LABORATORIO 11

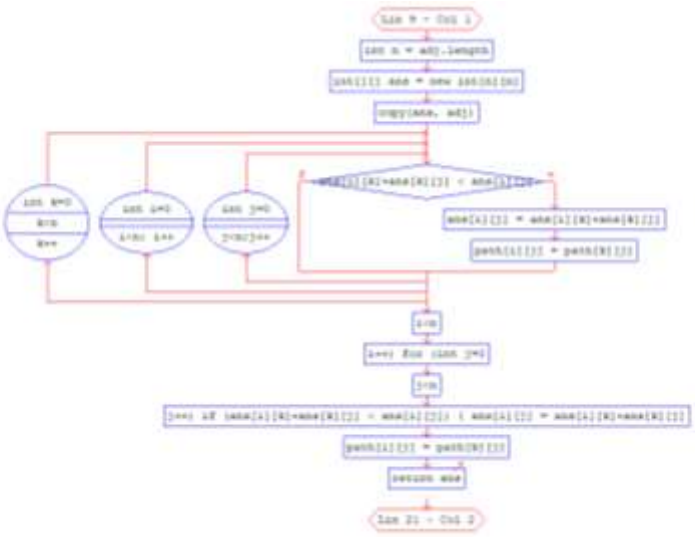
INGENIERIA EN COMPUTACION

MARIANA ESTEFANIA BARCENAS RODRIGUEZ

UAZ 3°A

Actividad 19

1. **Diagrama de Flujo :**



2.codigo:

```
//estructura de datos
//mariana estefania barcenas rodriguez
// 22/05/2022

import java.util.*;
public class Floyd
{
    public static int[][] shortestpath(int[][] adj, int[][] path)
    {
        int n = adj.length;
        int[][] ans = new int[n][n];
        copy(ans, adj);
        for (int k=0; k<n;k++)
            for (int i=0; i<n; i++)
            for (int j=0; j<n;j++)
                    if (ans[i][k]+ans[k][j] < ans[i][j]) {
                        ans[i][j] = ans[i][k]+ans[k][j];
                        path[i][j] = path[k][j];
                    }
        return ans;
    }
```

```
    public static void copy(int[][] a, int[][] b)
    {
        for (int i=0;i<a.length;i++)
            for (int j=0;j<a[0].length;j++)
                a[i][j] = b[i][j];
    }
    public static void main(String[] args)
    {
        Scanner stdin = new Scanner(System.in);

        int[][] m = new int[5][5];
        m[0][0] = 'a';
        m[0][1] = 2;
        m[0][2] = 0;
        m[0][3] = 2;
        m[0][4] = 'b';
        m[1][0] = 4;
        m[1][1] = 2;
        m[1][2] = 2;
        m[1][3] = 'c';
        m[1][4] = 4;
        m[2][0] = 2;
        m[2][1] = 2;
        m[2][2] = 'd';
        m[2][3] = 3;
        m[2][4] = 3;
        m[3][0] = 0;
        m[3][1] = 'e';
        m[3][2] = 3;
        m[3][3] = 1;
        m[3][4] = 2;
        m[4][0] = 10000;
        m[4][1] = 10000;
        m[4][2] = 2;
        m[4][3] = 7;
        m[4][4] = 0;
        int[][] shortpath;
        int[][] path = new int[5][5];

        for (int i=0; i<5; i++)
            for (int j=0; j<5; j++)
                if (m[i][j] == 10000)
                    path[i][j] = -1;
            else
                    path[i][j] = i;
```

```
        for (int i=0; i<5; i++)
            path[i][i] = i;

        shortpath = shortestpath(m, path);

        System.out.println(" a=1 b=2 c=3 d=4");

        System.out.println("  0 1 2 3 4");
        System.out.println("  ---------");
        for (int i=0; i<5;i++) {
            System.out.print(i + "|");
            for (int j=0; j<5;j++)
                System.out.print(shortpath[i][j]+" ");
            System.out.println();
        }
        System.out.println("Ruta mas corta de un vertice a otro (0 a 4)");
        System.out.print("Vertice inicial: ");
        int start = stdin.nextInt();
        System.out.print("Vertice final: ");
        int end = stdin.nextInt();
         String myPath = end + "";
         System.out.println();
        System.out.println("  0 1 2 3 4");
        System.out.println("  ---------");
         for (int i=0; i<5;i++) {
            System.out.print(i + "|");
            for (int j=0; j<5;j++)
                System.out.print(path[i][j]+" ");
            System.out.println();
        }
        while (path[start][end] != start) {
            myPath = path[start][end] + " -> " + myPath;
            end = path[start][end];
        }
        myPath = start + " -> " + myPath;
        System.out.println("Esta es la ruta: " + myPath);
    }
}
```

ACTIVIDAD 20.

Diagrama.

Código.

```
//estructura de datos
//mariana estefania barcenas rodriguez
// 24/05/2022

#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <ctype.h>
typedef struct Nodos{
    char info;
    struct Nodos *liga;
}tiponodols;
typedef struct nodoG{
    char info;
    int numAristasInc;
    int numAristaEntra;
    int numAristaSal;
    int pos;
```

```
}tiponodo;
typedef struct arista{
    int info;
    bool dirigida;
    struct nodoG *origen;
    struct nodoG *destino;
    int pos;
}tipoArista;
typedef tiponodo *NodoG;
typedef tipoArista *AristaG;
typedef tiponodols *nodo;
nodo p,q,t=NULL;
nodo Lista;
void CrearFinal(int Elem);
//conjuntos de vértices
NodoG* Grafos; //utilización de un arreglo (dinámico)
//conjunto de aristas
AristaG* Aristas;
int NumVertices;
int NumAristas;
//operaciones crear grafo en memoria
NodoG NuevoNodoG(int,int,int,int,int);
NodoG* CargarGrafo(); //retorna el conjunto de vértices (arreglo)
void MuestraConjuntoVertice(NodoG*);
void MuestraGrafo(NodoG*);
//operaciones aristas
AristaG NuevaArista(int,NodoG,NodoG,bool,int );
AristaG* CrearAristas();
NodoG aVertice(char,NodoG *);
void MuestraConjuntoAristas(AristaG*);
void MatrizAdy(NodoG*,AristaG*);//genera y muestra
void ListaAdyacente(NodoG*,AristaG*);
void MatrizDistanciaF(NodoG* G,AristaG* Ag);
int esInfinitoCero(int a, int b);

//operaciones de lista enlazada para crear la ListaAdyacente
nodo nuevoNodoL(char inf,nodo liga){
    nodo q;
    q=(nodo)malloc(sizeof(tiponodols));
    if (!q){
        printf("\n Error al crear el nuevo nodo");
        exit(0);
    }
    q->info=inf;
    q->liga=liga;
```

```
    return q;
}
    void CrearFinal(int Elem){
        nodo n;
        if (p==NULL) //ES el primer nodo
        {
            p=nuevoNodoL(Elem,NULL);
            t=p;
        }
        else{ //nodos subsecuentes
            n=nuevoNodoL(Elem,NULL);
            t->liga=n;
            t=n;
        }
    }
        nodo RecorrerUltimo(nodo lista){
            t=lista;
            while (lista!=NULL){
                t=lista;
                lista=lista->liga;
            }
            return t;
        }

            void Recorrer(nodo Lista){
                while (Lista!=NULL){
                    printf("%c -> ",Lista->info);
                    Lista=Lista->liga;
                }
            }

                //variables globales
                NodoG *ArrayGrafos;
                AristaG *ArrayAristas ;
                // matiz dinamica
                int **MatrizDistancia;
                int main()
                {
                    ArrayGrafos=CargarGrafo ();
                    printf ("Conjunto de vértices del grafo \n");
                    MuestraConjuntoVertice (ArrayGrafos);
                    ArrayAristas=CrearAristas ();
                    printf ("Conjunto de aristas del grafo \n");
                    MuestraConjuntoAristas (ArrayAristas );
```

```
                    MatrizAdy (ArrayGrafos,ArrayAristas );
                    ListaAdyacente(ArrayGrafos,ArrayAristas);
                    printf("Mactriz de distancia" );
                    MatrizDistanciaF (ArrayGrafos,ArrayAristas );

                    getch ();
                    return 0 ;
/*
                    FILE *fp;
                    char *cadena;
                    if( (fp=fopen("ConjuntoVerticesD2.txt","r")) !=NULL)
                    {

                         do
                          {
                                fgets(cadena, 20, fp);//fgets
                                MuestraConjuntoVertice (ArrayGrafos);
                                ArrayAristas=CrearAristas ();
                          }while(!feof(fp));

                         fclose(fp);
                    }
                    else
                        printf("\n Error al leer el archivo");

                    if( (fp=fopen("ConjuntoAristasD2.txt","r")) !=NULL)
                    {
                        do
                          {
                                fgets(cadena, 20, fp);//fgets
                                MuestraConjuntoAristas (ArrayAristas );
                                MatrizAdy (ArrayGrafos,ArrayAristas );
                               ListaAdyacente(ArrayGrafos,ArrayAristas);
                               printf("Mactriz de distancia" );
                               MatrizDistanciaF (ArrayGrafos,ArrayAristas );
                          }while(!feof(fp));
                    }
                    else
                        printf("\n Error al leer el archivo");
                    */

                    return 0;

                }
                //cuerpo funciones
```

```
NodoG NuevoNodoG(int dato,int Ninc ,int Nentra,int Nsal, int
p)
{
    NodoG n;
    n=(NodoG)malloc(sizeof (tiponodo));
    if (!n) {
        printf ("\n Error al crear vértice");
        exit(0);
    }


    n->info=dato;
    n->numAristasInc=Ninc;
    n->numAristaEntra=Nentra;
    n->numAristaSal=Nsal;
    n->pos=p ;

    return n;

}

NodoG* CargarGrafo()
{
    char dato;
    int nInc=0,nEntra=0,nSal=0;
    NodoG * Grafos;
    /*printf ("\n cuantos vertices tiene el grafo \n");
    scanf("%d",&NumVertices);*/
     FILE *fp;
    char *cadena;
    if( (fp=fopen("ConjuntoVerticesD2.txt","r")) !=NULL)
    {

         do
          {
                fgets(cadena, 20, fp);//fgets
                MuestraConjuntoVertice (ArrayGrafos);
                ArrayAristas=CrearAristas ();
           }while(!feof(fp));

         fclose(fp);
    }
    else
        printf("\n Error al leer el archivo");
```

```
                    fflush(stdin);
                    Grafos = (NodoG *)malloc(NumVertices *
sizeof(tiponodo));
                    /*for(int i=0;i<NumVertices;i++)
                    {
                        printf ("\n Dame la informacion del vertice %d",i);
                        scanf("%c",&dato);
                        fflush(stdin);
                        printf ("\n Dame el numero de la arista incidente
del vertice %d",i);
                        scanf("%d",&nInc);
                        printf ("\n Dame el num de la arista entrante del
vertice  %d",i);
                        scanf("%d",&nEntra);
                        printf ("\n Dame el num de la arista saliente del
vertice  %d",i);
                        scanf("%d",&nSal);
                        Grafos[i]=NuevoNodoG(dato,nInc,nEntra,nSal,i);
                    }*/
                    getch();
                    return Grafos;
                }

                void MuestraConjuntoVertice(NodoG* Vg){
                    for (int i = 0; i < NumVertices; i++){
                        printf("vertice = [%c]\n",Vg[i]->info);
                    }
                }

                AristaG NuevaArista( int dato,NodoG Origen,NodoG
Destino,bool dirigida, int p)
                {
                    AristaG a;
                    a=(AristaG)malloc(sizeof(tipoArista));
                    if (!a)
                    {
                        printf ("\n error no se pudo crear la arista");
                        return NULL;
                    }

                    a->info=dato;
                    a->origen=Origen;
```

```
                    a->destino=Destino;
                    a->dirigida=dirigida;
                    a->pos=p;
                    return a;

                }

                AristaG* CrearAristas()
                {
                    int dato;
                    char o,d;
                    bool dir;
                    char opc;
                    NodoG ori,dest;
                    FILE *fp;
                    char *cadena;
                    /*printf ("\n cuantas aristas tiene el grafo");
                    scanf("%d",&NumAristas);*/
                    if( (fp=fopen("ConjuntoAristasD2.txt","r")) !=NULL)
                    {
                        do
                          {
                                fgets(cadena, 20, fp);//fgets
                                MuestraConjuntoAristas (ArrayAristas );
                                MatrizAdy (ArrayGrafos,ArrayAristas );
                               ListaAdyacente(ArrayGrafos,ArrayAristas);
                               printf("Mactriz de distancia" );
                               MatrizDistanciaF (ArrayGrafos,ArrayAristas );
                          }while(!feof(fp));
                    }
                    else
                        printf("\n Error al leer el archivo");
                    fflush(stdin);
                    Aristas=(AristaG *)malloc(NumAristas *
sizeof(tipoArista));
                    for (int i = 0; i < NumAristas; i++)
                    {
                        fflush(stdin);
                       //printf ("\n Dame el nombre o peso de la arista");
                        //scanf("%d",&dato);
                        fflush(stdin);
                        printf("Dame el nodo origen \n");
                        scanf("%c",&o);
                        fflush(stdin);
                        printf("dame el nodo destino \n");
```

```
                        scanf("%c",&d);
                        fflush(stdin);
                        ori =aVertice(o,ArrayGrafos);
                        if(ori==NULL){
                            printf("no exite en el vertice origen \n");
                            exit(0);
                        }
                        dest=aVertice(d,ArrayGrafos);
                        if(dest == NULL ){
                            printf("no exite el vertice destino \n");
                            exit(0);
                        }
                        printf("La arista es dirigida (s) o no dirigida
(n)");
                        scanf("%c",&opc);
                        if(toupper(opc)=='S')
                            dir = true;
                        else if(toupper(opc)=='N')
                            dir = false;
                        Aristas[i]=NuevaArista(dato,ori,dest,dir,i);

                    }//fin for
                    return Aristas;

                }//fin funcion
                NodoG aVertice(char v,NodoG *grafo){
                    if(grafo != NULL){
                        for(int i =0;i<NumVertices;i++){
                            if(v==grafo[i]->info)
                                return grafo[i];
                        }
                    }
                    return NULL;
                }
                void MuestraConjuntoAristas(AristaG *Ag){
                    printf("Nodo origen ------------Arista----------Nodo
Destino\n");
                    for(int i=0;i<NumAristas;i++){
                        if(Ag[i]->dirigida==false)
                            printf("  (%c)  ------------ %d ----------
-   (%c)   \n",
                             Ag[i]->origen->info,Ag[i]->info,Ag[i]->destino-
>info);
                        else if(Ag[i]->dirigida==true)
```

```
                            printf("   (%c)   ----------%d-----------
>    (%c)   \n",
                                    Ag[i]->origen->info,Ag[i]->info,Ag[i]-
>destino->info);
                        printf("\n");
                     }
                }
                void MatrizAdy(NodoG* G,AristaG * Ag){
                    int **MatrizAdyacente;
                    MatrizAdyacente=(int **)malloc(NumVertices*sizeof(int
*));
                    for(int i = 0;i<NumVertices;i++){
                        MatrizAdyacente[i]=(int
*)malloc(NumVertices*sizeof(int *));
                    }
                    for(int i=0;i<NumVertices;i++){
                        for(int j=0;j<NumVertices;j++){
                            MatrizAdyacente[i][j]=0;
                        }
                    }
                    for(int i=0;i<NumAristas;i++){
                        MatrizAdyacente[Ag[i]->origen->pos][Ag[i]->destino-
>pos]=1;
                    }
                    for(int i=0; i< NumVertices;i++){
                        printf("%d ",i+1);
                    }
                    for(int i=0;i<NumVertices;i++){
                        printf("\n%d",i+1);
                        for(int j=0;j<NumVertices;j++){
                            printf(" [%d] ",MatrizAdyacente[i][j]);
                        }
                    }

                }
                void ListaAdyacente(NodoG* G,AristaG* Ag){
                    nodo *ListaAdy;
                    ListaAdy=(nodo *)malloc((NumVertices)*sizeof(nodo *));
                    for(int i=0;i<NumVertices ;i++){
                        ListaAdy[i]=nuevoNodoL(-1,NULL);
                    }
                    for(int i=0;i<NumAristas;i++){
                        if(ListaAdy[Ag[i]->origen->pos]->info==-1){
                            ListaAdy[Ag[i]->origen->pos]->info=Ag[i]-
>origen->info;
```

```
                        }
                        Lista=ListaAdy[Ag[i]->origen->pos];
                        p=Lista;
                        t=RecorrerUltimo(Lista);
                        CrearFinal(Ag[i]->destino->info);
                    }
                    printf("\n");
                    for(int i=0;i<NumVertices;i++){
                        printf("ListaAdy [%c]-> ",ListaAdy[i]->info);
                        Recorrer(ListaAdy[i]->liga);
                        printf("\n");

                    }
                }

                void MatrizDistanciaF(NodoG* G,AristaG* Ag){
                    MatrizDistancia=(int
**)malloc(NumVertices*sizeof(int*));
                    for(int i=0;i<=NumVertices;i++){
                            MatrizDistancia[i]=(int
*)malloc(NumVertices*sizeof(int));
                    }
                    for(int i=0;i<NumVertices;i++){
                        for(int j=0;j<NumVertices;j++){
                            if(i==j){
                                MatrizDistancia[i][j]=0;
                            }
                            else{
                                MatrizDistancia[i][j]=-1;
                            }
                        }
                    }
                    for(int i=0;i<NumAristas;i++){
                        MatrizDistancia[Ag[i]->origen->pos][Ag[i]->destino-
>pos]=Ag[i]->info;
                    }
                    for(int i=0;i<NumVertices;i++){
                        printf("  %c  ",G[i]->info);
                        for(int j=0;j<NumVertices;j++){
                            printf(" [%d] ",MatrizDistancia[i][j]);
                        }
                    }
                }
```