mes,dia,tiempo,dirIp y razon.

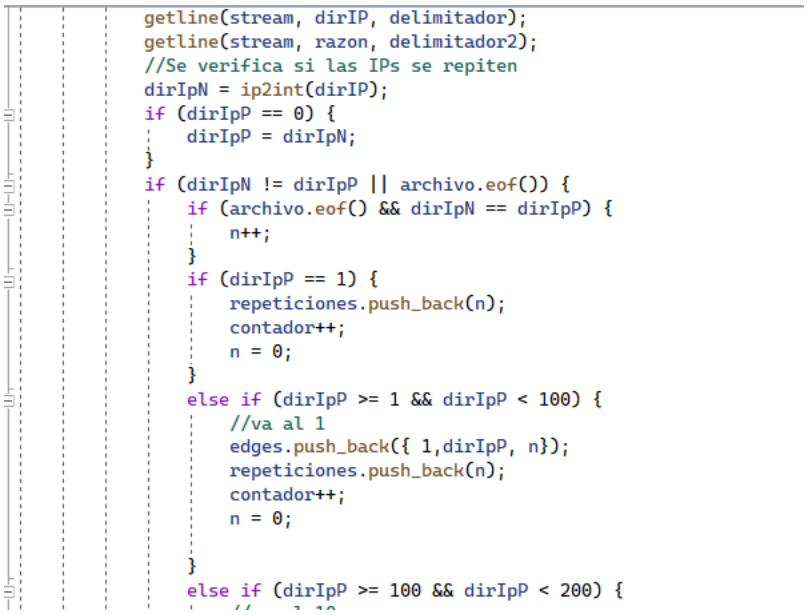


```
string linea;
char delimitador = ' ';
char delimitador2 = '\13';
int dirIpN = 0, dirIpP = 0;
int n = 0;
int contador = 1;

fstream archivo;
vector<Edge> edges;
vector<int> repeticiones;
archivo.open(filename);

if (archivo.fail()) {
    cout << "Error al abrir archivo" << endl;
}
else {
    repeticiones.push_back(0);
    // Leemos todas las lineas
    while (getline(archivo, linea)) {
        stringstream stream(linea);  // Convertir la cadena a un stream
        string mes, dia, segundos, dirIP, razon, tiempo;
        // Extraer todos los valores de esa fila
        getline(stream, mes, delimitador);
        getline(stream, dia, delimitador);
        getline(stream, tiempo, delimitador);
```

Después se verifica si las Ip se repiten luego se verifica las ip mediante la dirección de ip , si se verifica la dirección se va sumando el contador y la variable n, y se hace un push back, delimitando la dirección de la ip, y las repeticiones con el valor de n, que se va sumando dependiendo la verificación.



```
getline(stream, dirIP, delimitador);
getline(stream, razon, delimitador2);
//Se verifica si las IPs se repiten
dirIpN = ip2int(dirIP);
if (dirIpP == 0) {
    dirIpP = dirIpN;
}
if (dirIpN != dirIpP || archivo.eof()) {
    if (archivo.eof() && dirIpN == dirIpP) {
        n++;
    }
    if (dirIpP == 1) {
        repeticiones.push_back(n);
        contador++;
        n = 0;
    }
    else if (dirIpP >= 1 && dirIpP < 100) {
        //va al 1
        edges.push_back({ 1,dirIpP, n});
        repeticiones.push_back(n);
        contador++;
        n = 0;

    }
    else if (dirIpP >= 100 && dirIpP < 200) {
```

Al final se suma a n++ y se le asigna a la direccion de IP con terminacion p la direccion IP con terminacion N y se cierra el ciclo, después se hace un pushbak de la direccion de memoria junto con las repticiones, y se declara n como 0 y se cierra el else.

```
            }
            else if (dirIpP >= 900 && dirIpP < 1000) {
                //va al 90
                edges.push_back({ 90,dirIpP, n});
                repeticiones.push_back(n);
                contador++;
                n = 0;

            }
        }
        n++;
        dirIpP = dirIpN;
    }
    edges.push_back({ 90,dirIpP, n });
    repeticiones.push_back(n);
    n = 0;
}
```

Luego se crea y se asigna la variable V de tipo int, y se asigna el valor de contador + 1, después se crea un grafo, con los valores Edges, y la variable V, y un vector bool, llamado discoveredB, y se crea un ciclo hasta la variable V, y se hace un comprobación se verifica si discoverdB si es falso, si esto es verdadero se declara al eje, con los valores i y las repeticiones , después se llama recursivamente, con los valores de graph, eje , discovered y todos los contadores.

Después se cierra el archivo.

```
    int V = contador + 1;
    Graph graph(edges, V);
    vector<bool> discoveredB(V, false);
    src eje;
    for (int i = 0; i < V; i++) {
        if (discoveredB[i] == false) {
            eje = { i, repeticiones[i] };
            BFS(graph, eje, discoveredB, cont1_99, cont100_199,
        }
    }
    archivo.close();

}
```

En el main se declara un string, con el archivo txt, y todos los contadores, después se llama a la función arbol con todas las variables antes declaradas.

```
int main()
{
    string filename;
    filename = "archivo nuevo.txt";
    int cont1_99 = 0;
    int cont100_199 = 0;
    int cont200_299 = 0;
    int cont300_399 = 0;
    int cont400_499 = 0;
    int cont500_599 = 0;
    int cont600_699 = 0;
    int cont700_799 = 0;
    int cont800_899 = 0;
    int cont900_999 = 0;
    llenarArbol(filename, cont1_99, cont100_199, cont200_299,
```

Luego se crea un arreglo con los valores de los contadores, y se declara un max de valor 0, y se hace un ciclo para saber el valor máximo de los contadores, después se imprime esto y se hace una comprobación que devuelve donde se encuentra el mayor fan out.

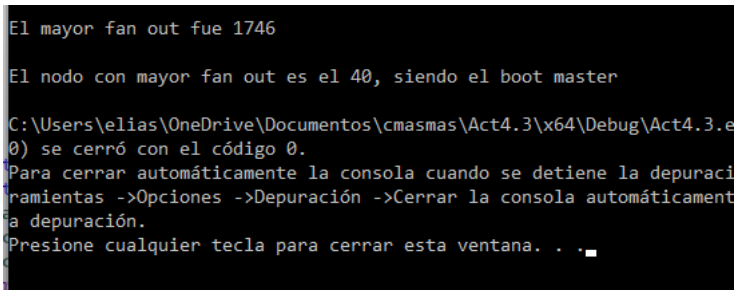
```
int max = 0;
int arreglo[10] = { cont1_99,cont100_199,cont200_299 ,cont300_399 ,cont400_499 ,cor
for (int i = 0; i < 10; i++)
{
    if (max < arreglo[i]) {
        max = arreglo[i];
    }
}
cout << "El mayor fan out fue "<< max << endl;
cout << endl;

if (max == cont1_99) {
    cout << "El nodo con mayor fan out es el 1, siendo el boot master" << endl;
}
if (max == cont100_199) {
    cout << "El nodo con mayor fan out es el 10, siendo el boot master " << endl;
}
if (max == cont200_299) {
    cout << "El nodo con mayor fan out es el 20, siendo el boot master " << endl;
}
if (max == cont300_399) {
    cout << "El nodo con mayor fan out es el 30, siendo el boot master " << endl;
}

if (max == cont400_499) {
    cout << "El nodo con mayor fan out es el 40, siendo el boot master " << endl;
}
if (max == cont500_599) {
    cout << "El nodo con mayor fan out es el 50, siendo el boot master " << endl;
}
if (max == cont600_699) {
    cout << "El nodo con mayor fan out es el 60, siendo el boot master " << endl;
}
if (max == cont700_799) {
    cout << "El nodo con mayor fan out es el 70, siendo el boot master " << endl;
}
if (max == cont800_899) {
    cout << "El nodo con mayor fan out es el 80, siendo el boot master " << endl;
}
if (max == cont900_999) {
    cout << "El nodo con mayor fan out es el 90, siendo el boot master " << endl;
}

return 0;
}
```

CASO PRUEBA:


El mayor fan out fue 1746
El nodo con mayor fan out es el 40, siendo el boot master
C:\Users\elias\OneDrive\Documentos\cmasmas\Act4.3\x64\Debug\Act4.3.e
0) se cerró con el código 0.
Para cerrar automáticamente la consola cuando se detiene la depuraci
ramientas ->Opciones ->Depuración ->Cerrar la consola automáticament
a depuración.
Presione cualquier tecla para cerrar esta ventana. . .