4.3. Código de Programa del Algoritmo de Jarnick

```
#include<stdio.h>
#include<stdlib.h>

void main(){
FILE* fichero;
int i,j,k;
int n,opcion;
int respaldo1,respaldo2;
int menor;
int *V;
int **adya,**peso;
//-----------------------------------------------
do{
//MENU PARA ELEGIR GRAFO A EVALUAR:
do{
printf("\n\nElige el grafo a evaluar:\n1. Grafo1\n2. Grafo2\n3.
Grafo3\n4. Grafo4\n5. Grafo5\n6. No elegir y Salir\n\n");
scanf("%d",&opcion);
switch(opcion){
case 1  :fichero = fopen("Grafo1.txt", "rt");break;
case 2  :fichero = fopen("Grafo2.txt", "rt");break;
case 3  :fichero = fopen("Grafo3.txt", "rt");break;
case 4  :fichero = fopen("Grafo4.txt", "rt");break;
case 5  :fichero = fopen("Grafo5.txt", "rt");break;
case 6  :printf("ADIOS\n");break;
default :printf("Opcion incorrecta! VUELVA A INTENTARLO\n\n");break;
}
}while(opcion<1||opcion>6);
//RESERVA DE MEMORIA:
if(opcion!=6){
fscanf(fichero,"%d",&n);

adya = (int **) malloc (n * sizeof(int *));
for (k=0 ; k < n ; k++) {
   adya[k] = (int *) malloc (n * sizeof(int));
}

peso = (int **) malloc (n * sizeof(int *));
for (k=0 ; k < n ; k++) {
   peso[k] = (int *) malloc (n * sizeof(int));
}

V = (int *) malloc (n * sizeof(int));

//-------------------------------------------------
// INICIALIZAVION DEL CONJUNTO V:

V[0]=1;
for (k=1 ; k < n ; k++) V[k]=0;

//-------------------------------------------------
// LECTURA DE LAS MATRICES:
for(i=0;i<n;i++)
for(j=0;j<n;j++)fscanf (fichero, "%d",&adya[i][j]);
for(i=0;i<n;i++)
for(j=0;j<n;j++)fscanf (fichero, "%d",&peso[i][j]);
```

```
//-----------------------------------------------------

// ALGORITMO:
printf("\n\nResultado del Algoritmo:\n");
printf
("-----------------------------------------------------------------
\n\n");
printf("E0={}\n\nV0={1}\n\n");
for(k=0;k<n-1;k++){
                menor=10000;
                for(i=0;i<n;i++){
                   if(V[i]==1){
                     for(j=0;j<n;j++){
                       if (peso[i][j]< menor&&adya[i][j]==1&&V[i]!=V[j]){
                                       menor=peso[i][j];
                                              respaldo1=i;
                                              respaldo2=j;
                         }
                      }

                    }

                 }
adya[respaldo1][respaldo2]=adya[respaldo2][respaldo1]=2;
V[respaldo2]=1;

// MOSTRAMOS EL CONJUNTO DE ARISTAS Y DE VERTICES SELECCIONADOS
// HASTA LA K-ESIMA ITERACION DEL ALGORITMO:
printf
("-----------------------------------------------------------------
\n\n");
printf("E%d={",k+1);
for(i=0;i<n;i++){
  for(j=i;j<n;j++) if(adya[i][j]==2) printf("{%d,%d},",i+1,j+1);
}
printf("\b}\n\nV%d={",k+1);
for(i=0;i<n;i++)  if(V[i]==1)  printf("%d,",i+1);
  printf("\b}\n\n");
}
printf("\n");
// FIN DE ALGORITMO.
//-----------------------------------------------------------
}
printf
("-----------------------------------------------------------------
\n\n");
}while(opcion!=6);
} // FIN DE PROGRAMA.
```

4.4. Código de Programa del Algoritmo de Boruvka

```
#include<stdio.h>
#include<stdlib.h>

void main(){
//-------------------------------------------------------------------------------

//VARIABLES A USAR:
FILE* fichero;
int i,j,k,l,t,contador;
int n,bandera,opcion;
int x,y;
int menor;
char *V,*Va;   // Los usaremos como vectores de etiquetas.
int *registro; // registará que vertices han sido
pintados.
int **adya,**peso,**Arbol;

//-------------------------------------------------------------------------------
do{
//MENU PARA ELEGIR GRAFO A EVALUAR:

do{
printf("\n\nElige el grafo a evaluar:\n1. Grafo1\n2. Grafo2\n3.
Grafo3\n4. Grafo4\n5. Grafo5\n6. No elegir y Salir\n\n");
scanf("%d",&opcion);
switch(opcion){
case 1  :fichero = fopen("Grafo1.txt", "rt");break;
case 2  :fichero = fopen("Grafo2.txt", "rt");break;
case 3  :fichero = fopen("Grafo3.txt", "rt");break;
case 4  :fichero = fopen("Grafo4.txt", "rt");break;
case 5  :fichero = fopen("Grafo5.txt", "rt");break;
case 6  :printf("Gracias por su visita.\n");break;
default :printf("Opcion incorrecta! VUELVA A INTENTARLO\n\n");break;
}
}while(opcion<1||opcion>6);

if(opcion!=6){
bandera=1;
fscanf(fichero,"%d",&n);
//RESERVA DE MEMORIA:
adya = (int **) malloc (n * sizeof(int *));
for (k=0 ; k < n ; k++) adya[k] = (int *) malloc (n * sizeof(int));

peso = (int **) malloc (n * sizeof(int *));
for (k=0 ; k < n ; k++) peso[k] = (int *) malloc (n * sizeof(int));

Arbol = (int **) malloc (n * sizeof(int *));
for (k=0 ; k < n ; k++) Arbol[k] = (int *) malloc (n * sizeof(int));

V = (char *) malloc (n * sizeof(char));

registro = (int *) malloc (n * sizeof(int));

Va = (char *) malloc (n * sizeof(char));
```

```
// INICIALIZACION DE CONJUNTOS Y MATRICES:

for (k=0 ; k < n ; k++) V[k]='a'+k;
for (k=0 ; k < n ; k++) Va[k]='a'+k;
for (k=0 ; k < n ; k++) registro[k]=0;
for (i=0 ; i < n ; i++) for (j=0 ; j < n ; j++) Arbol[i][j]=0;
//---------------------------------------------------------------------------

// LECTURA DE LAS MATRICES:

for(i=0;i<n;i++) for(j=0;j<n;j++) fscanf (fichero, "%d",&adya[i]
[j]);

for(i=0;i<n;i++) for(j=0;j<n;j++) fscanf (fichero, "%d",&peso[i]
[j]);

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// ALGORITMO:
printf("\n\nResultado del Algoritmo:\n");
printf
("\n----------------------------------------------------------------
\n\nE0={}\n");
contador=1;
while(bandera==1){
bandera=0;
printf("\nV={");
for(t=0;t<n;t++)printf("%d, ",t+1);
printf("\b\b}\n   ");
for(t=0;t<n;t++){
if((t+1)/10==0)printf("%c  ",V[t]);
else printf(" %c  ",V[t]);
}
printf
("\n----------------------------------------------------------------");
printf("\n");
// Recorre todas las componentes con etiqueta k:
for (k=0 ; k < n ; k++){
              menor=10000;
    for (i=0 ; i < n ; i++){
// Evalua solo los vertices "i" que tienen la etiqueta k:
               if(V[i]=='a'+k){

// Calculamos la arista con menor peso,saliendo de la componente k.
                for(j=0;j<n;j++){
                   if (adya[i][j]==1 && menor>peso[i][j] && V[i]!=V[j]){
                              menor=peso[i][j];
                              x=i;
                              y=j;

                     }
           }
        }
    }
// Luego de tener una arista nueva,dos componentes se vuelven una,y
```

```
// el vector Va guarda la nueva configuracion de etiquetas:

 registro[x]=1; // Se deja registro de que pasamos por el vertice x.

//-> Si el vertice "y" ya ha sido pintado:
 if(registro[y]==1){
// Pintamos todas las etiquetas que tienen valor V[x] con Va[y]
 for(t=0;t<n;t++)if(Va[t]==Va[x])Va[t]=Va[y];
 }
//-> Si el vertice "y" aun no ha sido pintado:
 else {
// Pintamos todas las etiquetas que tienen valor V[y] con Va[x]
 for(t=0;t<n;t++)if(Va[t]==V[y])Va[t]=Va[x];
 registro[y]=1; // Registramos que estamos pintando el vertice y.
 }
 Arbol[x][y]=Arbol[y][x]=1; // La arista se registra en nuestro
arbol.
}
// Se actualiza el vector de etiquetas con el vector de etiquetas
modificado:
for (k=0;k<n;k++)V[k]=Va[k];
// Los valores de registro vuelven a cero,listo para una nueva
iteracion.
for (k=0;k<n;k++)registro[k]=0;

// Imprimimos todas las aristas de nuestroo arbol:
printf("\nE%d={",contador);
for(i=0;i<n;i++) for(j=i;j<n;j++) if(Arbol[i][j]==1) printf("{%d,%
d},",i+1,j+1);
printf("\b}\n");
for(k=0;k<n-1;k++) if(V[k]!=V[k+1]) bandera=1 ;
// Si no se logra entrar a  "if(V[k]!==V[k+1])" ni una sola
vez,significara
// que todas las etiquetas tienen un solo valor,por lo cual bandera
seguira
// con valor 0,terminando con el bucle while.
contador++;
}
printf("\nV={");
for(t=0;t<n;t++)printf("%d, ",t+1);
printf("\b\b}\n   ");
for(t=0;t<n;t++){
   if((t+1)/10==0)printf("%c  ",V[t]);
   else printf(" %c  ",V[t]);
}
printf
("\n-----------------------------------------------------------------");
//-------------------------------------------------------------------------------
printf("\n");
}

}while(opcion!=6);
}// FIN DE PROGRAMA.
```

5.1 Pruebas de Algoritmo de Jarnick

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

1

Resultado del Algoritmo:

E0={}

V0={1}

E1={{1,3}}

V1={1,3}

E2={{1,2},{1,3}}

V2={1,2,3}

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

2

Resultado del Algoritmo:

E0={}

V0={1}

E1={{1,3}}

V1={1,3}

E2={{1,3},{3,4}}

V2={1,3,4}

E3={{1,2},{1,3},{3,4}}

V3={1,2,3,4}

Elige el grafo a evaluar:
1. Grafo1

2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

3

Resultado del Algoritmo:

E0={}

V0={1}

E1={{1,3}}

V1={1,3}

E2={{1,3},{1,4}}

V2={1,3,4}

E3={{1,3},{1,4},{2,3}}

V3={1,2,3,4}

E4={{1,3},{1,4},{2,3},{2,5}}

V4={1,2,3,4,5}

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

4

Resultado del Algoritmo:

E0={}

V0={1}

E1={{1,2}}

V1={1,2}

E2={{1,2},{2,5}}

V2={1,2,5}

E3={{1,2},{2,5},{5,6}}

V3={1,2,5,6}

E4={{1,2},{2,5},{3,6},{5,6}}

V4={1,2,3,5,6}

E5={{1,2},{2,5},{3,6},{5,6},{6,8}}

V5={1,2,3,5,6,8}

E6={{1,2},{2,5},{3,6},{5,6},{5,7},{6,8}}

V6={1,2,3,5,6,7,8}

E7={{1,2},{2,5},{3,6},{4,5},{5,6},{5,7},{6,8}}

V7={1,2,3,4,5,6,7,8}

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

5

Resultado del Algoritmo:

E0={}

V0={1}

E1={{1,2}}

V1={1,2}

E2={{1,2},{2,3}}

V2={1,2,3}

E3={{1,2},{2,3},{3,7}}

V3={1,2,3,7}

E4={{1,2},{2,3},{3,7},{7,11}}

V4={1,2,3,7,11}

E5={{1,2},{2,3},{3,7},{7,8},{7,11}}

V5={1,2,3,7,8,11}

E6={{1,2},{2,3},{3,7},{4,8},{7,8},{7,11}}

V6={1,2,3,4,7,8,11}

E7={{1,2},{2,3},{3,7},{4,8},{6,7},{7,8},{7,11}}

V7={1,2,3,4,6,7,8,11}

E8={{1,2},{2,3},{3,7},{4,8},{5,6},{6,7},{7,8},{7,11}}

V8={1,2,3,4,5,6,7,8,11}

E9={{1,2},{2,3},{3,7},{4,8},{5,6},{5,9},{6,7},{7,8},{7,11}}

V9={1,2,3,4,5,6,7,8,9,11}

E10={{1,2},{2,3},{3,7},{4,8},{5,6},{5,9},{6,7},{7,8},{7,11},{9,10}}

V10={1,2,3,4,5,6,7,8,9,10,11}

E11={{1,2},{2,3},{3,7},{4,8},{5,6},{5,9},{6,7},{7,8},{7,11},{8,12},
{9,10}}

V11={1,2,3,4,5,6,7,8,9,10,11,12}

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

6
ADIOS

5.2. Pruebas de Algoritmo de Boruvka

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

1

Resultado del Algoritmo:

E0={}

V={1, 2, 3}
  a b c

E1={{1,2},{1,3}}

V={1, 2, 3}
  a a a

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

2

Resultado del Algoritmo:

E0={}

V={1, 2, 3, 4}
  a b c d

E1={{1,2},{1,3},{3,4}}

V={1, 2, 3, 4}
  a a a a

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

3

Resultado del Algoritmo:

E0={}

V={1, 2, 3, 4, 5}
  a b c d e

E1={{1,3},{1,4},{2,3},{2,5}}

V={1, 2, 3, 4, 5}
  a a a a a

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

4

Resultado del Algoritmo:

E0={}

V={1, 2, 3, 4, 5, 6, 7, 8}
  a b c d e f g h

E1={{1,2},{3,6},{4,5},{5,6},{5,7},{6,8}}

V={1, 2, 3, 4, 5, 6, 7, 8}
  a a c c c c c c

E2={{1,2},{2,5},{3,6},{4,5},{5,6},{5,7},{6,8}}

V={1, 2, 3, 4, 5, 6, 7, 8}
  a a a a a a a a

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

5

Resultado del Algoritmo:

E0={}

V={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
  a b c d e f g h i j k l

E1={{1,2},{2,3},{4,8},{5,6},{7,8},{7,11},{8,12},{9,10}}

V={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
  a a a g e e g g i i g g

E2={{1,2},{2,3},{3,7},{4,8},{5,6},{5,9},{7,8},{7,11},{8,12},{9,10}}

V={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
  a a a a e e a a e e a a

E3={{1,2},{2,3},{3,7},{4,8},{5,6},{5,9},{6,7},{7,8},{7,11},{8,12},
{9,10}}

V={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
  a a a a a a a a a a a a

Elige el grafo a evaluar:
1. Grafo1
2. Grafo2
3. Grafo3
4. Grafo4
5. Grafo5
6. No elegir y Salir

6

Gracias por su visita.

4. Implementación en Lenguaje C

4.1. Pseudocódigo del algoritmo de Jarnick

INICIO

```
 i,j,k     ← Contadores
menor ← Peso menor del Grafo
n         ← Numero de vertices del grafo
respaldo1,respaldo2 ← Respaldo de Posiciones
V         ←Vector binario que indica los vertices escogidos
peso      ←Matriz de pesos del grafo
adya      ←Matriz de Adyacencia del grafo

Para k=0 hasta k= n-2 Hacer
        menor ←1000000
        Para  i=0 hasta i= n-1 Hacer
            Si V[i]==1 entonces
                Para  j=0 hasta j= n-1 Hacer
                   Si peso[i][j]< menor y adya[i][j]==1 y V[i]!=V[j] entonces
                         menor       ←peso[i][j]
                         respaldo1 ← i
                         respaldo2 ← j
                    Fin Si.
                    Fin Para j.
            Fin Si.
        Fin Para i.
        adya[respaldo1][respaldo2] ← 2
        adya[respaldo2][respaldo1] ← 2
        V[respaldo2] ← 1

    Mostrar Resultados

    Fin Para k.
FIN DE ALGORITMO
```

4.2. Pseudocódigo del algoritmo de Boruvka

INICIO
i,j,k,t ← Contadores
menor ← Peso menor del Grafo
n ← Numero de vertices del grafo
x,y ← Respaldo de Posiciones
registro ← Vector binario que indica que vertices han sido etiquetados.
V,Va ←Vectores de etiquetas
peso ←Matriz de pesos del grafo
adya ←Matriz de Adyacencia del grafo
Arbol ← Matriz de adyacencia del arbol

Mientras bandera==1 **Hacer**
bandera ← 0
Para k=0 hasta k= n-1 **Hacer**
menor ←1000000
Para i=0 hasta i= n-1 **Hacer**
Si V[i]=='a'+k **entonces**
Para j=0 hasta j= n-1 **Hacer**
Si adya[i][j]==1 y menor>peso[i][j] y V[i]!=V[j] **entonces**
menor ←peso[i][j]
x ← i
y ← j
Fin Si.
Fin Para j.
Fin Si.
Fin Para i.
Pintar etiquetas y actualizar vector V.
Arbol[x][y] ← 1
Arbol[y][x] ← 1
Fin Para k.
Actualizar Vector Va con V.

Mostrar Resultados
Fin Mientras.
FIN DE ALGORITMO

6. Conclusiones

Los 3 algoritmos implementados,resuelven el problema de encontrar el arbol minimal de un Grafo con Pesos en las aristas, aunque estas implementaciones no han sido optimizadas, representan la base para futuras implementaciones en otros lenguajes con un paradigma mas avanzado.

7. Referencias Bibliograficas

1.JOYANES, AGUILAR LUIS Y ZAHONERO, MARTÍNEZ IGNACIO. ESTRUCTURA DE DATOS. ALGORITMOS, ABSTRACCIÓN Y OBJETOS. EDIT MC GRAW-HILL, MADRID, 1998.

2.JOYANES, AGUILAR LUIS. FUNDAMENTOS DE PROGRAMACIÓN. 2ª ED. EDIT. MCGRAW-HILL, MADRID, 1996.

3.LANGSAM, YEDIDYAH; AUGENSTEIN, MOSHE y TENEMBAUM, AARON M. ESTRUCTURAS DE DATOS EN C 2ª ED. EDIT. PRENTICE-HALL . MÉXICO, 1996.

4.CAIRÓ, OSVALDO y GUARDATI SILVIA. ESTRUCTURAS DE DATOS. EDIT. McGRAW-HILL. MÉXICO, 1992

1. Introducción

Hablando en un ambiente mundialista, observamos que todo viajero inteligente mas que solo asistir a los partidos piensa en cuanto le costara ver cada uno de estos, en particular los costos de transportes. Como sabemos los partidos del mundial se daran en 12 estadios y es de suponer que para dirigirnos desde el hotel a cada uno de estos estadios hay distintos caminos de los cuales uno de estos caminos es el ideal (menor costo) para llegar a un estadio en particular.

El problema del arbol generador minimal fue formulado y resuelto por primera vez en 1926 por Otakar Boruvka. Consultado sobre el diseno de una red electrica que permitiese una distribucion eficiente de la electricidad en la region de Moravia, el matemático checo modelo la situacion como problema de arboles y las aristas que unian tenian asociados un peso equivalente a las distancias. Boruvka propuso una solucion basada en la adicion repetida de conexiones entre las componentes del grafo hasta llegar a formar un arbol de expansion minimo.

Observamos en lo expuesto anteriormente ejemplos de problematicas reales que con ayuda de los grafos y en particular de los arboles se puede dar solucion a estos. Dentro de todos los metodos (algoritmos) que se conocen para resolver esta problematica este informe estara concentrado en exponer 2 de estos, los cuales son el algoritmo de Jarnick y Boruvka.

2. Marco teórico

Hablando intuitivamente, un grafo es un conjunto de vertices unidos por un conjunto de lineas o flechas. Por lo general, los vertices son entes de procesamiento o estructuras que contienen algun tipo de informacion y las lineas o flechas son conexiones o relaciones entre estos entes. Si se utilizan flechas para conectar los vertices decimos que el grafo es dirigido porque las realciones entre los vertices tienen una direccion. En caso contrario el grafo es no dirigido. En cualquiera de los dos casos, bien sea que se utilicen aristas o flechas , a estas relaciones se les puede llamar simplemente aristas. Frecuentemente las aristas tambien tienen algun tipo de informacion asociada(distancia, costo, fiabilidad, etc), en cuyo caso estamos en presencia de un grafo pesado.

Las secuencias de aristas forman caminos o ciclos. Un ciclo es un camino que termino en el mismo vertice donde comenzo. Si el camino recorre todos los vertices del grafo es llamado paseo. El numero de aristas en un camino es la longitud del camino.

Se dice que un grafo es conexo si se puede llegar desde cualquier vertice hasta cualquier otro mediante un camino. De lo contrario no es conexo, pero puede dividirse en componentes conexas, que son subconjuntos de vertices y aristas del grafo original que si son conexos. Un grafo conexo sin ciclos es llamado un arbol.

Dado un grafo no dirigido pesado y conexo, se pretende encontrar un subgrafo del mismo que sea un arbol de expansion cuyo peso sea minimo. Es decir, un subgrafo que conecte todos sus vertices y en el que la suma del peso de sus aristas sea menor o igual a la suma del peso de las aristas que forman cualquier otro arbol de expansion presente en el grafo.

Algoritmo (creacion de un arbol generador)

Sea G=(V,E) el grafo que tiene n vertices y m ramas. Contruimos de forma recursiva los conjuntos V0,V1,V2,… incluidos en V de vertices y los conjuntos E0,E1,E2,…. Incluidos en E de ramas. Sea E0={} y V0={v}, donde v es un vertice arbitrario. Habiendo ya contruido Vi-1 y Ei-1, encontramos una rama ei={xi,yi} perteneciente a E(G) tal que xi pertenece a Vi-1 e yi pertenece a V\Vi-1 , e introducimos el conjunto Vi= Vi-1 unido {yi}, Ei = Ei-1 unido a {ei}. Si no existe una rama tal, el algoritmo termina y nos da tambien el grafo construido T=(Vt,Et).

Algoritmo de jarnick

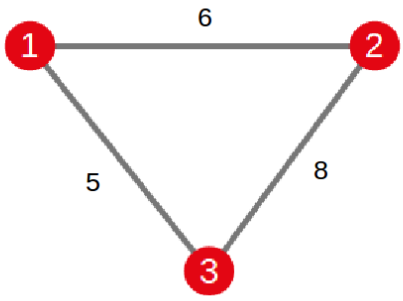
Procede según el algoritmo de creacion de un arbol generador, y siempre elige una nueva rama ei como una rama del msnor peso posible dentro del conjunto ({x,y} perteneciente a E(G): x pertenece Vi-1, y no pertenece a Vi-1}

Algoritmo de boruvka

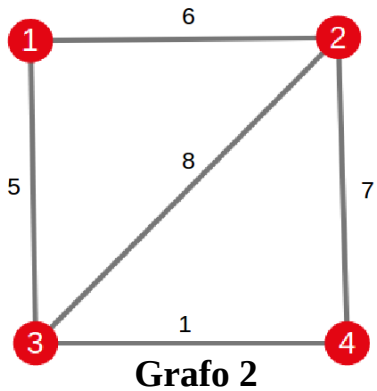
El algoritmo comienza sin aristas y con todos los vertices,se examina las componentes conexas que existen,añadiendo la rama de menor peso que sale de una componente en especifico, sin tener en cuenta las ramas ya agregadas,evalua asi cada componente conexa hasta evaluar todas. Finalmente en cada paso del algoritmo las componentes conexas van reduciendoce en numero, hasta convertirse en una sola componente.

5. Pruebas y Resultados

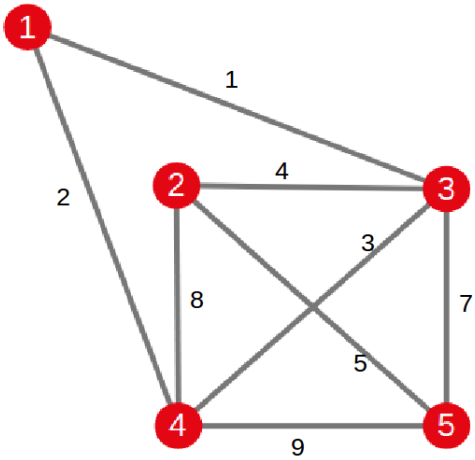
Para testear nuestro código y demostrar su efectividad para obtener el arbol generador minimal, se han seleccionado 5 grafos:



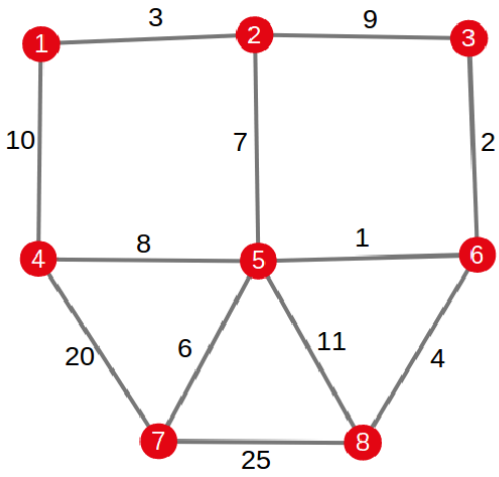
Grafo 1



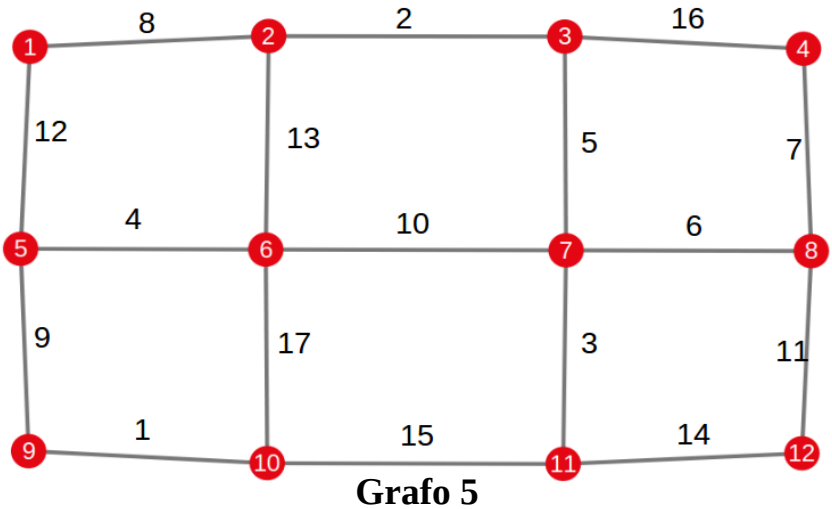
Grafo 2



Grafo 3



Grafo 4



Grafo 5

Es importante enfatizar que los programas realizados, reciben los datos de estos grafos mediante la lectura de ficheros en C, para ser precisos estos datos estan almacenados en 5 archivos de texto, los cuales poseen las matrices de pesos y de adyacencia de los grafos 1,2,3,4 y 5.

3. Demostración Matemática del Algoritmo de Jarnick

- Sea T=(V;E') el árbol mínimo generador resultante, con ramas enumeradas e1;e2; ... ; en-1; como resultado de Jarnick.
- Supongamos que T no es el árbol mínimo, entonces tomamos T' y definimos a un índice k como k(T') para quien:

  $e1;e2; \dots ; ek$ son de T' pero $ek+1 \notin E(T')$
- Ahora elegiremos un T'' el cuál será el que tenga al máximo índice k.
  Denotaremos T"=(V;E") y a k=k(T")
- Sea Tk=(Vk; Ek)
  Un árbol formado por las ramas e1;e2; … ; ek
  Cuando el algoritmo se ejecute, se añadirá la rama ek+1 tiene la forma {x;y}
  tal que $x \in Tk$, $y \notin Tk$
- Veamos que si consideramos T" + ek+1 logramos ver que es conexo y además que tiene mas de n-1 ramas.

  Por lo tanto T" + ek+1 es cíclico.
- El ciclo C consiste en que hay más de un solo camino P para ir de x a y en el árbol generador T".
  tener en cuenta que la rama ek+1 que tiene la forma {x,y}

Una rama del camino P tiene como mínimo un vértice en Vk y otro fuera de Vk

- Sea $e \neq ek+1$ una de estas ramas ($e \in E"$, $ek+1 \notin E"$)
  Ambas ramas conectan un vértice que está en Vk y otro fuera de Vk
  Sea W: peso

- Ahora consideramos el grafo

  $T' = (T" + ek+1) - e.$

  Este grafo tiene n-1 ramas y es conexo,por lo tanto T' es un árbol generador.

- Por el algoritmo de selección de ramas:

  $W(ek+1) \leq W(e) \dots (1)$

- Veamos el siguiente arreglo:

  $W(E(T')) = W(E") - W(e) + W(ek+1)$
- De (1) usamos la desigualdad:

  $W(E(T')) = W(E") - W(e) + W(ek+1) \leq W(E")$

- Notamos que T' es un árbol generador mínimo también pero con

  $k(T'') < k(T')$

Lo cual contradice lo supuesto