Tecnológico de Monterrey

Alumno: Roberto Eduardo Cardona Luis

Matricula:A00833365

Para realizar la entrega de este trabajo nosotros utilizamos los nodos en C++ para así nosotros crear un grafo.

Estos grafos nos son de gran utilidad en esta situación problema, ya que estos nos permiten almacenar distintos tipos de elementos y/o datos que nosotros podemos utilizar en algún otro momento para procesar los datos específicos que hay en esta. Los nodos pertenecientes a un grafo pueden interrelacionarse con otros permitiéndonos así analizarlos de una mejor manera. Al igual los grados al poder tener altos números de datos en ellos nos facilitan el procesar la información mediante buscadores o gestores en el programa realizado, también nos ayudan en el proyecto debido a que nos permiten entender cual es la relación que existe entre algunos datos.

Estos grafos nos ayudan a representar de mejor manera los datos que hay en una base de datos ya que estos datos los podemos nosotros poner en nodos siendo estos los vectores/aristas que representan las relaciones o lazos que existen entre los actores.

Complejidad

Definir las funciones de nodeIp

```
class NodeIP {
  private:
    int nVecinos, index;
    string IPstring;

  public:
    NodeIP();
    NodeIP(string,int);
    void addNeighbor(int);
    int getnVecinos();
    string getIP();
    vector<int> neighbors;
};

  NodeIP::NodeIP(){
    nVecinos=0;
    IPstring="";
    neighbors={0};
    index=0;
  }
```

Definir las funciones de graph

```
class Graph {
private:
  vector<NodeIP> v;
  int size;
  void swap(int,int);
  void Graphify(int);
  void GraphifyD(int,int);
public:
  Graph();
  void insertNode(NodeIP);
  void deleteRoot();
  NodeIP top();
  bool empty();
  int getSize();
  string print(int, int);
  string printBootMaster();
  void GraphSort();
};
```

Insertar los nodos

```
void Graph::insertNode(NodeIP key) {
  size ++;
  v.push_back(key);
  Graphify(size - 1);
}
```

Analizar el tamaño de las conexiones

```
void Graph::GraphifyD(int n, int i) {
  int small = i;
  int l = 2 * i + 1;
  int r = 2 * i + 2;
  if (l < n && v[l].getnVecinos() < v[small].getnVecinos())
      small = l;
  if (r < n && v[r].getnVecinos() < v[small].getnVecinos())
      small = r;
  if (small != i) {
      swap(i,small);
      GraphifyD(n,small);
  }
}
```

Eliminar filas

```
void Graph::deleteRoot() {
  NodeIP lastElement = v[size - 1];
  v[0] = lastElement;
  v.pop_back();
  size = size - 1;
  GraphifyD(size,0);
}
```

Obtener el tamaño del grafo

```
int Graph::getSize() {
  return size;
}
```

Github

https://github.com/rob437ertoe/TC1031-Portafolio_Final.git