Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario de Algoritmia

REPORTE DE PRÁCTICA

IDENTIFICACIÓN DE LA PRÁCTICA

Práctica	**9**	**Nombre de la práctica**	**Algoritmo de Prim**
Fecha	**05/11/2021**	**Nombre del profesor**	**Alma Nayeli Rodríguez Vázquez**
Nombre de los integrantes del equipo		**Cárdenas Pérez Calvin Cristopher**	
		Farfán de León José Osvaldo	
		García Martínez Noe Aaron	

OBJETIVO

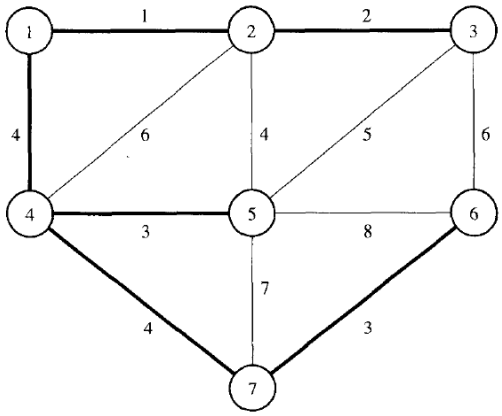
El objetivo de esta práctica consiste en implementar el algoritmo de Prim para encontrar el árbol de recubrimiento mínimo de un grafo.

PROCEDIMIENTO

Realiza la implementación siguiendo estas instrucciones.

Implementa el algoritmo de Prim utilizando Matlab y C++ / Python. Para la implementación, apóyate del siguiente algoritmo y del ejemplo:

function $Prim(G = \langle N, A\rangle$: *graph*; *length*: $A \to \mathbb{R}^+$): *set of edges*
    {initialization}
    $T \leftarrow \varnothing$
    $B \leftarrow$ {an arbitrary member of N}
    while $B \neq N$ **do**
        find $e = \{u, v\}$ of minimum *length* such that
            $u \in B$ and $v \in N \setminus B$
        $T \leftarrow T \cup \{e\}$
        $B \leftarrow B \cup \{v\}$
    return T



Step	$\{u, v\}$	B
Initialization	—	{1}
1	{1,2}	{1,2}
2	{2,3}	{1,2,3}
3	{1,4}	{1,2,3,4}
4	{4,5}	{1,2,3,4,5}
5	{4,7}	{1,2,3,4,5,7}
6	{7,6}	{1,2,3,4,5,6,7}

IMPLEMENTACIÓN

Agrega el código de tu implementación aquí.

```
no_nodos=7;
no_conjuntos=no_nodo
s;grafo= [1 2 1;
          1 4 4;
          2 3 2;
          2 4 6;
          2 5 4;
          3 5 5;
          3 6 6;
          4 5 3;
          4 7 4;
          5 6 8;
          5 7 7;
          6 7 3];
  pesos=grafo(:,3);
  aristas=grafo(:,1:2);
  [pesos,ind]=sort(pesos);
  aristas=aristas(ind,:);
  no_aristas=size(aristas,1
  );T=[];
  conjuntos={};
  for
  i=1:no_nodos
     conjuntos(i)=
  i;end

  for i=1:no_aristas
     arista_i=aristas(i,:
     );peso i=pesos;
     for j=1:no_conjuntos
          if
               ismember(arista_i(1),conjuntos
               (j))conjunto1=j;
               break;
end
```

```
      e
      n
      d
      e
      n
      d
   for j=1:no_conjuntos
      if
           ismember(arista_i(2),conjuntos
           (j))conjunto2=j;
           end
      end
   if conjunto1~=conjunto2
       conjuntos{conjunto1}=[conjuntos{conjunto1};conjuntos{conjunto2
       }]conjuntos{conjunto2}=[];
       T=[T; arista_i
      peso_i];end

 sizeT=size(T,1);
 if
      sizeT==no_nod
      os-1break;
      end
 e
 n
 d
 e
 n
 d

 T
pesoTotal=sum(T(:,3))
```

Código de Matlab

```
#include
<iostream>
#include <vector>
#include <queue>
#include <utility>
#include
<functional>
#include <unordered_set> //requerido para la cola
#define p pair < int, int >
```

```
#define pa pair < int, p
>#define infinite 32767
#define nil 32767

using namespace
std;bool debug;

unordered_set <int>
Q;int N, E;
vector<int> pie, key, V;

priority_queue < pa, vector<pa>, greater<pa> > minE;

/*
 * El gráfico y los costos / pesos de los bordes se mantienen en matrices 2D
*/



  vector < vector <int> > w, G;

  void init(){
          w.resize(N);
          G.resize(N);
          for(int i=0; i<N; i++){
                  V.push_ba
                  ck(i);
                  w[i].resize(
                  N);
          }
          pie.resiz
          e(N);
          key.resiz
          e(N);
  }

  void addCola(int x){
          for(int y : G[x]){
                  minE.push(pa(w[x][y], p(x,
                  y)));if(debug == true){
                          cout<<"\t added "<<y+1<<" from"<<x+1;
                  }
          }
  }

  void MST_PRIM(int
          r=0){for(auto
          u : V){
```

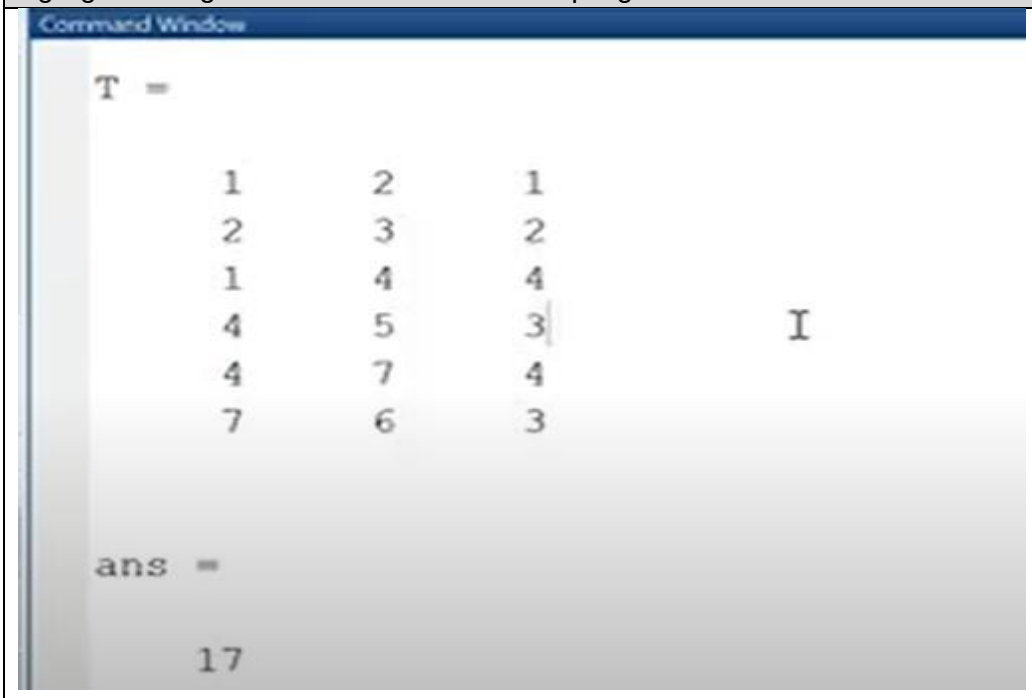
```
                key[u]=infi
                nite;
                pie[u]=nil;
        }

        key[r]=0; addCola(r);

        while(minE.empty()!=t

        rue){

                if(debug==true) cout<<"\tBucle interior\n";

                int u=minE.top().second.first;
                int
                v=minE.top().second.secon
                d;int c=minE.top().first;
                minE.pop();

                if(c<key[v]){
                        pie[v]=u;
                        key[v]=
                        c;
                        addCol
                        a(v);
                }
        }
 }
int main(){
        debug=false;
        cout<<"Algortimo de prim.\n Ingrese el numero de vertices y aristas:
        " ;cin>>N>>E;
        cout<<"\nIngrese todos los bordes en el grafico en formato - v1 v2 borde-costo : \n
[apartir de 1, a la raiz]";
        int tmp1, tmp2,
        tmp3;init();
        for(int i=0; i<E; i++){
                cin>>tmp1>>tmp2>>tmp3;
                G[tmp1-1].push_back(tmp2-
                1);G[tmp2-
                1].push_back(tmp1-1);
                w[tmp1-1][tmp2-1]=tmp3;
                w[tmp2-1][tmp1-1]=tmp3;
        }
        int cost=0;
        pie[0]=0;
        MST_PRIM
        ();
```

```
        cout<<" ::MST::\La raiz es :
        1";for(int i=1; i<N; i++){
                cout<<"( "<<i+1<<", "<<pie[i]+1<<" ) :
                "<<key[i]<<"\n";cost+=key[i];
        }
        cout<<"Costo minimo =
        "<<cost;return 0;
}
```

Código en C++/Python

RESULTADOS

Agrega la imagen de la consola con el despliegue de los resultados obtenidos.



Resultados Matlab (La matriz “T” y la suma de los pesos)

```
Algortimo de prim.
 Ingrese el numero de vertices y aristas: 3
4

Ingrese todos los bordes en el grafico en formato - v1 v2 borde-costo :
 [a partir de 1, a la raiz]1
2
3
4
4
3
```

Resultados C++/Python (La matriz “T” y la suma de los pesos)

CONCLUSIONES

Escribe tus observaciones y conclusiones.
Es un buen algoritmo junto al de Kruskal para encontrar el camino mínimo nada más que en este algoritmo puedes empezar del vértice que gustes y de ahí empiezas a tomar las aristas con menor peso hasta conectar todo, no es tan difícil de entender. La actividad no se complico tanto, realmente me gusto hacerla porque es entretenida.