Actividad Integral de grafos (Evidencia Competencia)

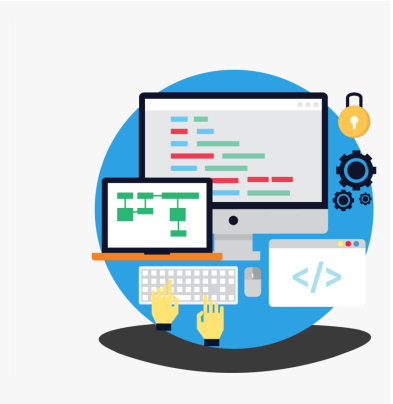
Tecnológico de Monterrey

Miguel Jiménez Padilla - A01423189

Profesora:

Mónica Larre Bolaños

18 de Noviembre de 2021



Reflexión:

En este bloque de la clase de estructuras de datos, pudimos aprender una forma sencilla pero eficiente de construir un grafo a manera de lista ligada, almacenando los arcos o conexiones entre nodos. Además, aprendimos la diferencia entre un grafo dirigido y un grafo normal, el cuál, por default tiene que tener dos direcciones entre dos nodos. Los grafos son muy importantes porque nos ayudan a entender cómo funcionan los recorridos entre distintos puntos, en este caso tomando como ejemplo los nodos.

En la actividad Integradora realizada, trabajamos con una aplicación real de los grafos en la que pudimos ver cuantos puertos podría recorrer un barco utilizando un contador. Asimismo, los grafos sirven para calcular rutas óptimas de viaje o para cualquier situación en dónde se deba calcular una o más rutas de movimiento.

Casos de prueba:

Caso de validación de puertos máximos

```
┌─[deckofminds@DESKTOP-FQ2ECQJ] - [/mnt/c/Users/mike-/Desktop/ActIntegralGRAFOS] - [Thu Nov 18, 22:04]
└─[$]> ./main.exe
Enter the number of ports in the network: 31

The number of ports must be greater than 0 and lower than 30, try again
Enter the number of ports in the network: 30
```

Caso de construcción del grafo, sin repetidos y en ambas direcciones

```
port: a
port: b
port: c
port: d

Enter the number of connections (Bows): 4

Enter the conections. Example: (initialPort secondPort)
Bow: a b
Bow: a c
Bow: a a
Bow: a d

____ GRAPH ____
[A] --->  [B C D ]
[B] --->  [A ]
[C] --->  [A ]
[D] --->  [A ]
```

Validación de veces para ingresar MNP

```
_________________________________________

Enter the number of cases you want to try for the MNPs: 2

_________________________________________
```

Caso dónde MNP = 2, en el grafo se recorren todos los nodos sin problema

```
____ GRAPH ____
[A] --->  [B C D ]
[B] --->  [A ]
[C] --->  [A ]
[D] --->  [A ]


_____________________________________________

Enter the number of cases you want to try for the MNPs: 2


_____________________________________________
_____________________________________________

Enter the Maximum Number of Ports (MNP): 2
Enter the Initial port: A

Case 1:  0 ports not reachable from port A with MNP = 2.
_____________________________________________
```

Caso dónde MNP = 1, iniciando desde A, recorre todos los nodos sin problema

```
_____________________________________________

Enter the Maximum Number of Ports (MNP): 1
Enter the Initial port: A

Case 2:  0 ports not reachable from port A with MNP = 1.
_____________________________________________
```

Casos de ejemplo:

```
Enter the number of ports in the network: 13

port: alexandria
port: algeciras
port: ambarli
port: antwerp
port: balboa
port: bandar
port: barcelona
port: bremen
port: busan
port: cai_mep
port: callao
port: cartagena
port: charleston

Enter the number of connections (Bows): 16

Enter the conections. Example: (initialPort secondPort)
Bow: alexandria algeciras
Bow: algeciras ambarli
Bow: ambarli antwerp
Bow: alexandria balboa
Bow: balboa bandar
Bow: bandar barcelona
Bow: antwerp bremen
Bow: bremen busan
Bow: algeciras cai_mep
Bow: cai_mep callao
Bow: ambarli cartagena
Bow: barcelona callao
Bow: cai_mep cartagena
Bow: callao charleston
Bow: cartagena charleston
Bow: charleston busan
```

```
____ GRAPH ____
[ALEXANDRIA] --->  [ALGECIRAS BALBOA ]
[ALGECIRAS] --->  [ALEXANDRIA AMBARLI CAI_MEP ]
[AMBARLI] --->  [ALGECIRAS ANTWERP CARTAGENA ]
[ANTWERP] --->  [AMBARLI BREMEN ]
[BALBOA] --->  [ALEXANDRIA BANDAR ]
[BANDAR] --->  [BALBOA BARCELONA ]
[BARCELONA] --->  [BANDAR CALLAO ]
[BREMEN] --->  [ANTWERP BUSAN ]
[BUSAN] --->  [BREMEN CHARLESTON ]
[CAI_MEP] --->  [ALGECIRAS CALLAO CARTAGENA ]
[CALLAO] --->  [CAI_MEP BARCELONA CHARLESTON ]
[CARTAGENA] --->  [AMBARLI CAI_MEP CHARLESTON ]
[CHARLESTON] --->  [CALLAO CARTAGENA BUSAN ]


_________________________________________

Enter the number of cases you want to try for the MNPs: 2


_________________________________________
_________________________________________

Enter the Maximum Number of Ports (MNP): 2
Enter the Initial port: cai_mep

Case 1:  5 ports not reachable from port CAI_MEP with MNP = 2.
_________________________________________
```

```
_________________________________________

Enter the Maximum Number of Ports (MNP): 3
Enter the Initial port: cai_mep

Case 2:  1 ports not reachable from port CAI_MEP with MNP = 3.
_________________________________________
```

Complejidades:

Complejidad->O(n^2)

```
void graph(Grafo &graf){

  int portsNo=0;
  while(true){
    cout<<"Enter the number of ports in the network: "; cin>>portsNo; cout<<endl;
    if(portsNo<=30 && portsNo>=0){break;}
    else{cout<<"The number of ports must be greater than 0 and lower than 30, try again "<<endl;}
  }

  string port=""; int nodesCounter=0;
  while (nodesCounter<portsNo){
    cout<< "port: "; cin>>port;

    for_each(port.begin(),port.end(),[](char &c){//cambia los arcos a mayúsculas
      c=::std::toupper(c);
    });

    graf.insertaOrden(port);

    nodesCounter++;
    port="";
  }
  cout<<endl;

  string bowsNo="";
  int bowsN=0;
```

```
while(true){
  cout<<"Enter the number of connections (Bows): "; cin>>bowsNo;
  bowsN=stoi(bowsNo);
  if(bowsN<=30 && bowsN>=0){break;}
  else{cout<<"the number of connections must be less than 30 and more than 0, try again
}


cout<<"\nEnter the conections. Example: (initialPort secondPort)"<<endl;
string portA="",portB="";
vector<string> ports;

for(int i=0; i<bowsN;i++){
  cout<<"Bow: ";cin>>portA>>portB;

  for_each(portA.begin(),portA.end(),[](char &c){//cambia los arcos a mayúsculas
    c=::std::toupper(c);
  });
  for_each(portB.begin(),portB.end(),[](char &c){//cambia los arcos a mayúsculas
    c=::std::toupper(c);
  });

  ports.push_back(portA);
  ports.push_back(portB);

  ports.push_back(portB);
  ports.push_back(portA);

  portA=""; portB="";
}
cout<<endl;

graf.insertaArcos(ports);
////////////////////////////////////////////////////////////////////////////////
```

Complejidad->O(n^2)

```
void mnpCases(Grafo& graf, int& casesCounter){
string mnpFun="";
  cout<<"_____"<<endl;
  cout<<"\nEnter the Maximum Number of Ports (MNP): "; cin>>mnpFun;

  int mnpFunInt=0;
  mnpFunInt=stoi(mnpFun);

  string initialNode="";
  vector<bool> visit={};
  int nonVisited=0;

  while(true){
    cout<<"Enter the Initial port: ", cin>>initialNode; cout<<endl;

    for_each(initialNode.begin(),initialNode.end(),[](char &c){//cambia los arcos a mayúsculas
      c=::std::toupper(c);
    });

    if(graf.nodeExists(initialNode)){
      //dfsAdapted;

      for(int h=0; h<graf.graphLength();h++){
        visit.push_back(false);
      }
      graf.dfsAdapted(graf.getNodeFromIndex(initialNode),visit,mnpFunInt);

      break;
```

```
    else{
      cout<<"That port doesn't exist, try again"<<endl;
    }
  }
    for( int x=0;x<visit.size();x++){
      if(visit[x]==false){
        nonVisited++;
      }
    }
}

cout<<"Case "<< casesCounter+1<<":  "<<nonVisited<<" ports not reachable from port "<< initialNode <<" with MNP = "<<mnpFunInt<<"."<<endl;
casesCounter++;
cout<<"_____"<<endl;
cout<<endl;

//graf.insertaArcos(ports);
//cout<<"Insercion hecha"<<endl;
```

Complejidad->O(n)

```
void Grafo::dfsAdapted(Nodo* vis, vector<bool>&visited,int MNP){
  visited[searchNode(vis->getDato())] = true;

  if(MNP>0){
   for(int g=0;g<vis->getLengthStrings();g++){

      dfsAdapted(getNodeFromIndex(vis->getBowWithIndex(g)),visited, MNP-1);
    }
  }
}
```

Complejidad->O(n^2)

```
void Grafo::insertaArcos(vector<string> arcos){
  Nodo* aux=head;

  while(aux!=NULL){// se recorre el grafo
    for(int x=0;x<arcos.size();x+=2){//ciclo para agregar strings al vector de arcos de un nodo
      string s="",s1="";

      s=arcos[x];//even
      s1=arcos[x+1];//odd

      if (s==aux->getDato() && isValid(aux,s1) && s1!=s){//valida que no se repitan las letras
        aux->addToVector(s1);//se agrega la letra de la derecha
        aux->setConections(aux->getLengthStrings()+1);//aumenta el atributo conexions (largo del vector del nodo)
      }
      //cout<<"here?"<<endl;


    }


    aux=aux->getSig();
  }
}
```

Complejidad->O(n)

```
void Grafo::insertaOrden(string n){//inserta un nodo en orden en el grafo
  Nodo* nuevo = new Nodo(n) ;
  if (head==NULL){//caso de que sea un grafo vacio
    head=nuevo;
  }
  else{
    Nodo* antes=head;
    Nodo* aux=head;
    while(aux!=NULL && aux->getDato()<n){//coloca el apuntador
      antes=aux;
      aux=aux->getSig();
    }
    if (antes==aux){ //caso insertar al inicio
      nuevo->setSig(head);
      head=nuevo;
    }
    else if (aux==NULL){//caso insertar al final
      antes->setSig(nuevo);
    }
    else{//se inserta entre dos
      antes->setSig(nuevo);
      nuevo->setSig(aux);
    }
  }
}
```