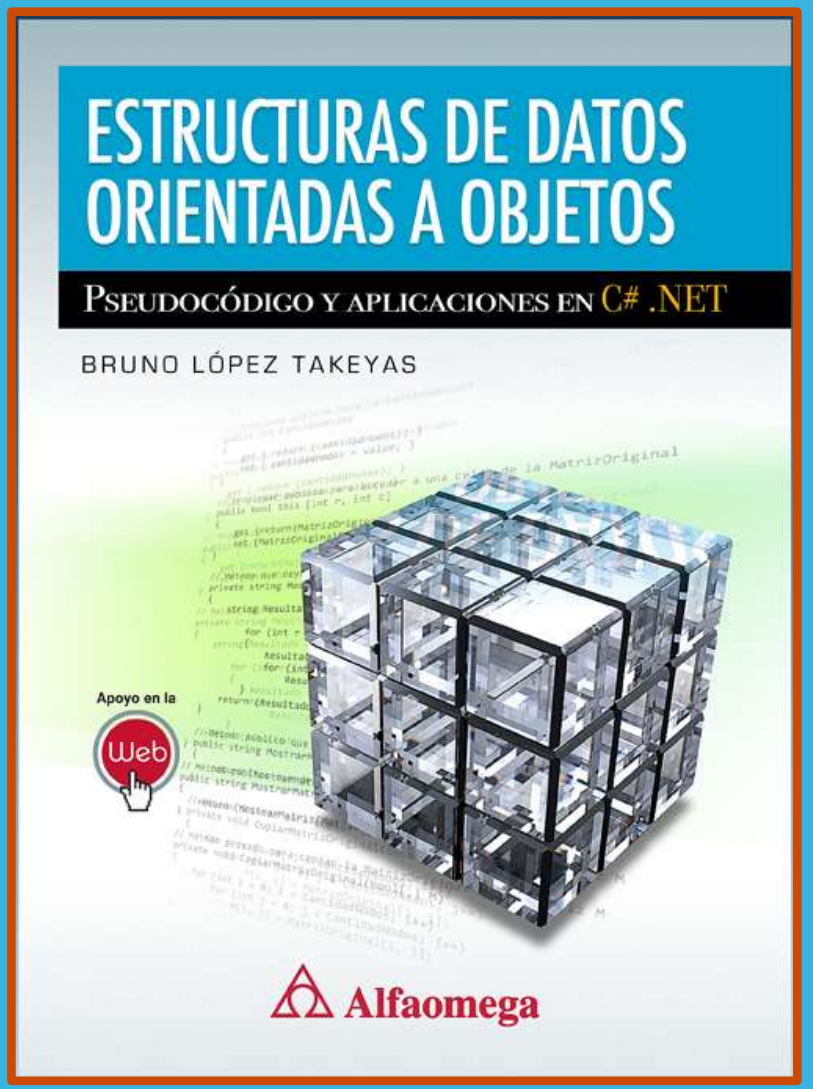
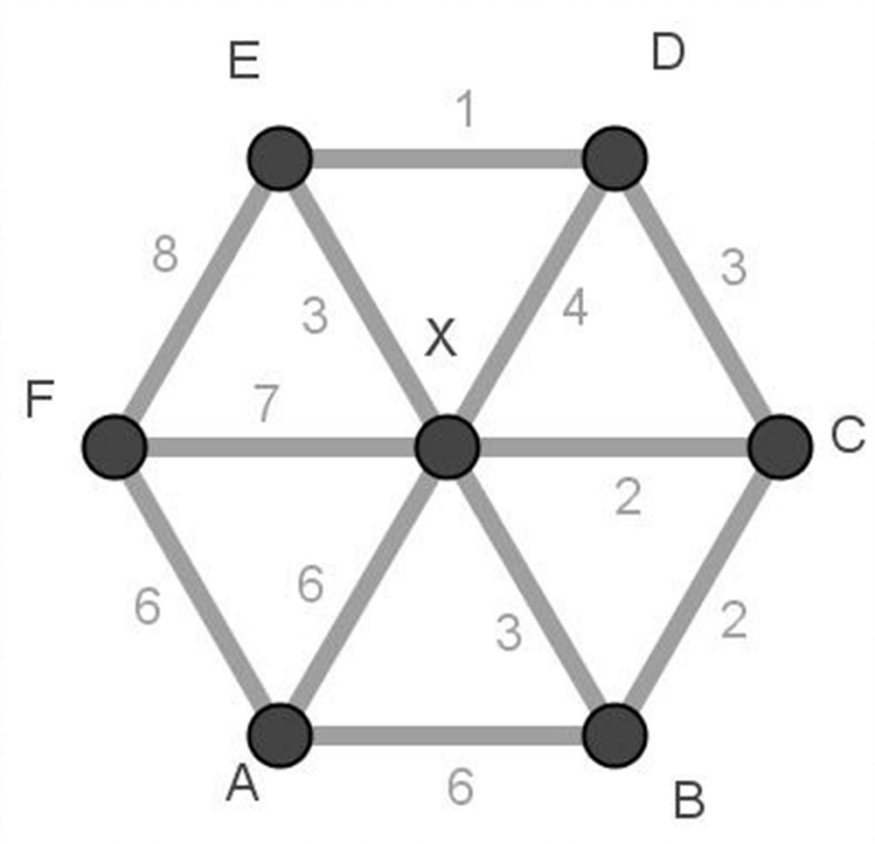
ESTRUCTURAS DE DATOS ORIENTADAS A OBJETOS

Pseudocódigo y aplicaciones en C# .NET

BRUNO LÓPEZ TAKEYAS



GRAFOS

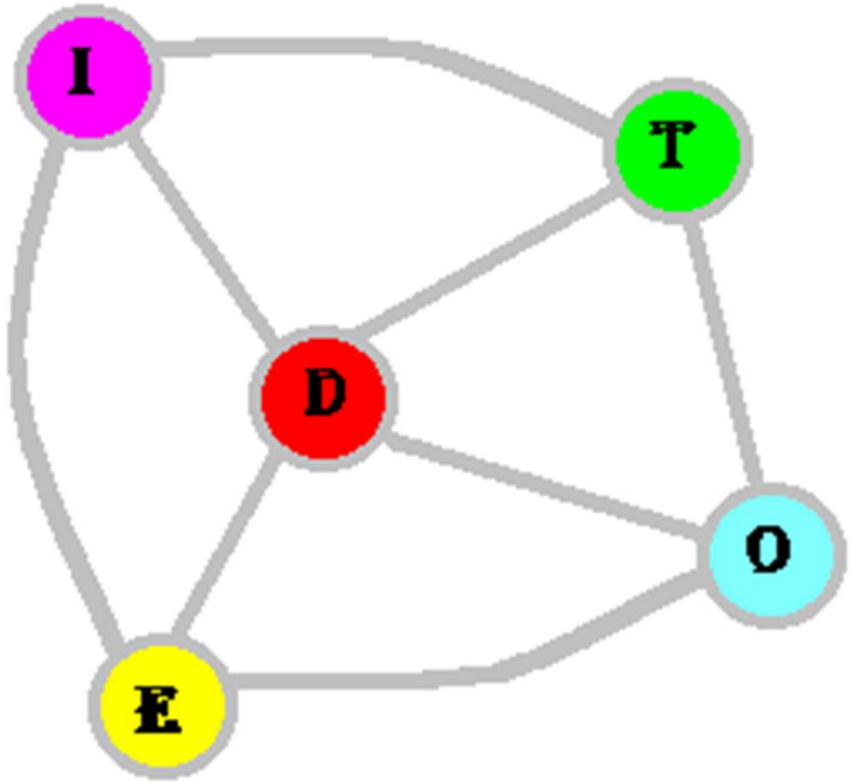


Preguntas detonadoras

- ¿Qué es un grafo?
- ¿Cómo se clasifican los grafos?
- ¿Qué características distinguen a un grafo dirigido?
- ¿ … y a un grafo no dirigido?
- ¿Cómo se representa gráficamente un grafo?
- ¿Qué operaciones se pueden realizar en un grafo?
- ¿Cómo se diseña un modelo orientado a objetos con un grafo?
- ¿Cómo se puede dibujar un grafo?
- ¿Dónde se aplican los grafos?

SESIÓN 1

Introducción y Representación secuencial



Grafo

Un grafo es una estructura de datos no lineal y dinámica que se compone de dos conjuntos:

- ***Nodos o vértices N(G):*** *Representan características, descripciones o situaciones particulares de un problema y se representa gráficamente como un círculo o elipse con un dato o información asociada.*

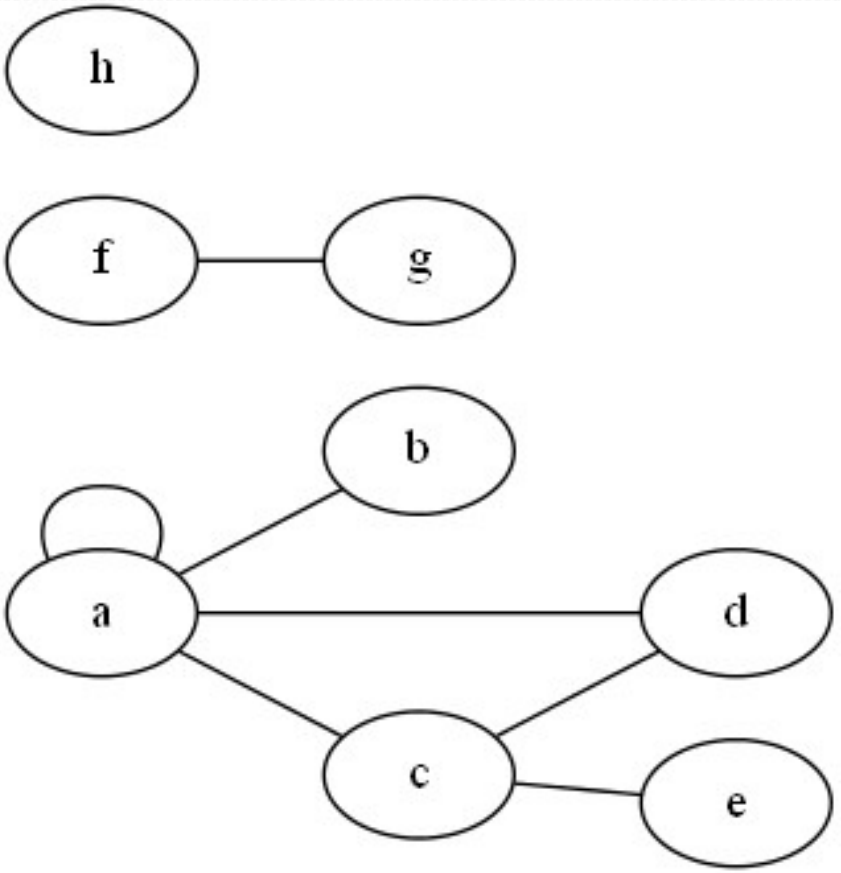
- ***Arcos o aristas A(G):*** *Son las relaciones o conexiones que unen a los nodos. Un arco se representa mediante una tupla o par de nodos de tipo (a,b), donde a y b son nodos del conjunto N y se representa gráficamente por una línea que une a los nodos a y b.*

Grafo

Un grafo se puede definir como

$G = (N, A)$

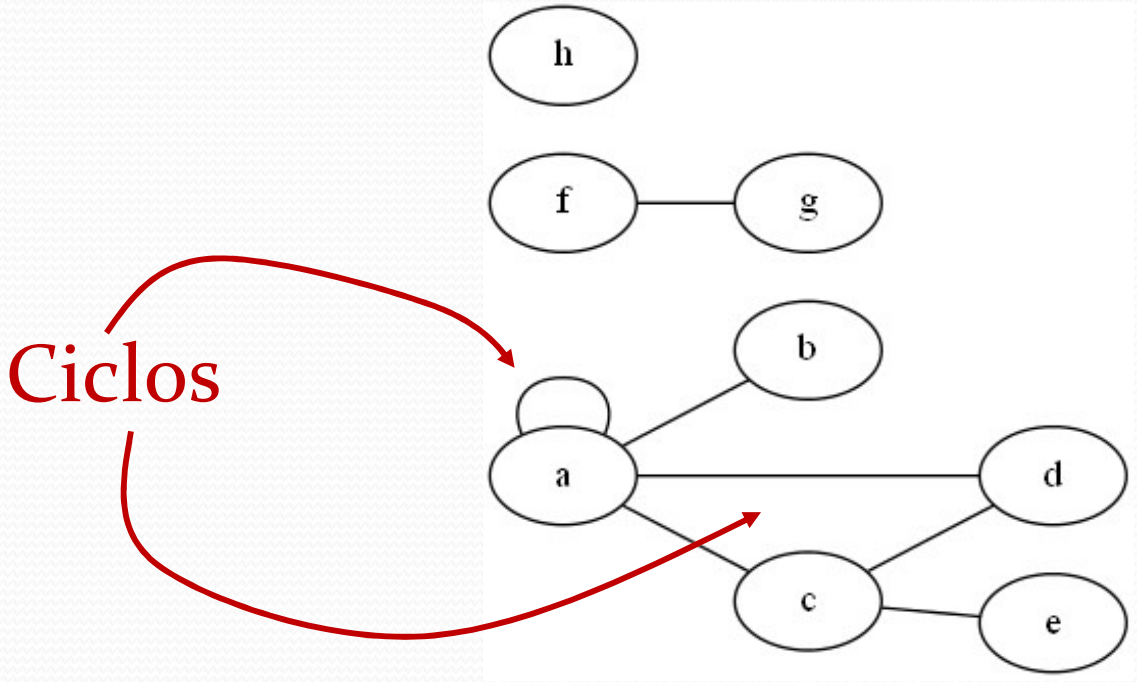
- ***N:*** *Conjunto de nodos*
- ***A:*** *Conjunto de arcos*



Conjuntos de un grafo

$G = (N, A)$

- $N=\{a, b, c, d, e, f, g, h\}$
- $A=\{ (a,a), (a,b), (a,c), (a,d), (c,d), (c,e), (f,g) \}$



Tipos de grafos

GRAFO

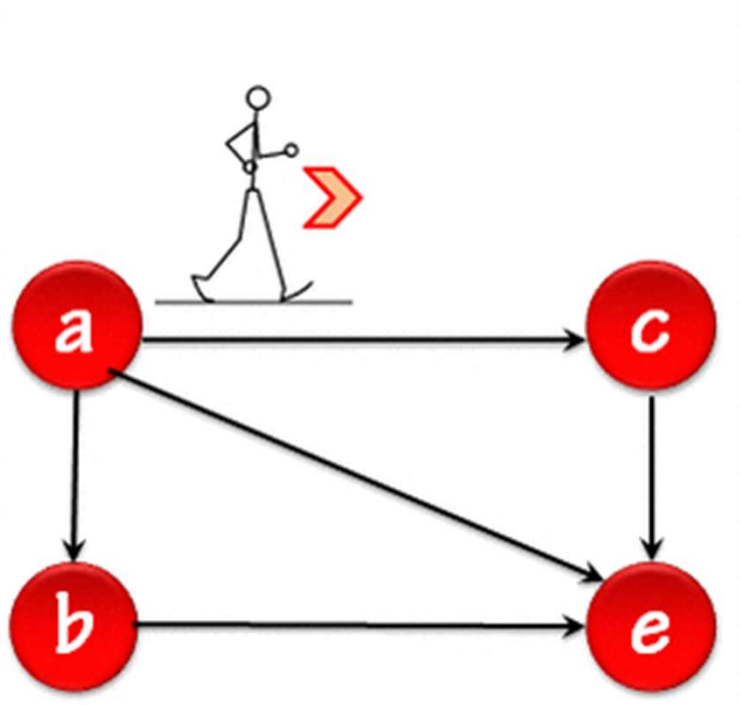
Dirigido

- *Las líneas tienen flecha*
- *Los arcos tienen sentido de dirección*

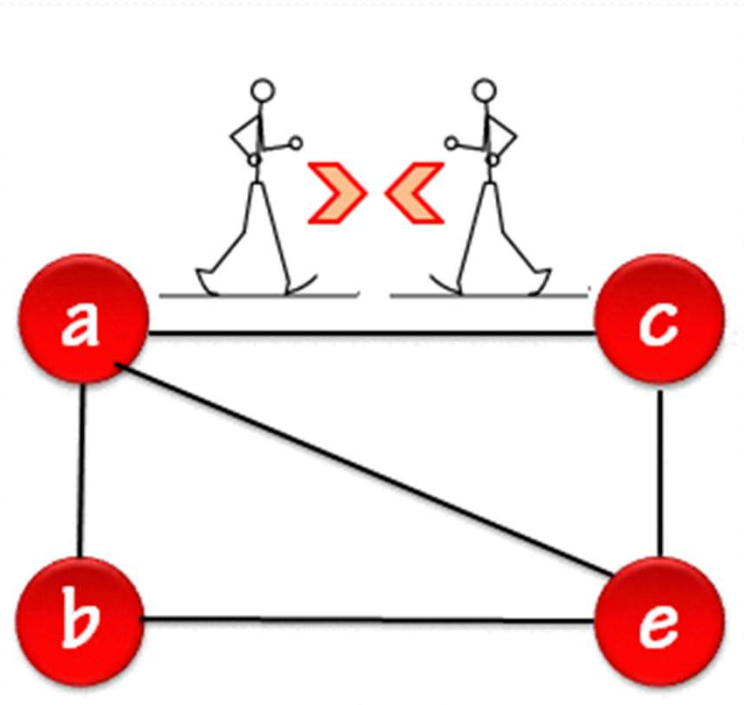
No dirigido

- *Las líneas NO tienen flecha*
- *Los arcos son bidireccionales*

Tipos de grafos

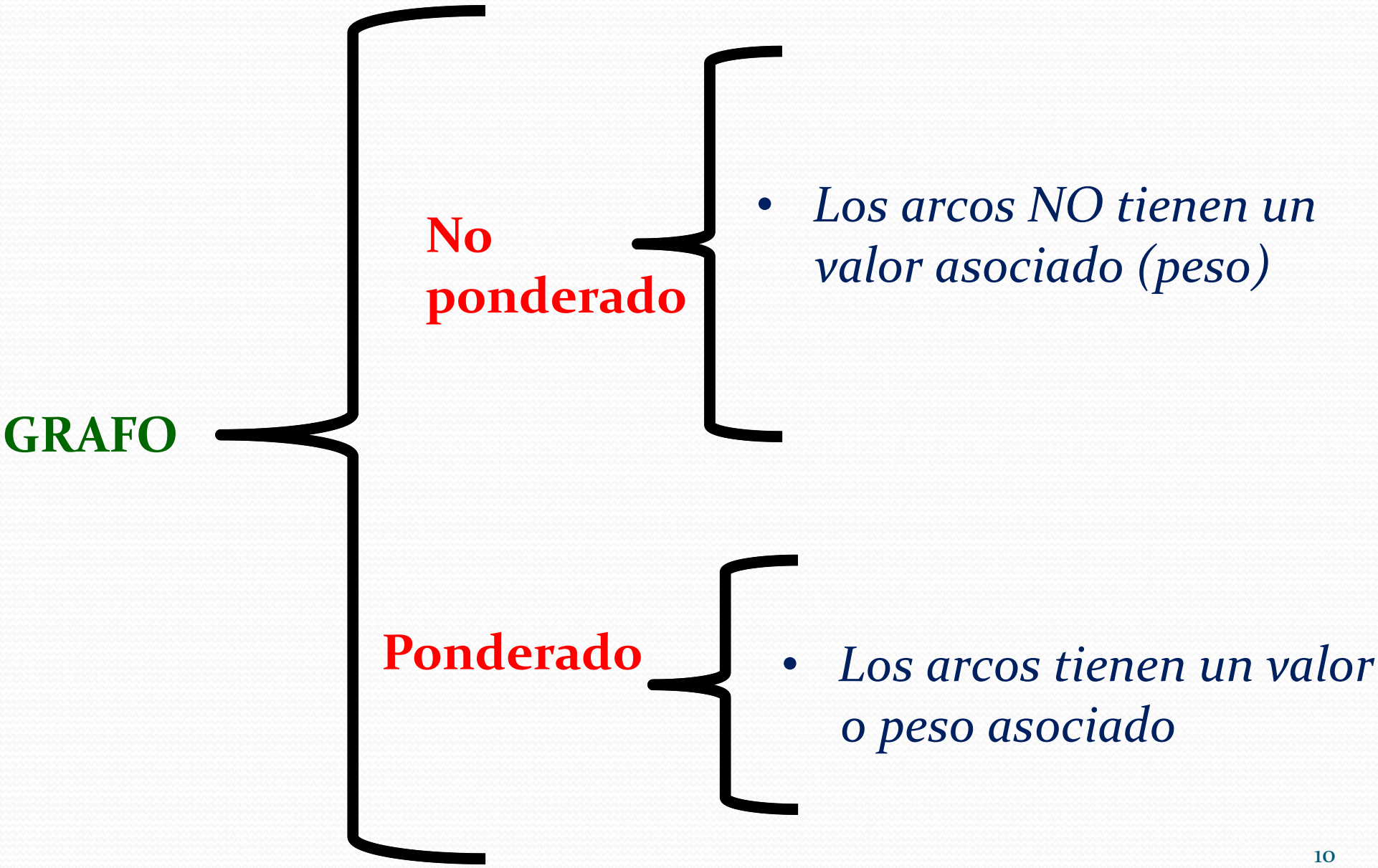


Grafo dirigido

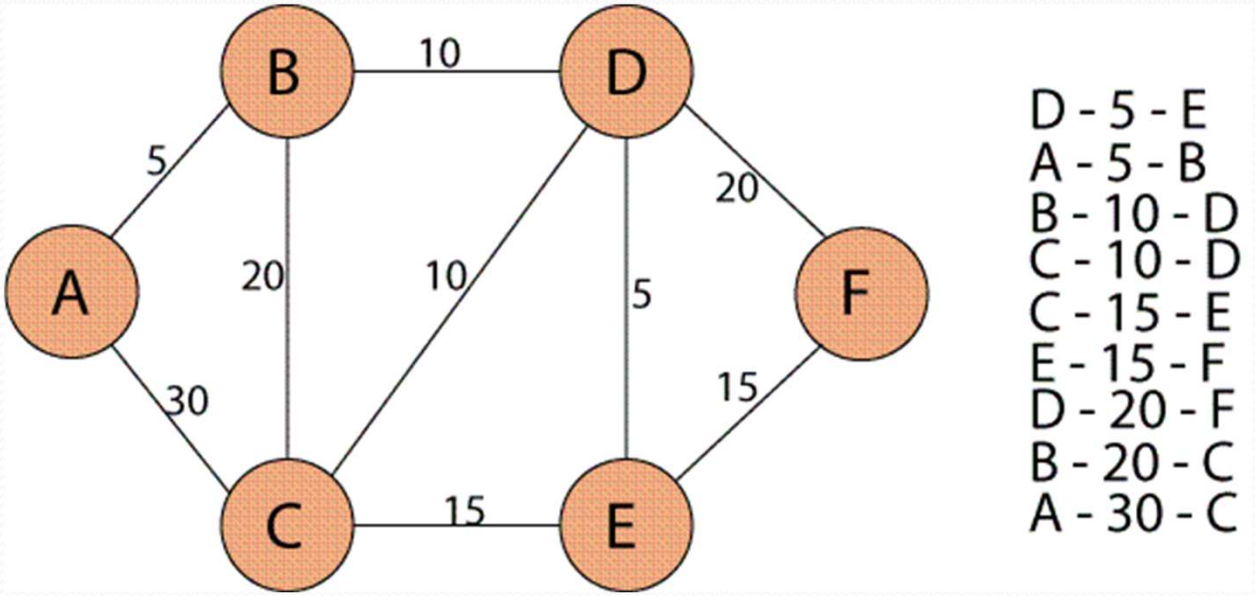


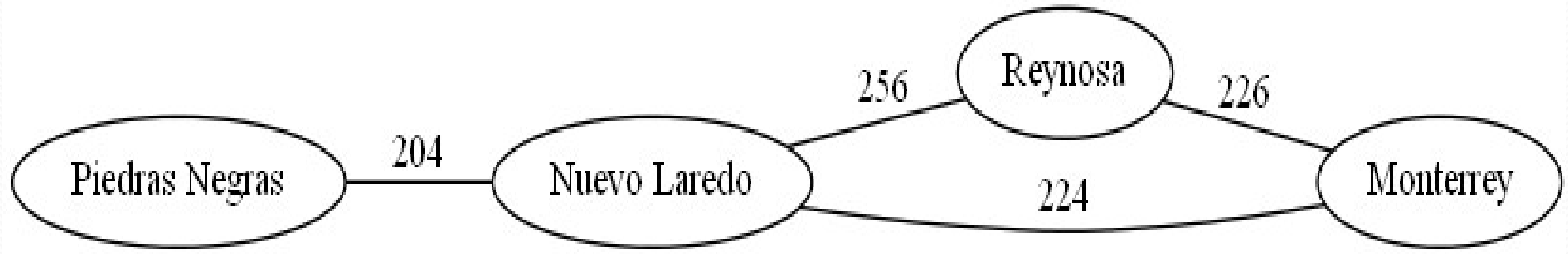
Grafo no dirigido

Tipos de grafos

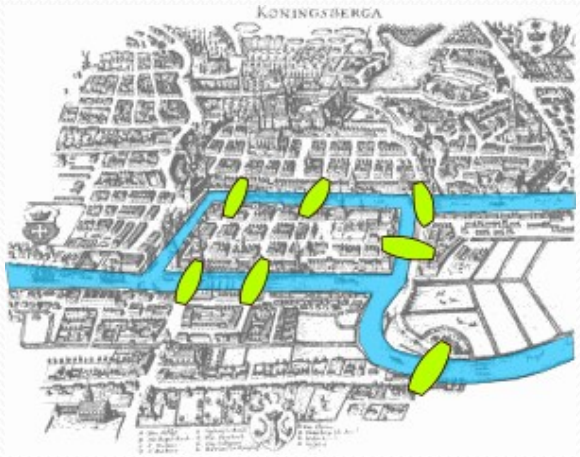


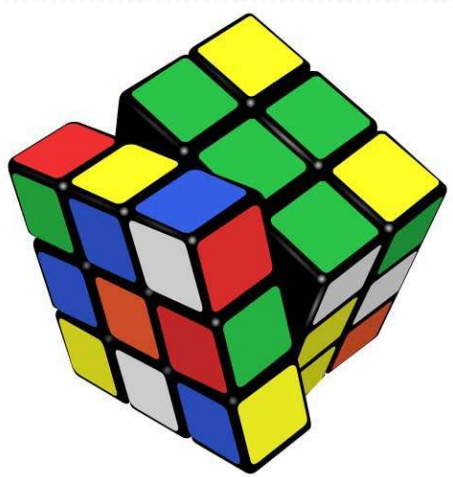
Ejemplos de grafos ponderados

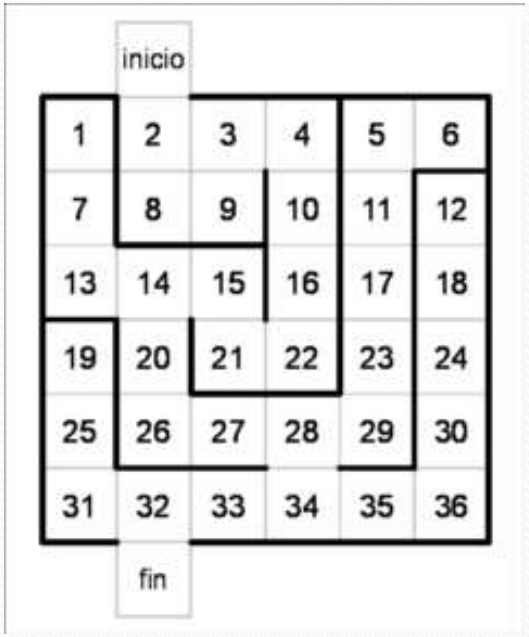




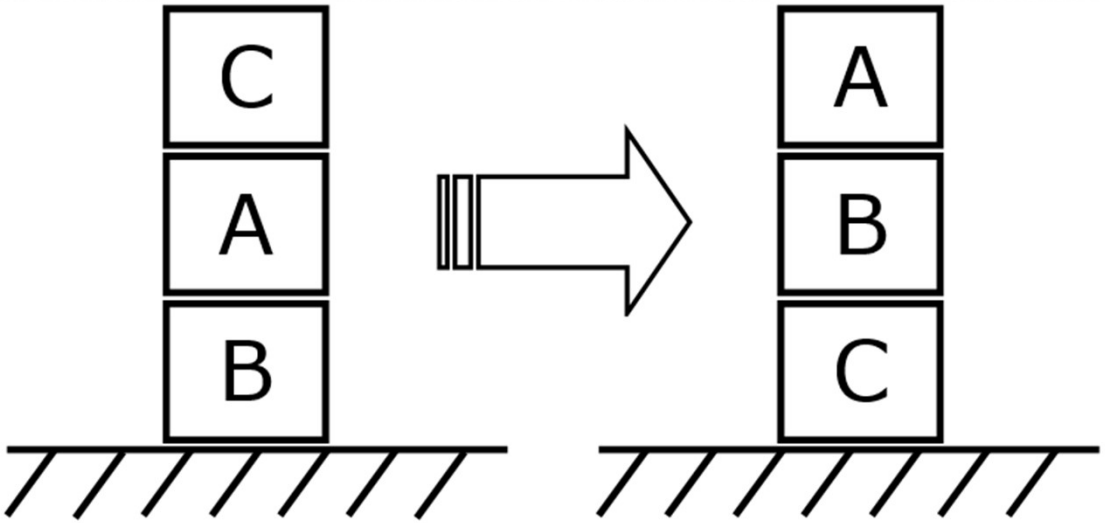
Aplicaciones con grafos

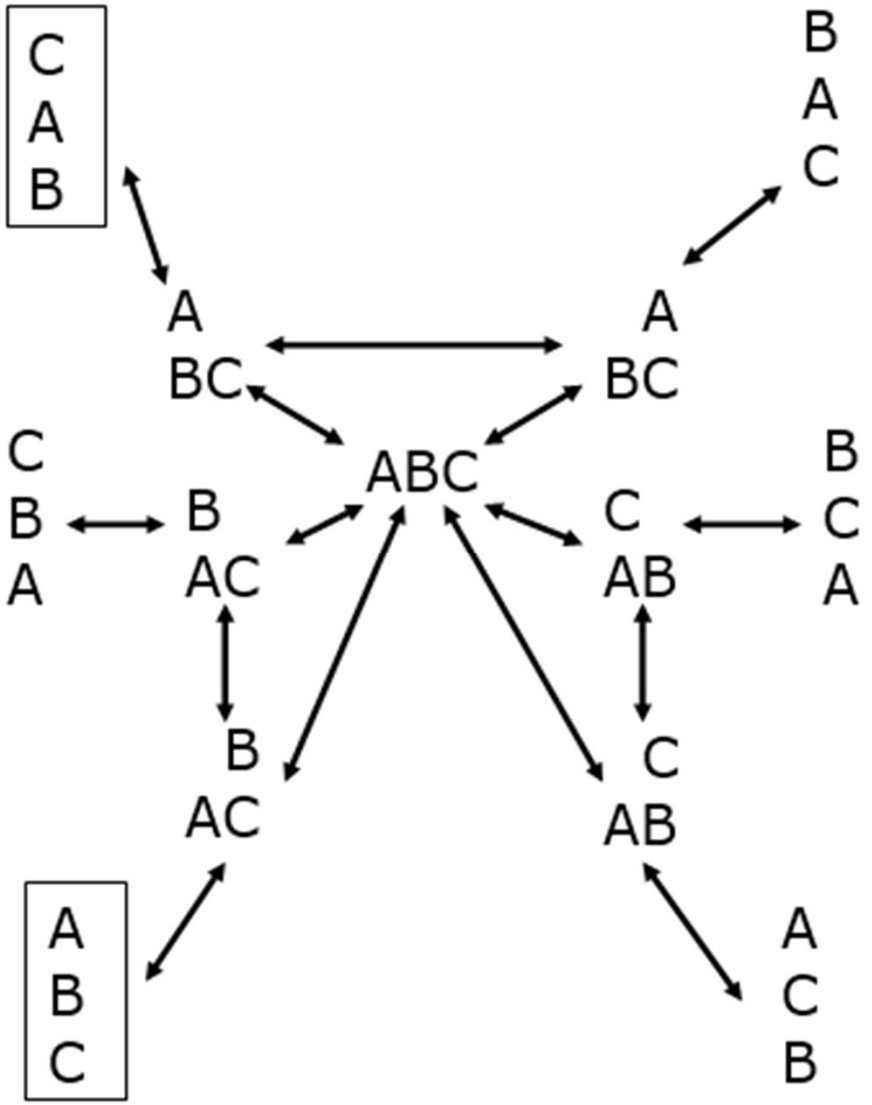




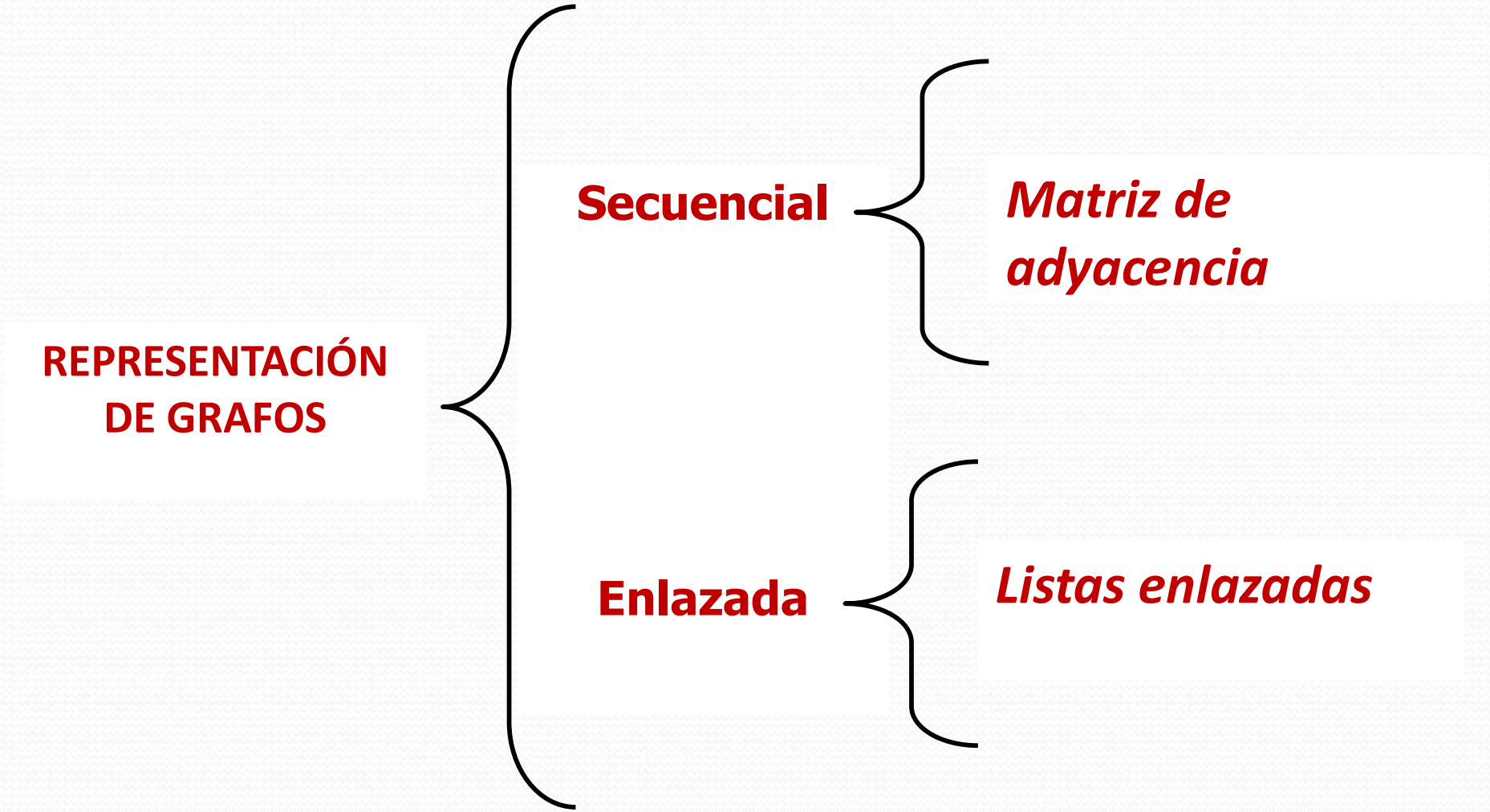


Ejemplos de grafos

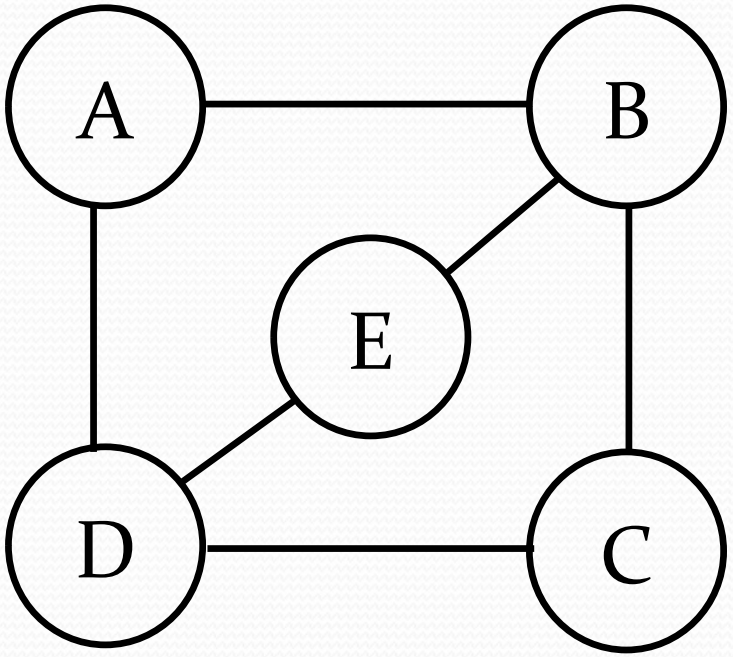




Representación de grafos

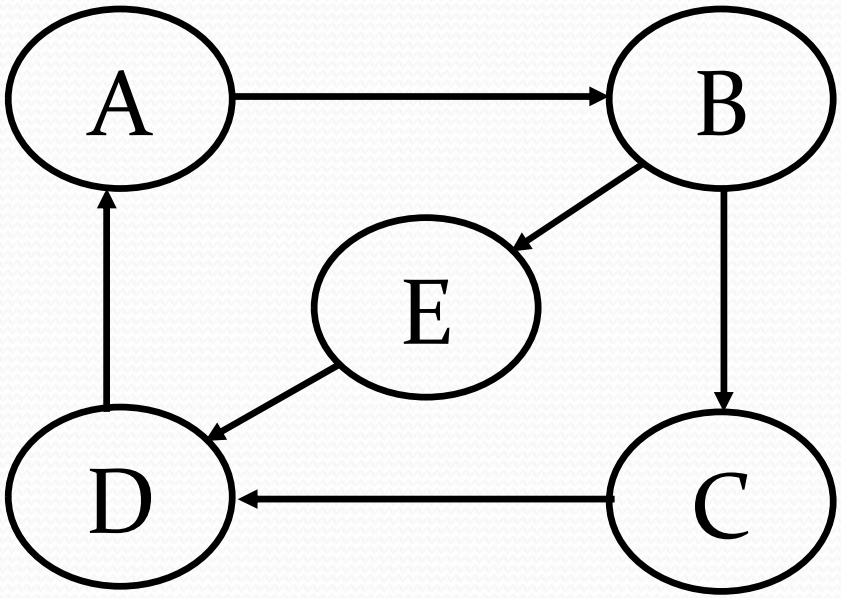


Matriz de adyacencia de un grafo no dirigido



Matriz	A	B	C	D	E
A	false	true	false	true	false
B	true	false	true	false	true
C	false	true	false	true	false
D	true	false	true	false	true
E	false	true	false	true	false

Matriz de adyacencia de un grafo dirigido



Matriz	A	B	C	D	E
A	false	true	false	false	false
B	false	false	true	false	true
C	false	false	false	true	false
D	true	false	false	false	false
E	false	false	false	true	false

Algoritmo de Warshall

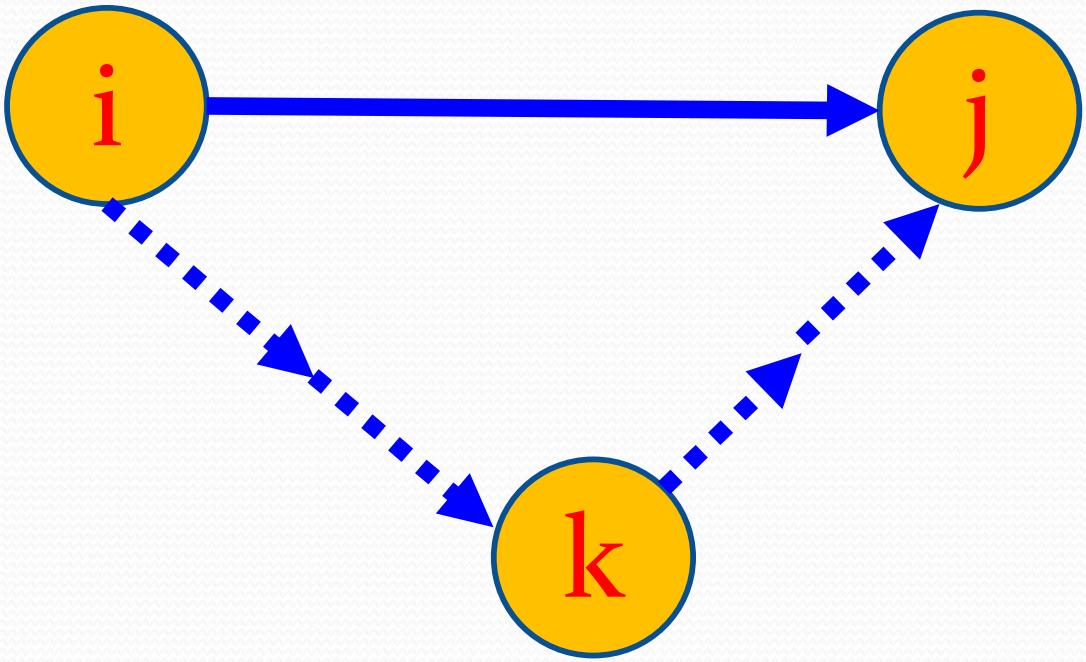
- *Sirve para determinar la existencia de un camino que conecte un nodo con otro*
- *Toma como base la matriz de adyacencia*

El Algoritmo de Warshall solamente determina si hay camino entre 2 nodos, pero **NO encuentra la ruta**

Interpretación del Algoritmo de Warshall

- *Hay camino entre el nodo i y el nodo j:*

1. *Hay arco directo*
2. *Hay nodos intermedios*



Algoritmo de Warshall

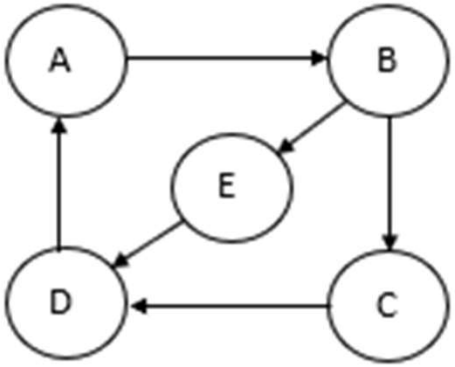
$$Matriz_k[i, j] = Matriz_{k-1}[i, j] \text{ OR } (Matriz_{k-1}[i, k] \text{ AND } Matriz_{k-1}[k, j])$$

- donde:
- Matriz: Arreglo bidimensional cuadrado de *n X n* (Matriz de adyacencia).
- i: Renglones de la matriz ($i = 0, 1, 2, \ldots, n-1$).
- j: Columnas de la matriz ($j = 0, 1, 2, \ldots, n-1$).
- k: Cantidad de matrices a calcular ($k = 0, 1, 2, \ldots, n-1$).
- n: Cantidad de nodos en el grafo.

Algoritmo de Warshall

```
Algoritmo_Warshall(): nulo

/* Algoritmo de Warshall para calcular la matriz de caminos */

1.- REPETIR CON k DESDE 0 HASTA n-1 CON INCREMENTO 1
    1.1. REPETIR CON i DESDE 0 HASTA n-1 CON INCREMENTO 1
         1.1.1. REPETIR CON j DESDE 0 HASTA n-1 CON INCREMENTO 1
                1.1.1.1 Matriz[i, j] = Matriz[i, j] OR (Matriz[i,k] AND Matriz[k, j])
         1.1.2. {FIN DEL CICLO DEL PASO 1.1.1}
    1.2. {FIN DEL CICLO DEL PASO 1.1}
2.- {FIN DEL CICLO DEL PASO 1}
```

Algoritmo de Warshall



Matriz	A	B	C	D	E
A	false	true	false	false	false
B	false	false	true	false	true
C	false	false	false	true	false
D	true	false	false	false	false
E	false	false	false	true	false

Matriz	A	B	C	D	E
A	false	true	false	false	false
B	false	false	true	false	true
C	false	false	false	true	false
D	true	**true**	false	false	false
E	false	false	false	true	false

a) $k=0$

Matriz	A	B	C	D	E
A	false	true	**true**	false	**true**
B	false	false	true	false	true
C	false	false	false	true	false
D	true	true	**true**	false	**true**
E	false	false	false	true	false

b) $k=1$

Matriz	A	B	C	D	E
A	false	true	true	**true**	true
B	false	false	true	**true**	true
C	false	false	false	true	false
D	true	true	true	**true**	true
E	false	false	false	true	false

d) $k=2$

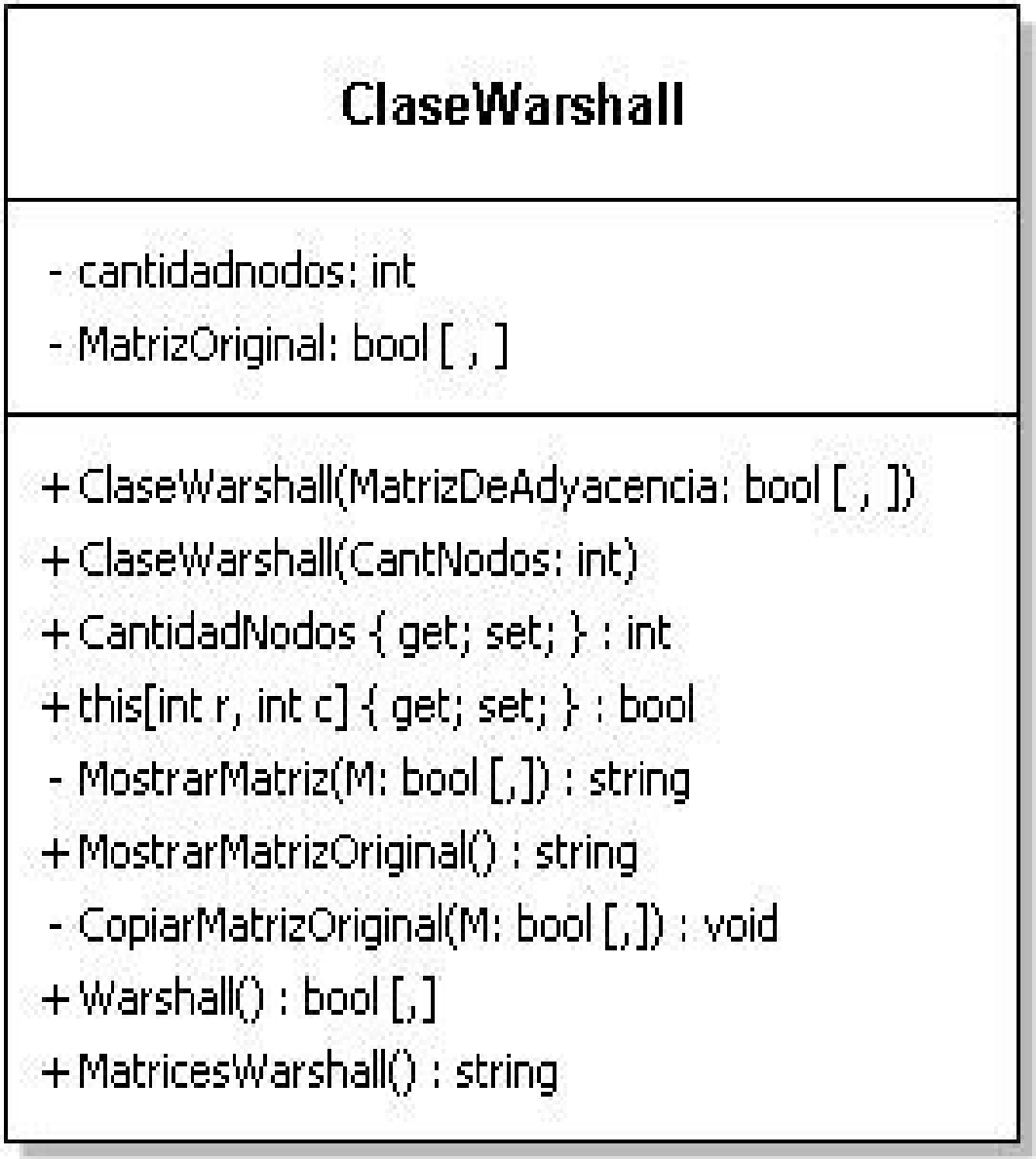
Matriz	A	B	C	D	E
A	**true**	true	true	true	true
B	**true**	**true**	true	true	true
C	**true**	**true**	**true**	true	**true**
D	true	true	true	true	true
E	**true**	**true**	**true**	true	**true**

c) $k=3$

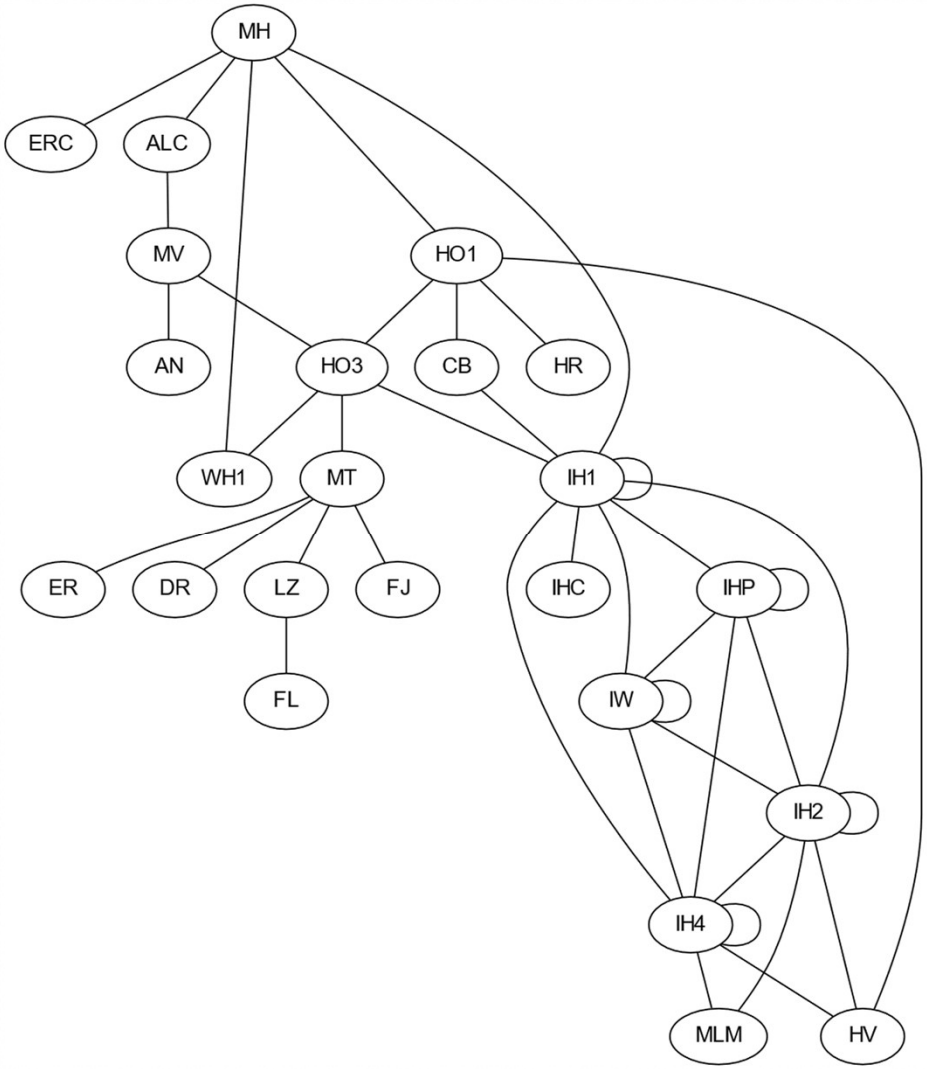
Matriz	A	B	C	D	E
A	true	true	true	true	true
B	true	true	true	true	true
C	true	true	True	true	true
D	true	true	True	true	true
E	true	true	True	true	true

a) $k=4$

Clase para el Algoritmo de Warshall



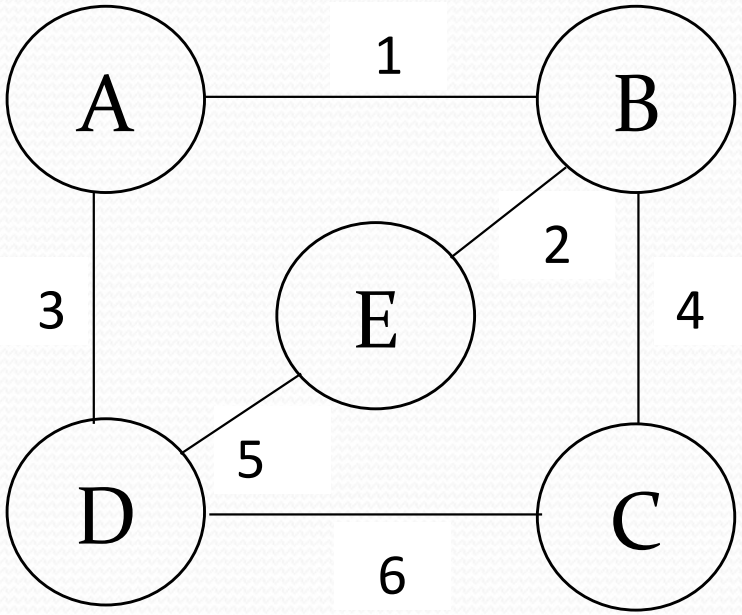
Demo



Algoritmo de Warshall

Prog.9.1.- Algoritmo de Warshall Consola

Matriz de pesos para un grafo ponderado



Matriz	A	B	C	D	E
A	∞	1	∞	3	∞
B	1	∞	4	∞	2
C	∞	4	∞	6	∞
D	3	∞	6	∞	5
E	∞	2	∞	5	∞

Algoritmo de Floyd

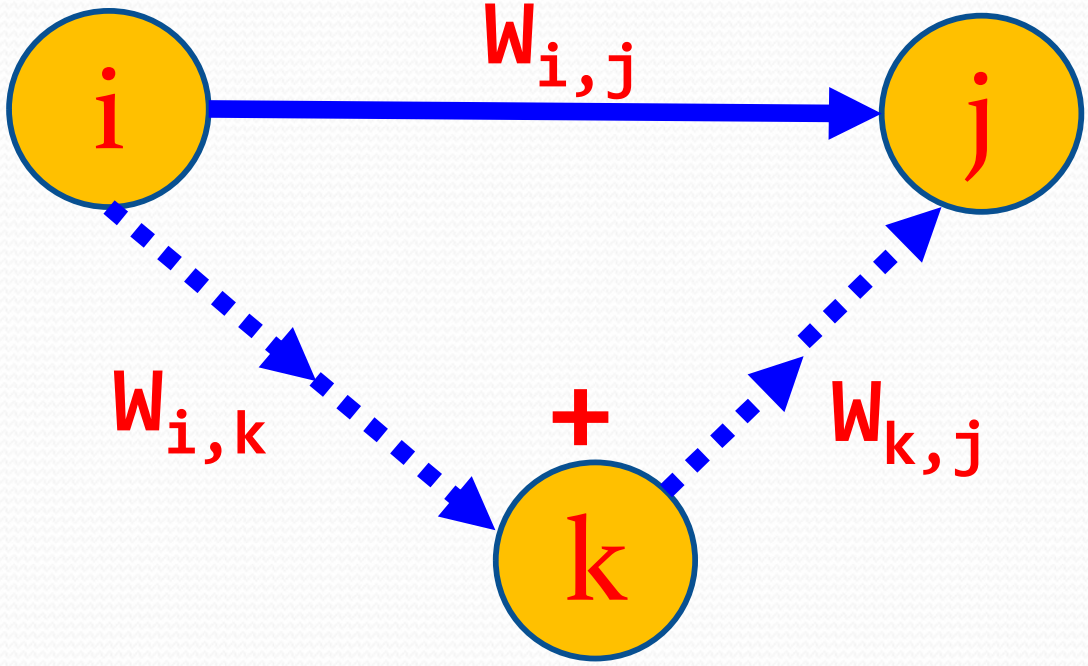
- *Sirve para determinar el valor del camino más corto entre 2 nodos*
- *Toma como base la matriz de pesos*

El Algoritmo de Floyd solamente calcula el **valor** del camino más corto entre 2 nodos, pero **NO encuentra la ruta**

Interpretación del Algoritmo de Floyd

- *Calcula la longitud (peso) del camino entre el nodo i y el nodo j:*

1. *Peso del arco directo*
2. *Suma de pesos de los nodos intermedios*



Algoritmo de Floyd

$$\text{Matriz}_k[i, j] = \text{MINIMO} \begin{cases} \text{Matriz}_{k-1}[i, j] \\ \text{Matriz}_{k-1}[i, k] + \text{Matriz}_{k-1}[k, j] \end{cases}$$

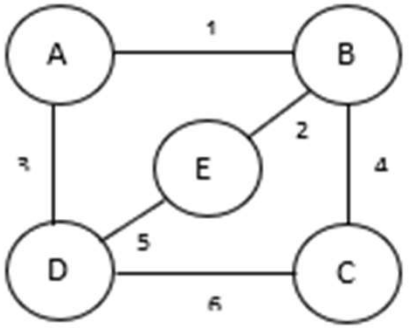
donde:

- `Matriz`: Arreglo bidimensional cuadrado de *n X n* (Matriz de pesos).
- `i`: Renglones de la matriz (*i = 0, 1, 2, …, n-1*).
- `j`: Columnas de la matriz (*j = 0, 1, 2, …, n-1*).
- `k`: Cantidad de matrices a calcular (*k = 0, 1, 2, …, n-1*).
- `n`: Cantidad de nodos en el grafo.

Algoritmo de Floyd

```
Algoritmo_Floyd(): nulo

/* Algoritmo de Floyd para calcular la matriz del camino más corto */

1.- REPETIR CON k DESDE 0 HASTA n-1 CON INCREMENTO 1
    1.1. REPETIR CON i DESDE 0 HASTA n-1 CON INCREMENTO 1
         1.1.1. REPETIR CON j DESDE 0 HASTA n-1 CON INCREMENTO 1
                1.1.1.1 SI Matriz[i,k]+Matriz[k,j] < Matriz[i,j] ENTONCES
                    1.1.1.1.1. Matriz[i,j]=Matriz[i,k]+Matriz[k, j]
                1.1.1.2. {FIN DE LA CONDICIONAL DEL PASO 1.1.1.1}
         1.1.2. {FIN DEL CICLO DEL PASO 1.1.1}
    1.2. {FIN DEL CICLO DEL PASO 1.1}
2.- {FIN DEL CICLO DEL PASO 1}
```

Algoritmo de Floyd



Matriz	A	B	C	D	E
A	∞	1	∞	3	∞
B	1	∞	4	∞	2
C	∞	4	∞	6	∞
D	3	∞	6	∞	5
E	∞	2	∞	5	∞

Matriz	A	B	C	D	E
A	–	1	–	3	–
B	1	**2**	4	**4**	2
C	–	4	–	6	–
D	3	**4**	6	**6**	5
E	–	2	–	5	–

e) *k=0*

Matriz	A	B	C	D	E
A	**2**	1	**5**	3	**3**
B	1	2	4	4	2
C	**5**	4	**8**	6	**6**
D	3	4	6	6	5
E	**3**	2	**6**	5	**4**

f) *k=1*

Matriz	A	B	C	D	E
A	2	1	5	3	3
B	1	2	4	4	2
C	5	4	8	6	6
D	3	4	6	6	5
E	3	2	6	5	4

h) *k=2*

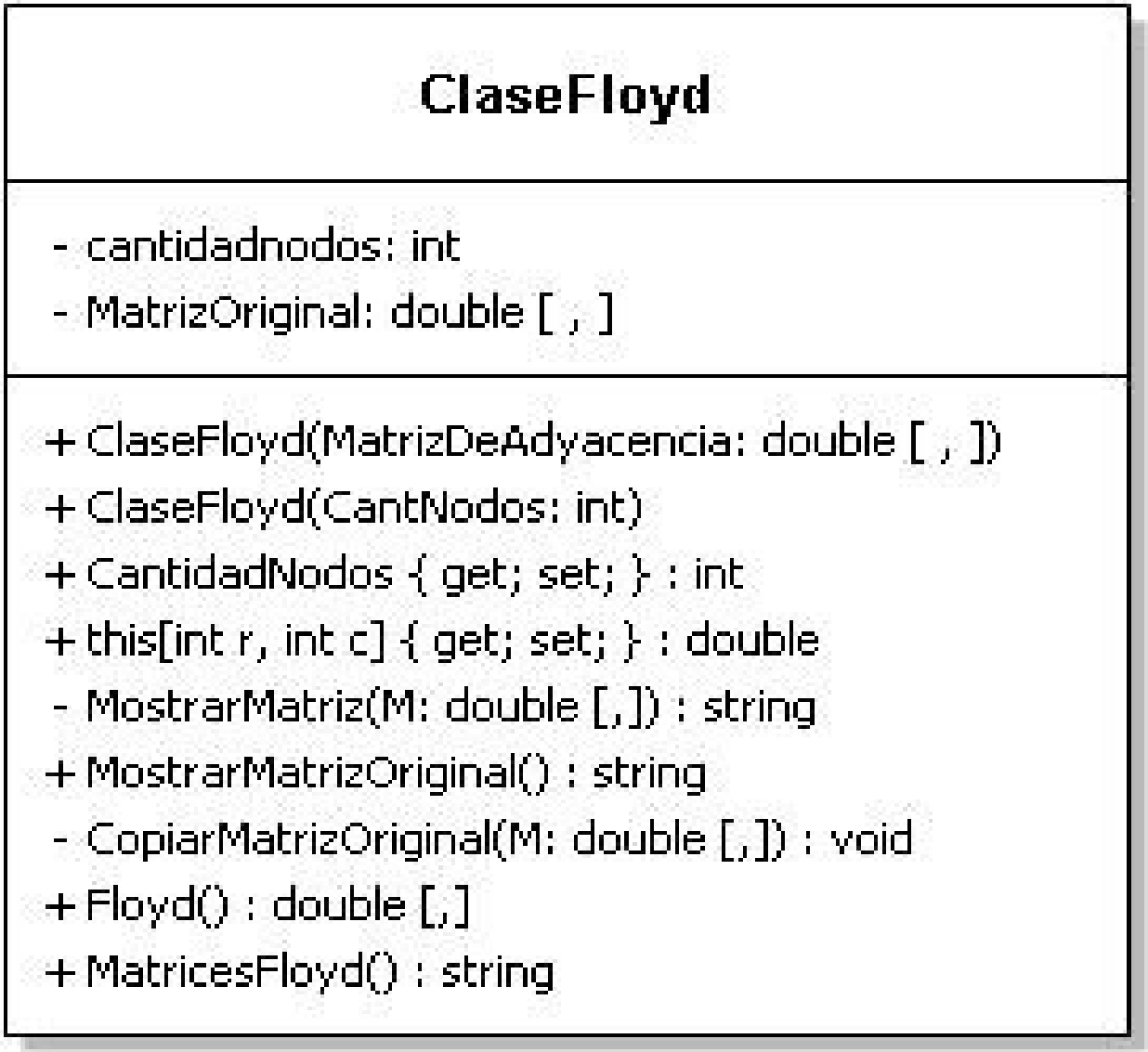
Matriz	A	B	C	D	E
A	2	1	5	3	3
B	1	2	4	4	2
C	5	4	8	6	6
D	3	4	6	6	5
E	3	2	6	5	4

g) *k=3*

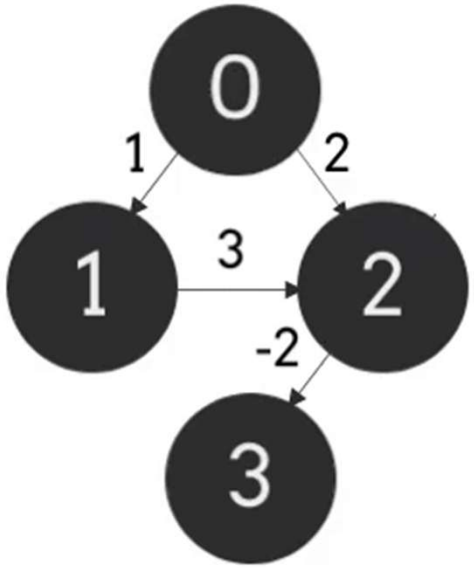
Matriz	A	B	C	D	E
A	2	1	5	3	3
B	1	2	4	4	2
C	5	4	8	6	6
D	3	4	6	6	5
E	3	2	6	5	4

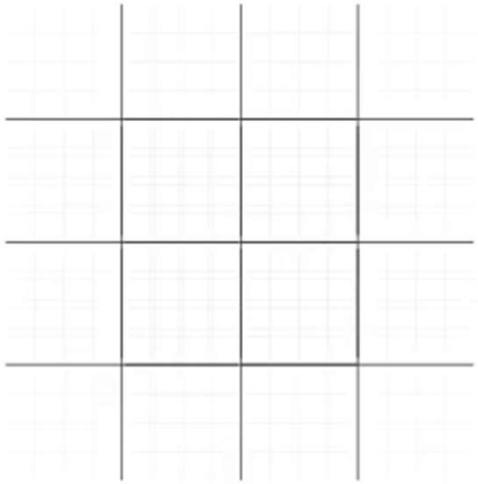
b) *k=4*

Clase para el Algoritmo de Floyd



Demo

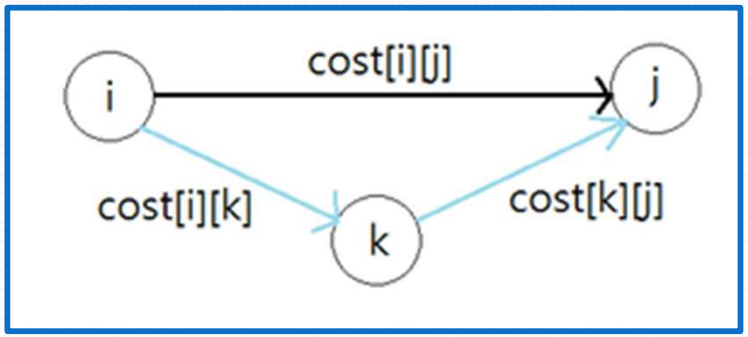




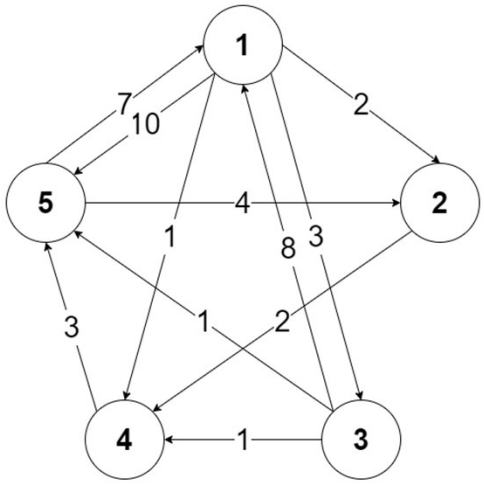
minDistance

Algoritmo de Floyd

Prog.9.2.- Algoritmo de Floyd Consola



Ejercicios



	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	INF	0	1	1
4	INF	INF	INF	0	3
5	7	4	INF	INF	0

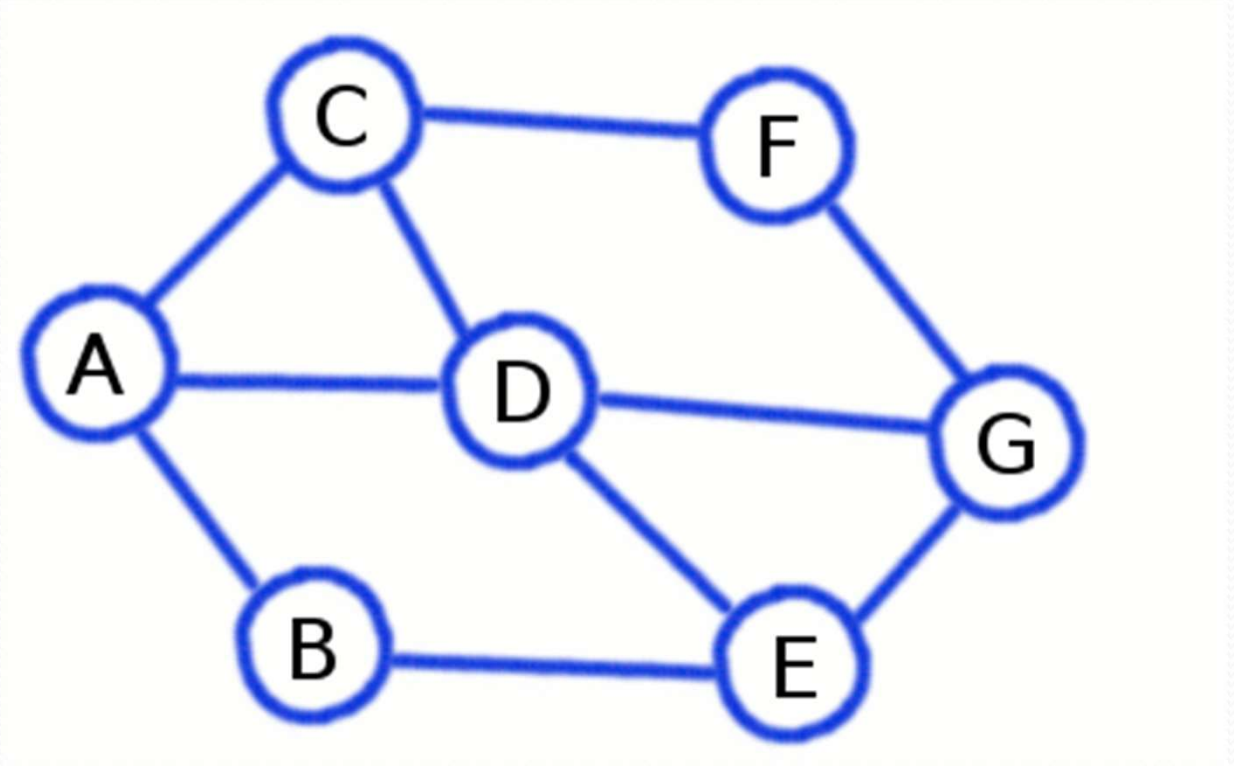
Algoritmo de Warshall

Algoritmo de Floyd

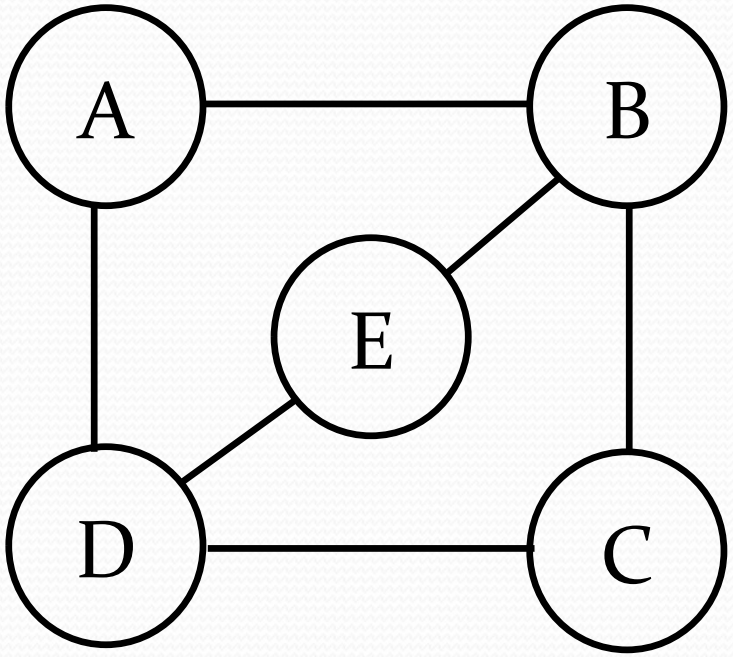
Represente un grafo mediante matrices de adyacencia según corresponda

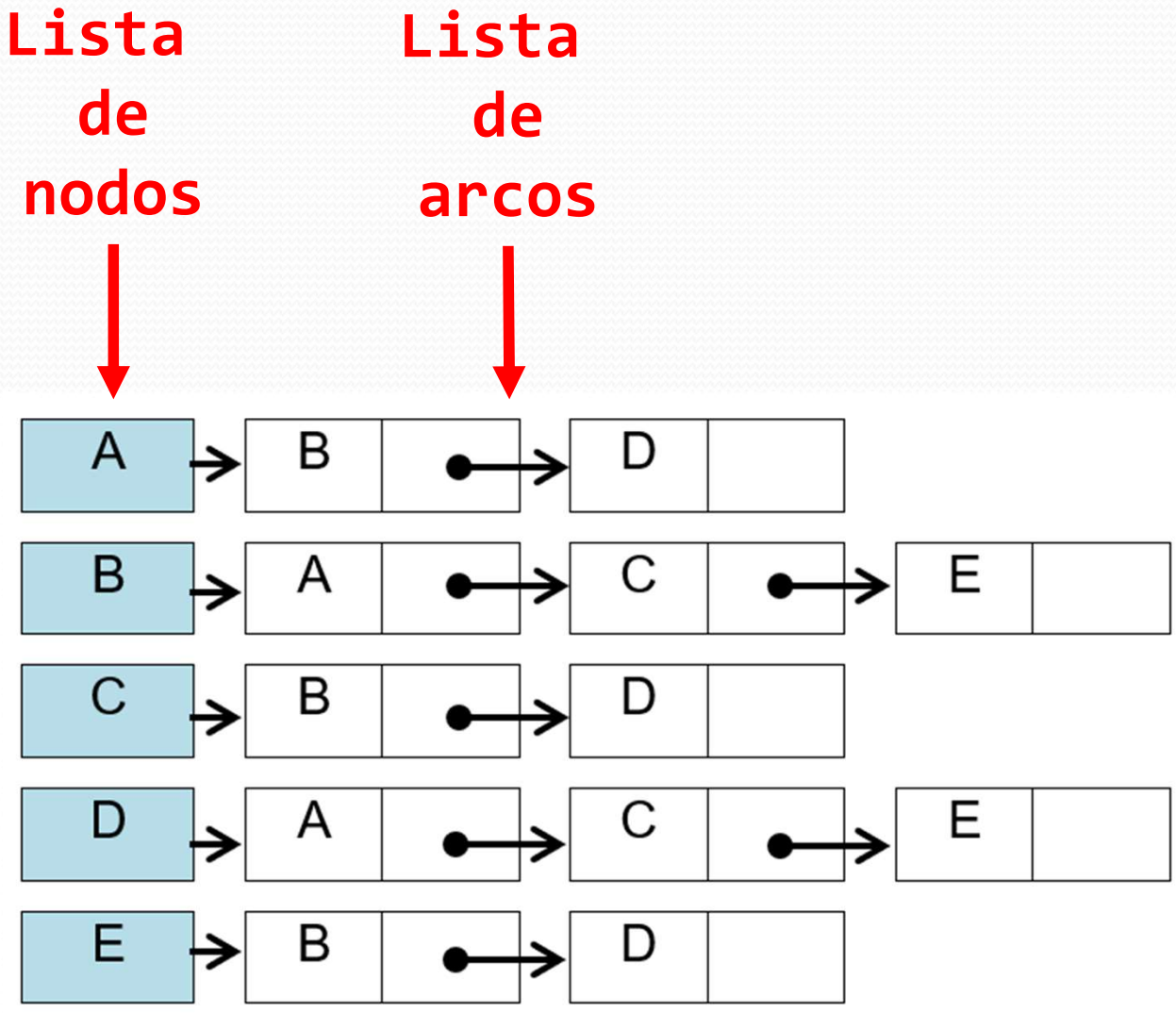
SESIÓN 2

Representación enlazada de grafos



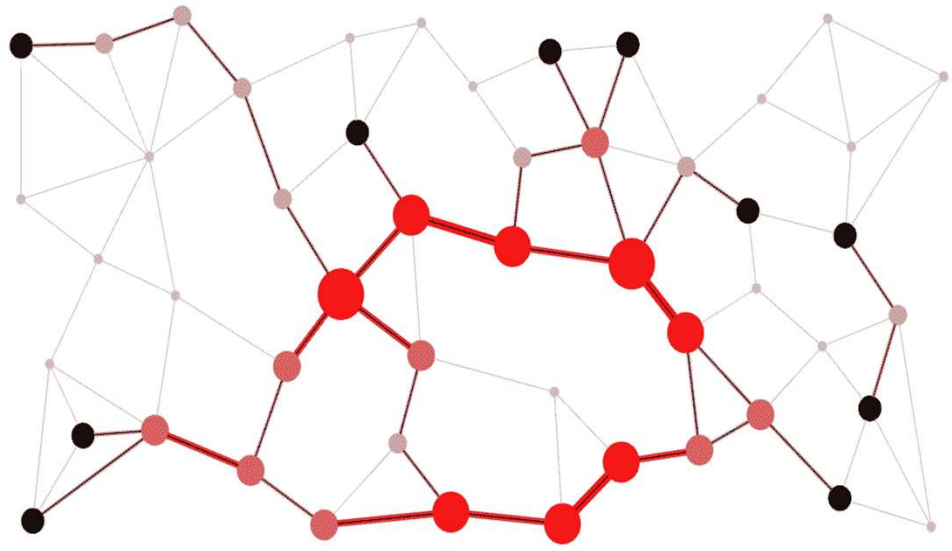
Representación enlazada de grafos mediante listas simples



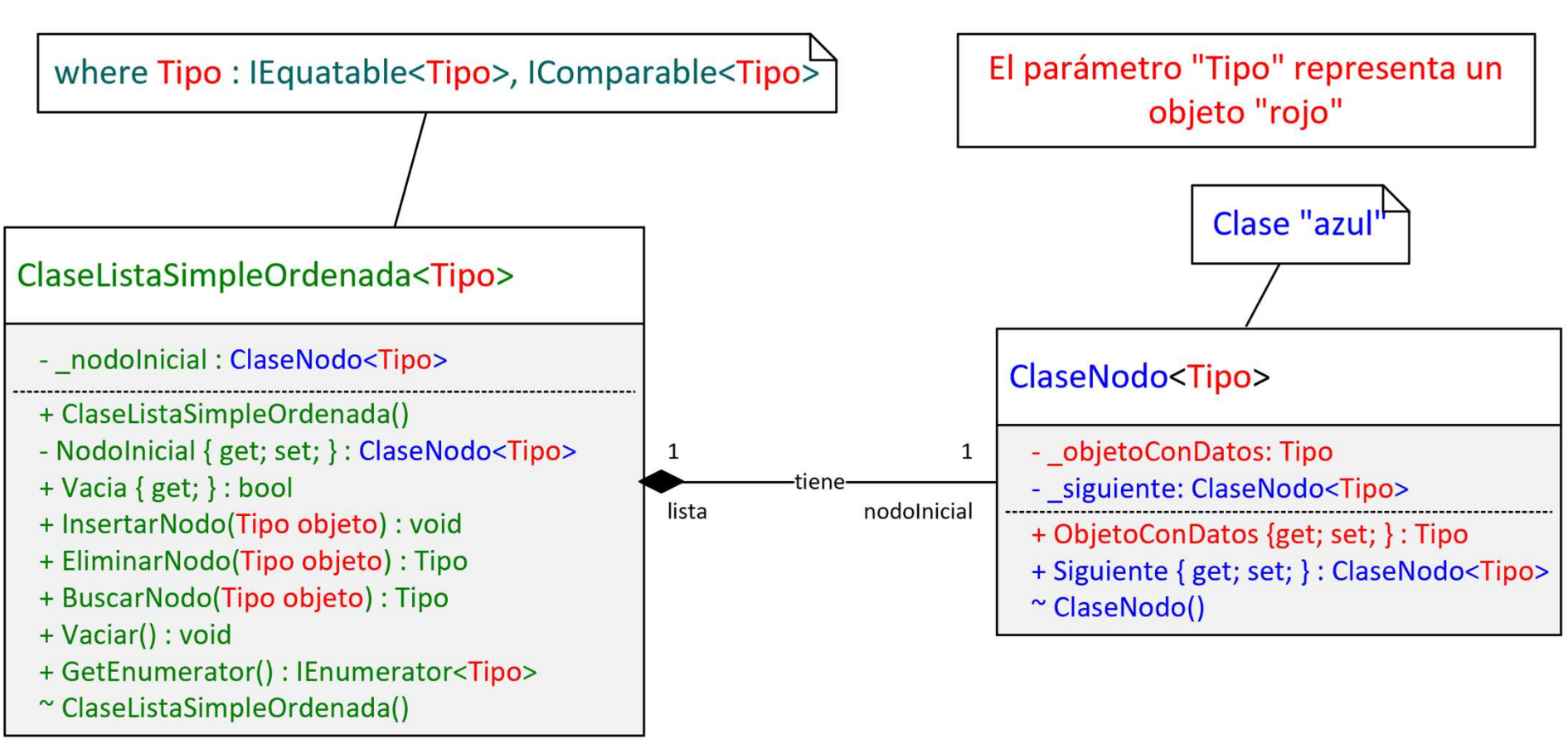


Operaciones en un grafo

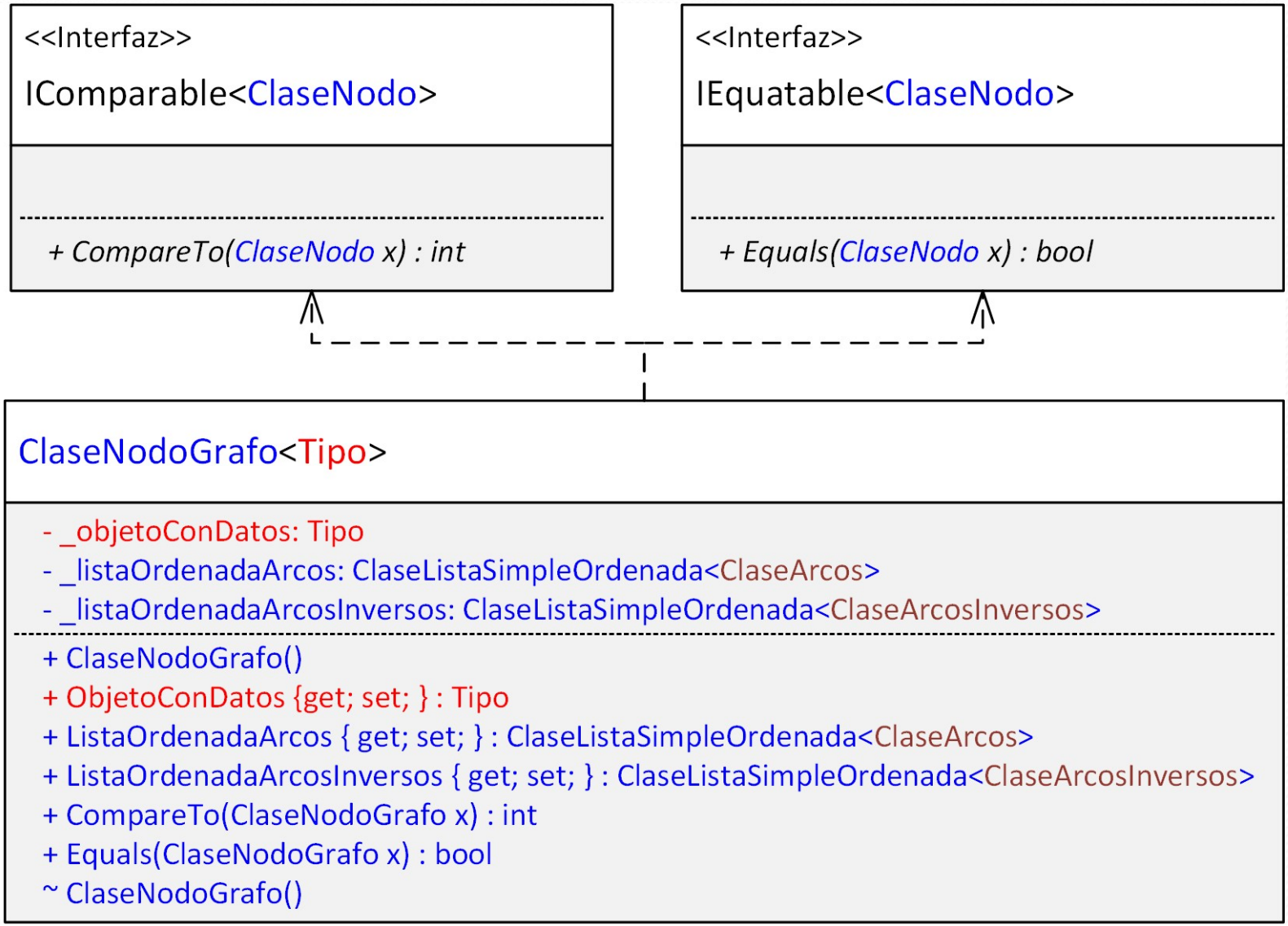
- *Insertar nodo*
- *Insertar arco*
- *Eliminar arco*
- *Eliminar nodo*
- *Búsqueda (recorrido)*



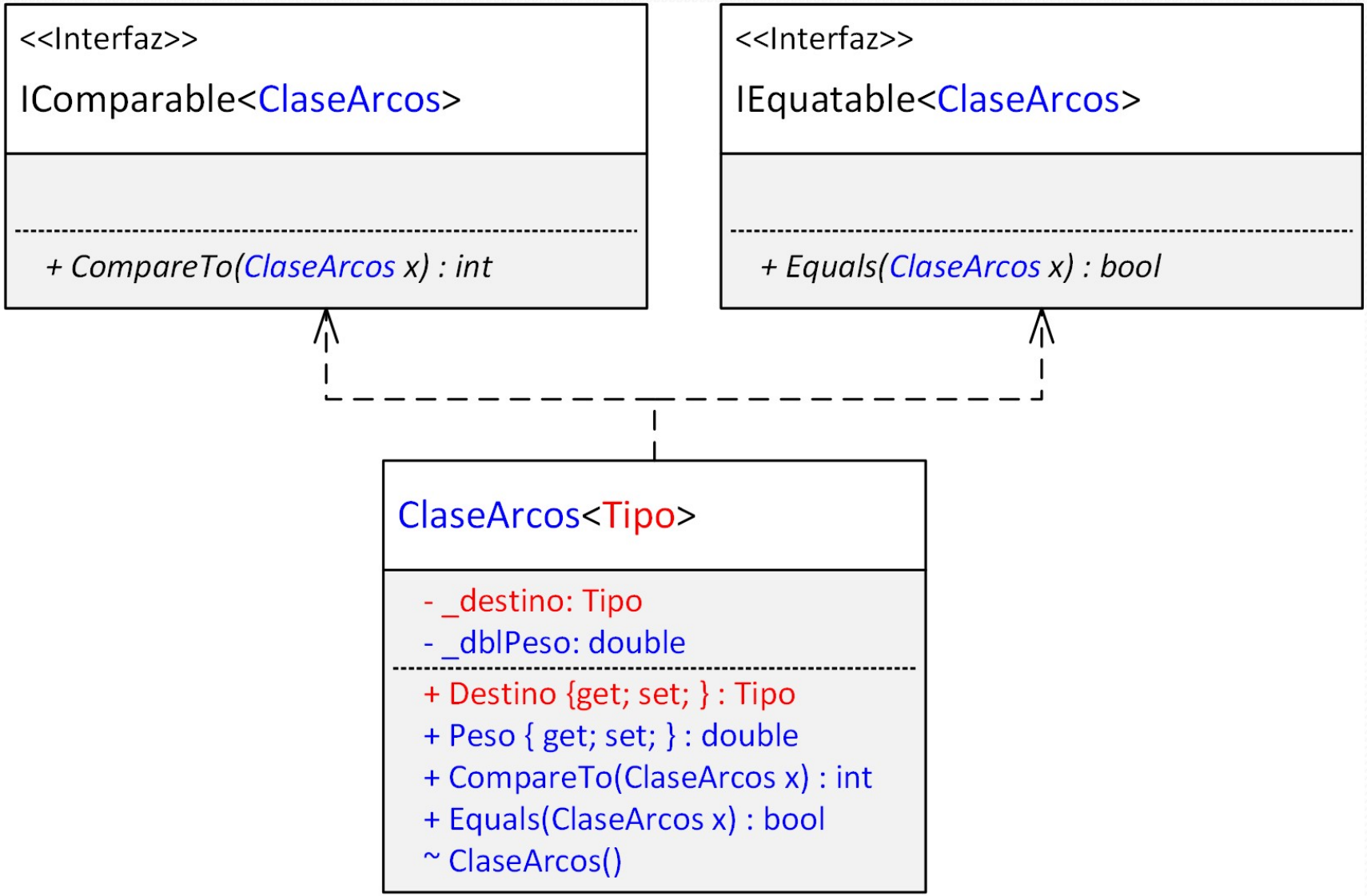
Clases para administrar las listas de nodos y de arcos



Clase para administrar los nodos del grafo

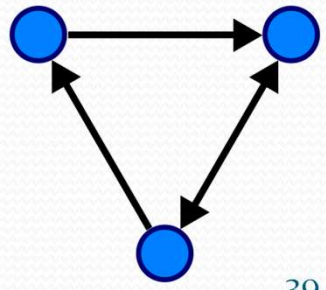


Clase para administrar los arcos del grafo



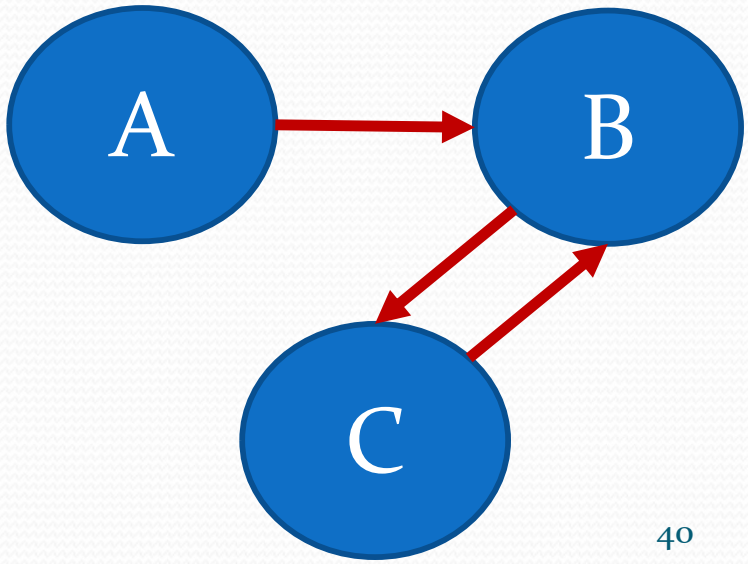
¿Para qué sirven los arcos inversos?

- *Además de la lista de arcos que "salen" de un nodo, también se debe administrar la lista de arcos que "entran" a él*
- *Los arcos inversos son útiles al eliminar un nodo*
- *Ya que se deben eliminar todos los arcos que "entran" al nodo eliminado*

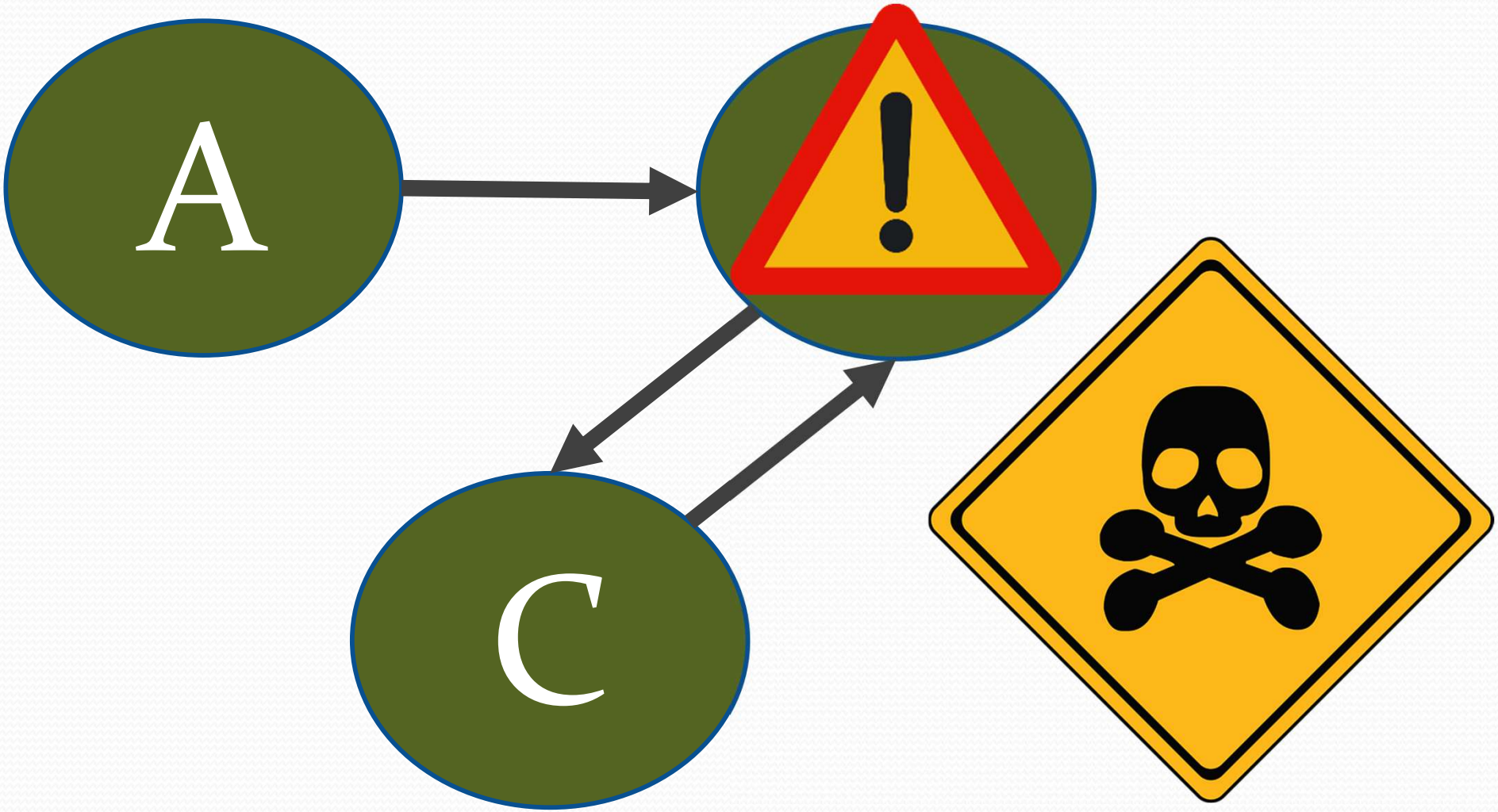


Uso de los arcos inversos

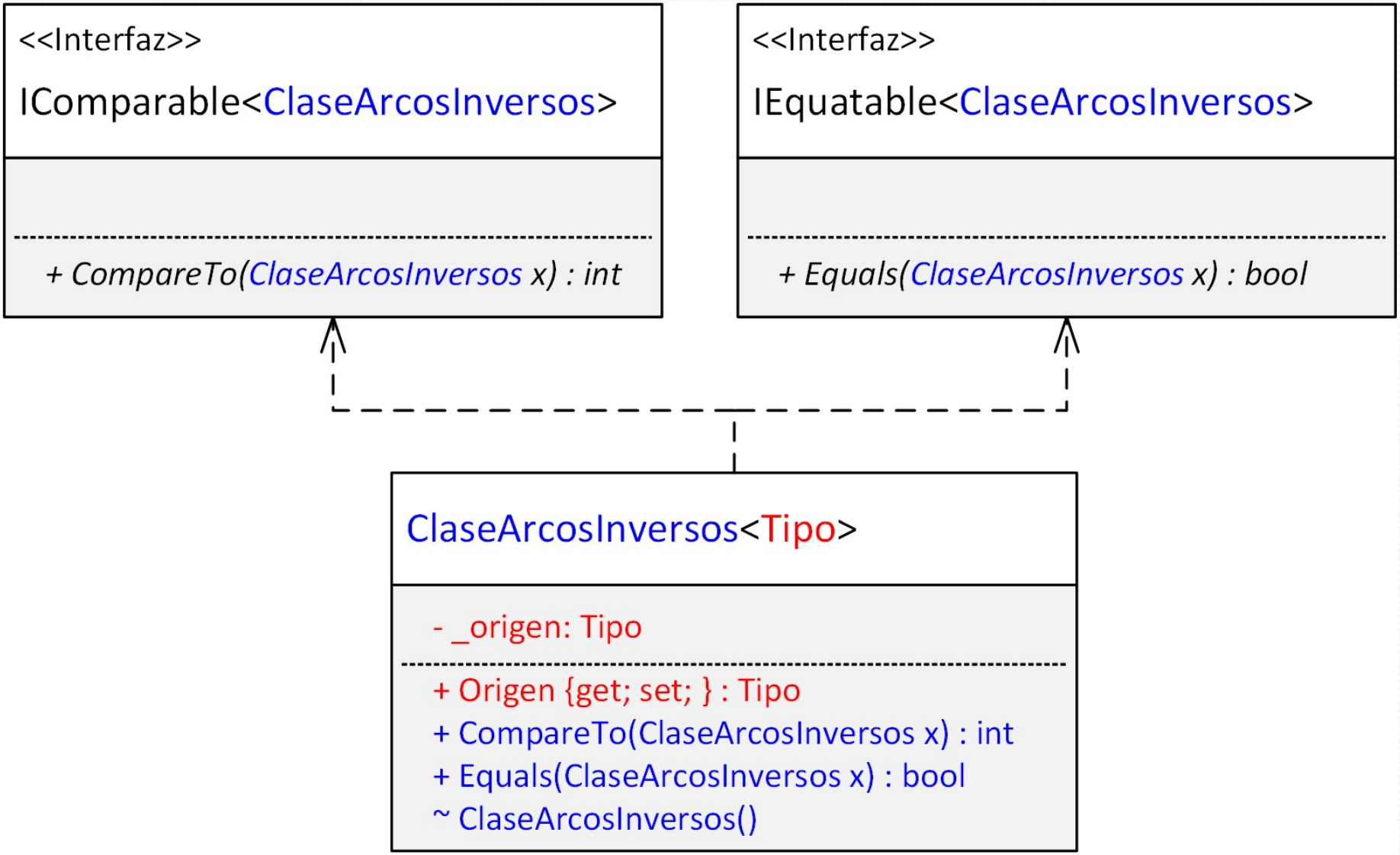
- *¿Qué pasaría si se elimina un nodo del grafo?*
- *Automáticamente se eliminan los arcos que "salen" de él*
- *Pero, ¿qué pasaría con los arcos que "entran" a ese nodo y provienen de otros nodos?*
- *¿Cómo saber de dónde provienen los arcos que entran al nodo eliminado?*



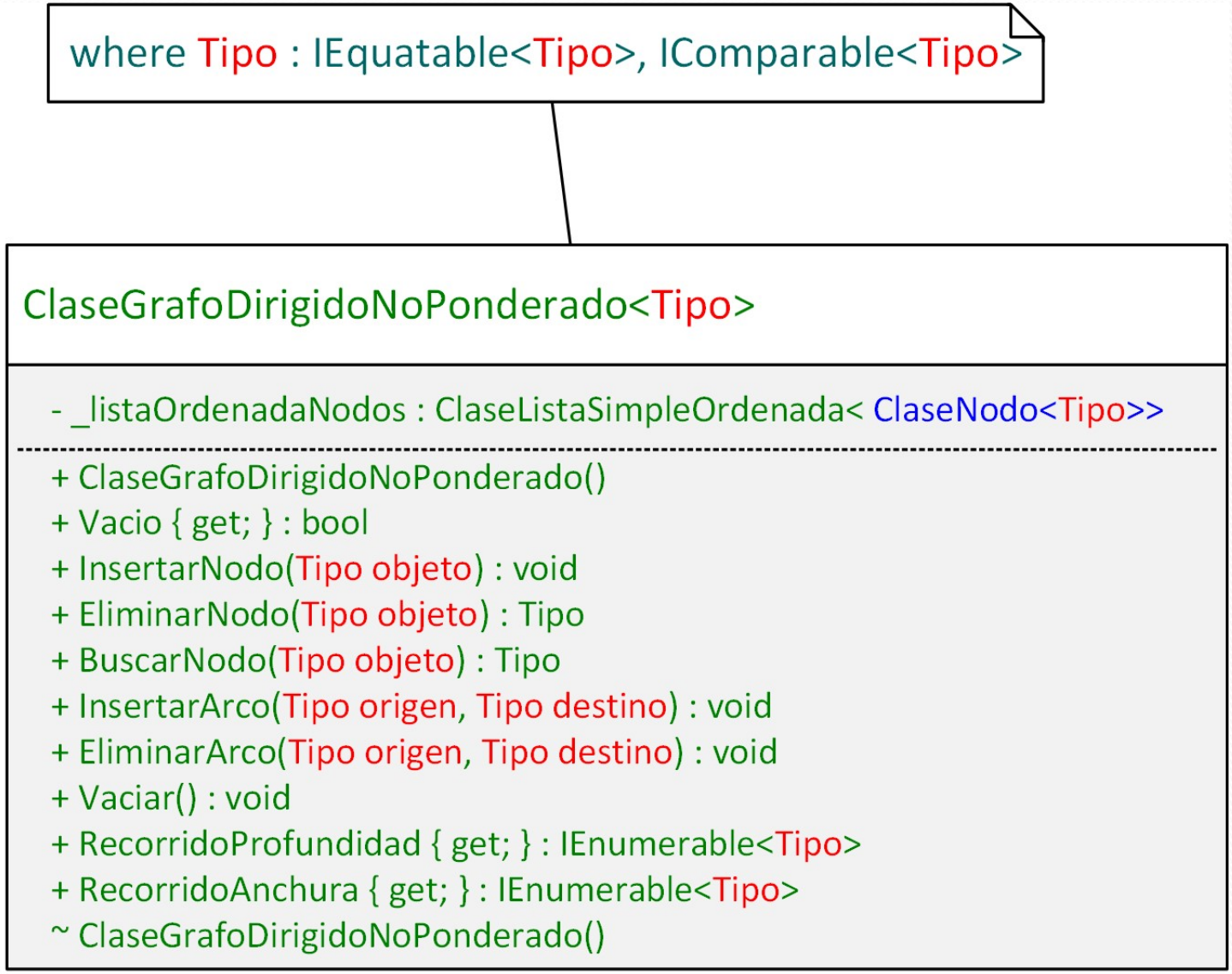
Uso de los arcos inversos



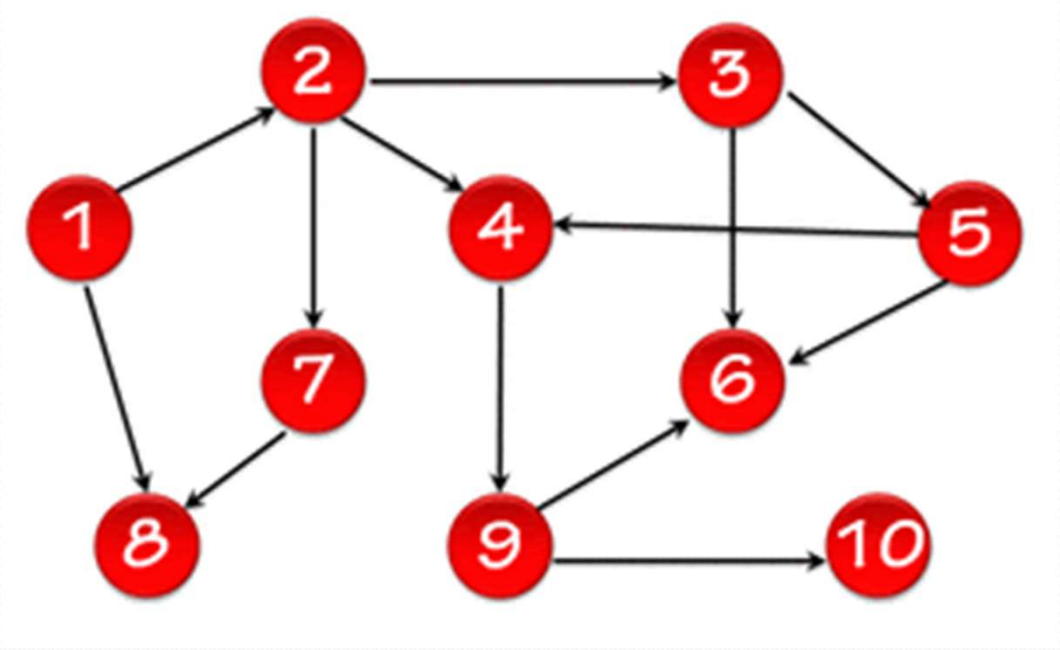
Clase para administrar los arcos inversos del grafo



Clase para un grafo dirigido y no ponderado



Demo

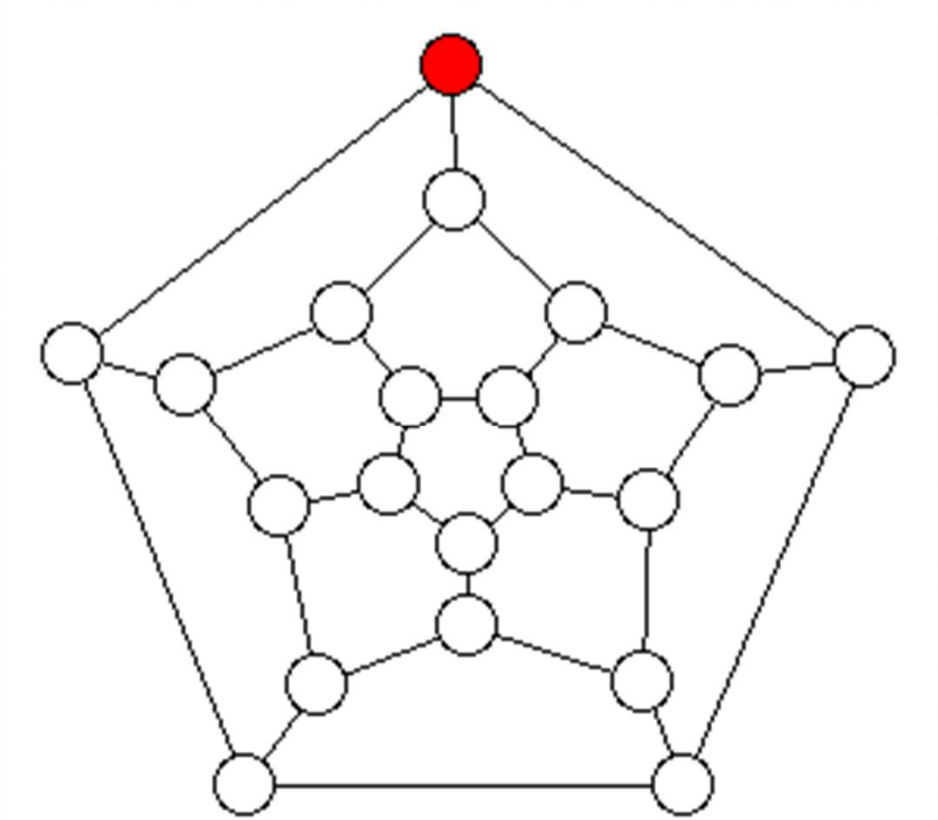


Grafo Dirigido y No Ponderado

Prog.9.3.- Grafo Dirigido y No Ponderado Consola

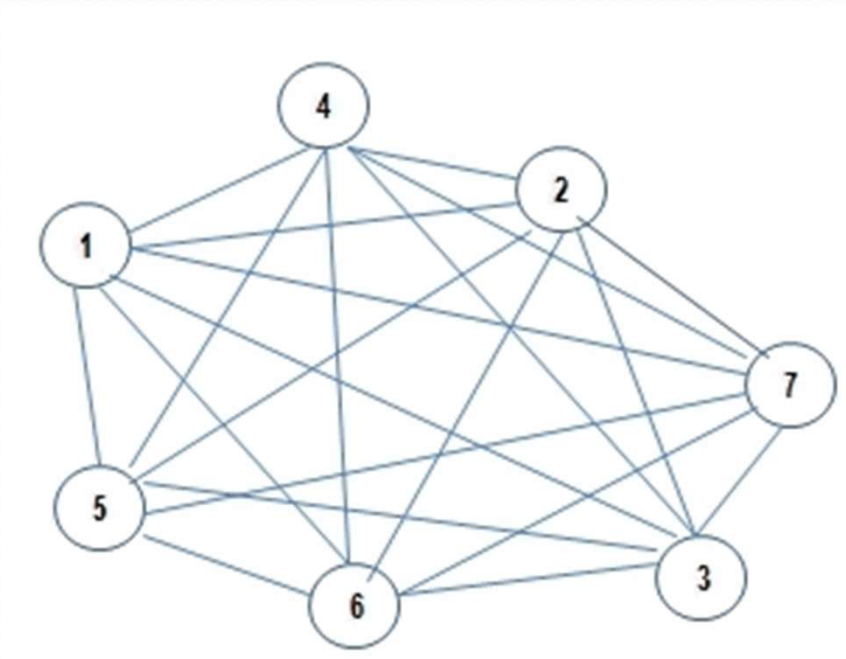
SESIÓN 3

Recorridos en grafos



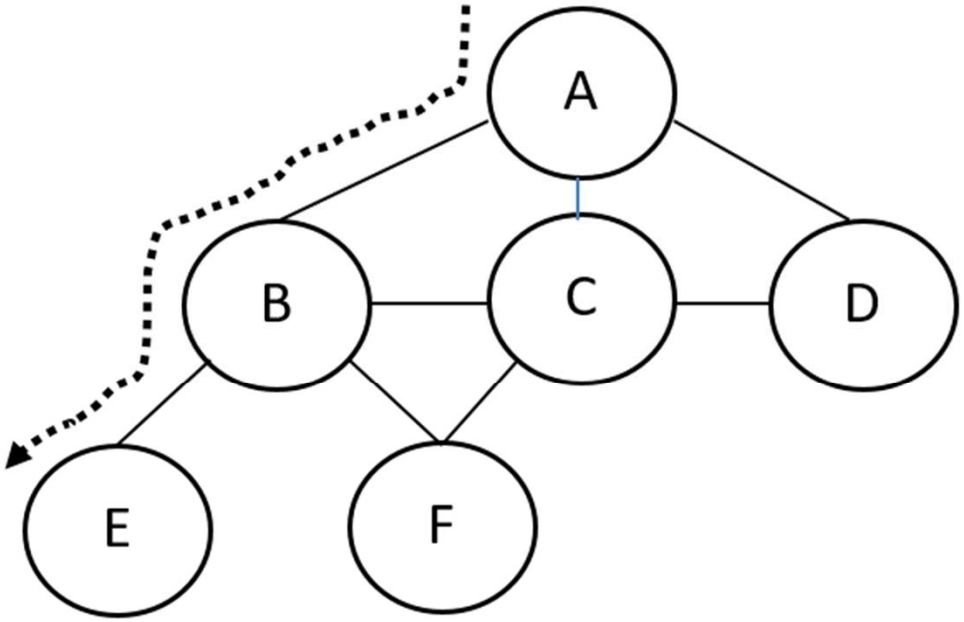
Recorridos en Grafos

- Su función es encontrar una trayectoria entre 2 nodos navegando por el grafo
- Utilizar un algoritmo que detecte ciclos para eliminarlas como posibles trayectorias.
- *Algoritmos básicos*
  - *En profundidad*
  - *En anchura*



Control de los nodos

- Controlar
  - *Nodos en espera de ser visitados*
  - *Nodos visitados*

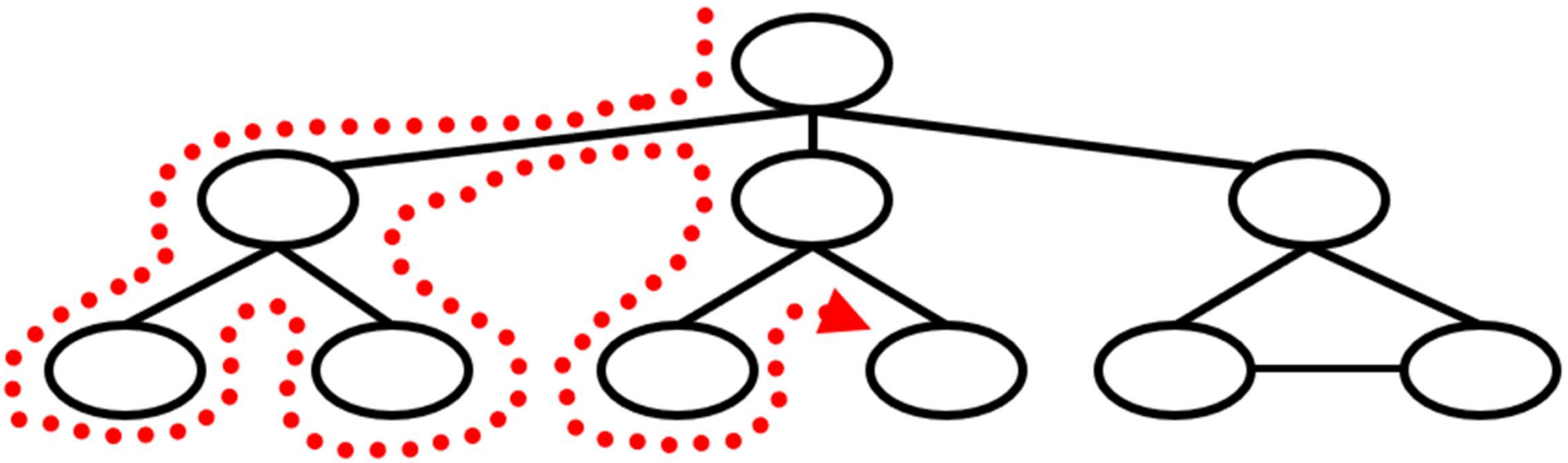


Nodos en espera: C,D, F

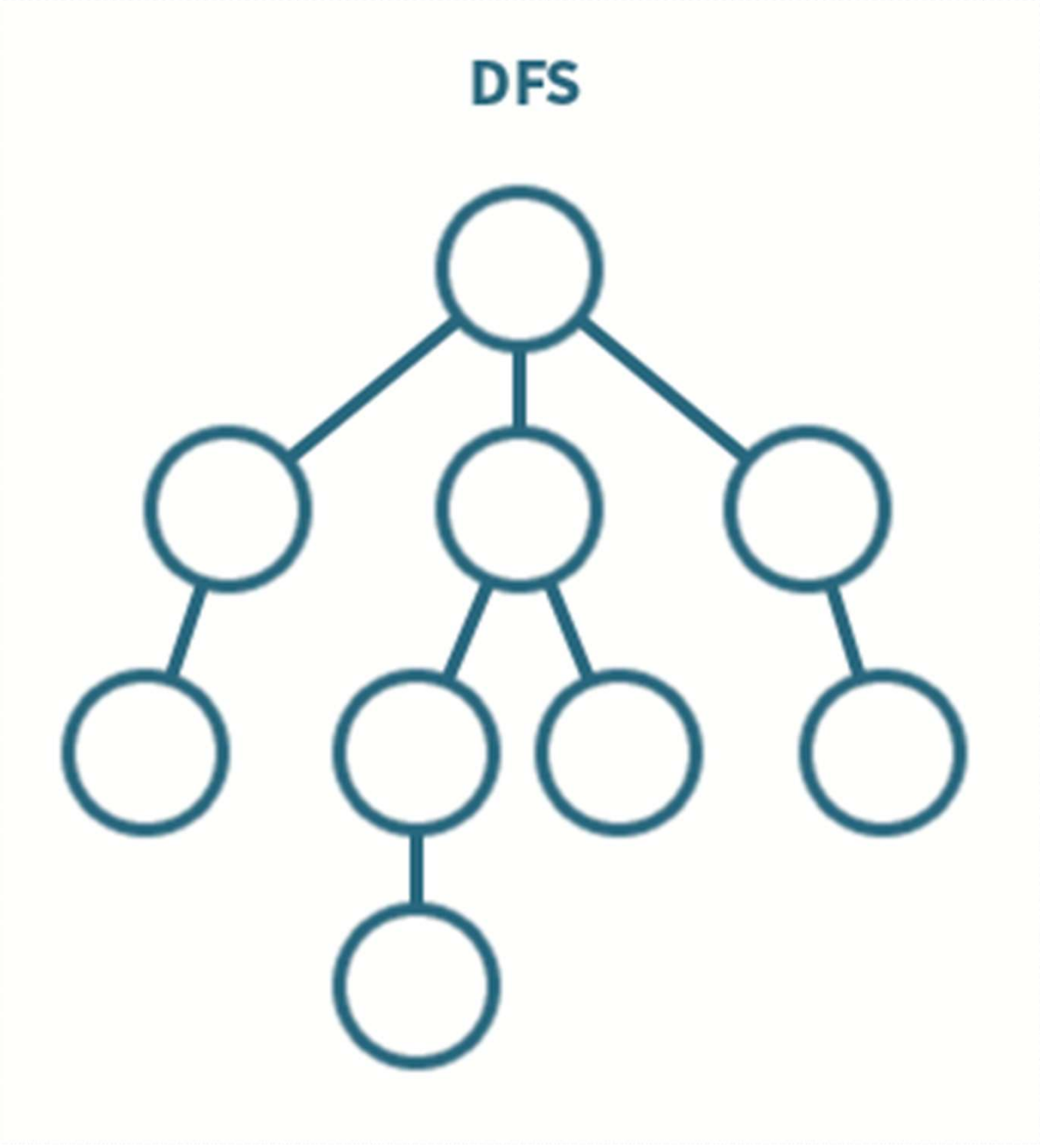
Nodos visitados: A-B-E

Nodos pendientes y nodos visitados durante el recorrido de un grafo.

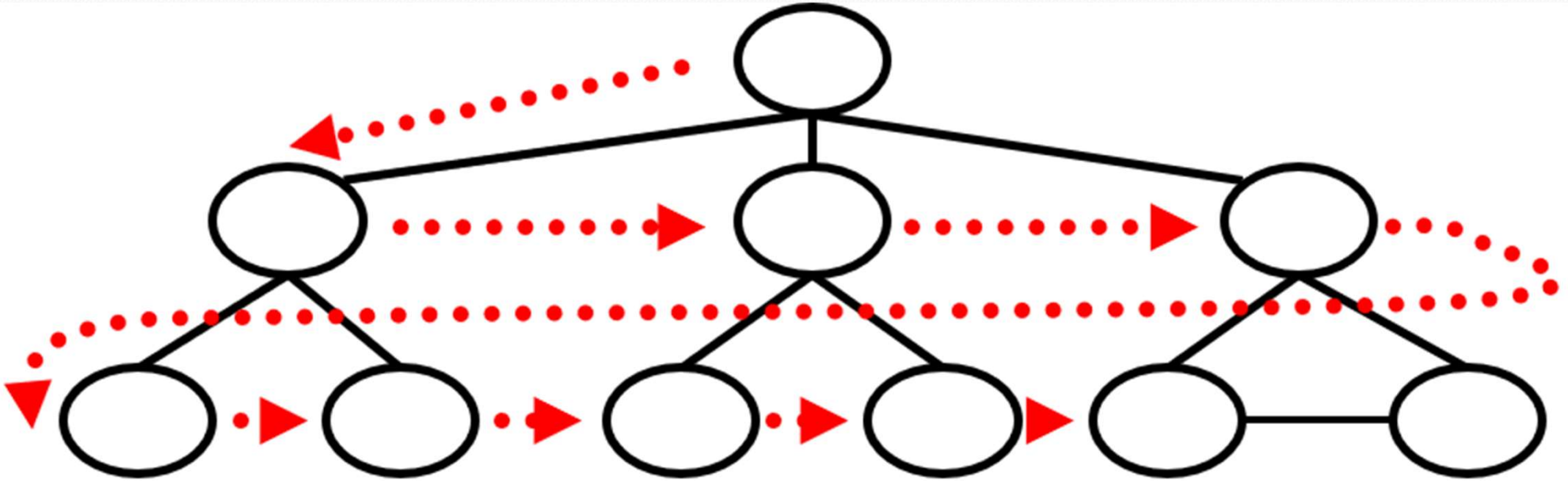
Recorrido en profundidad



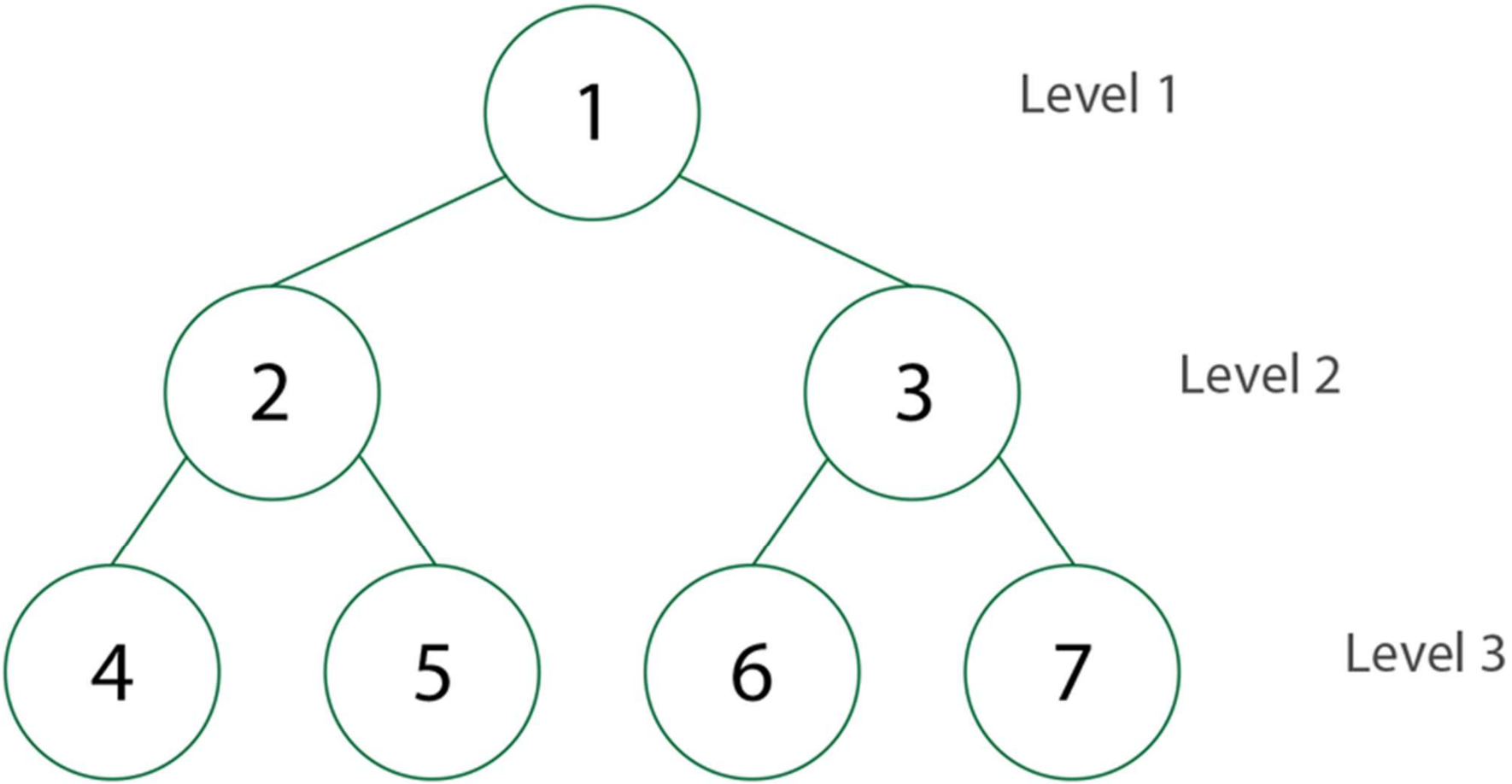
Recorrido en profundidad



Recorrido en anchura



Recorrido en anchura

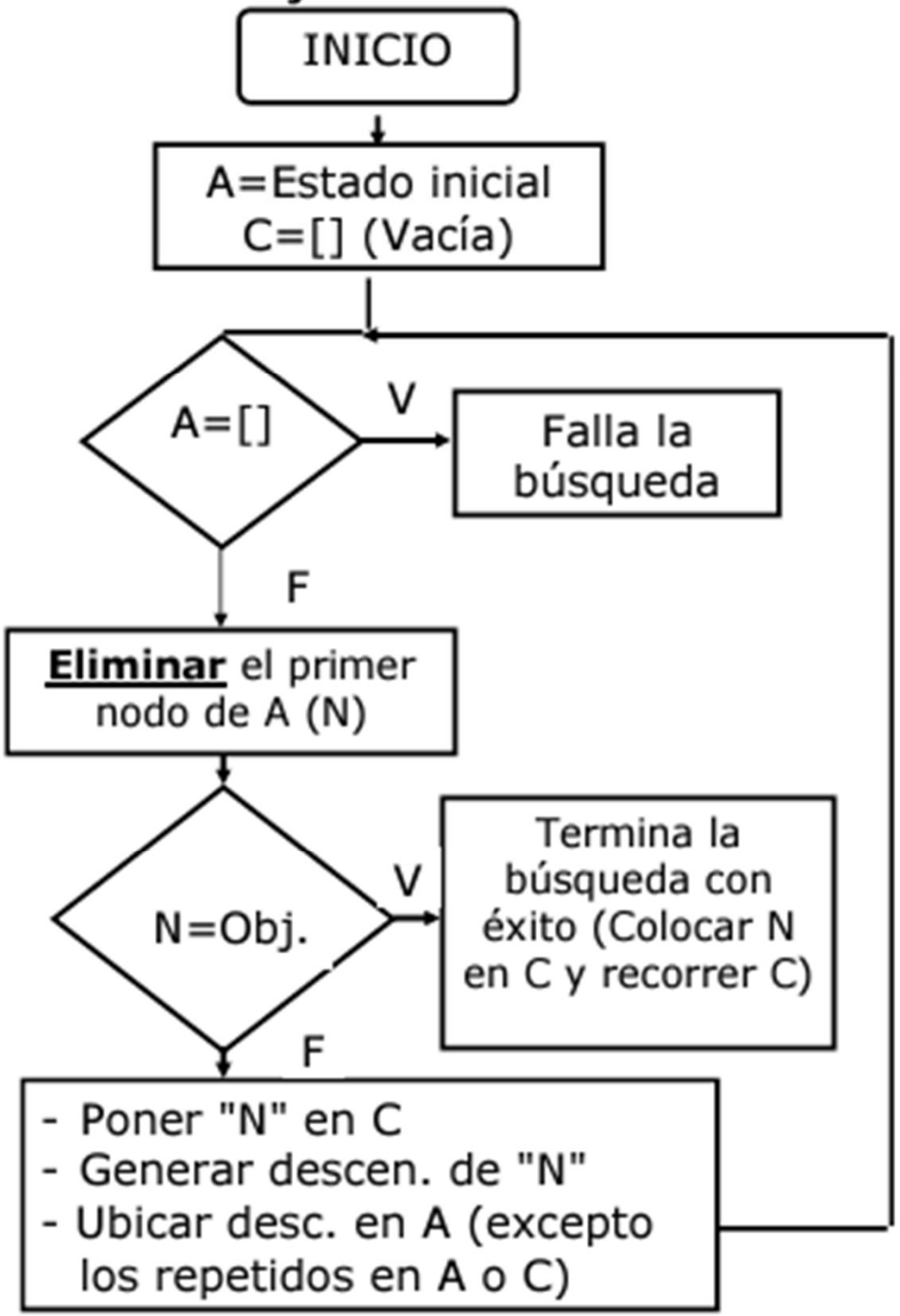


Requisitos para el recorrido

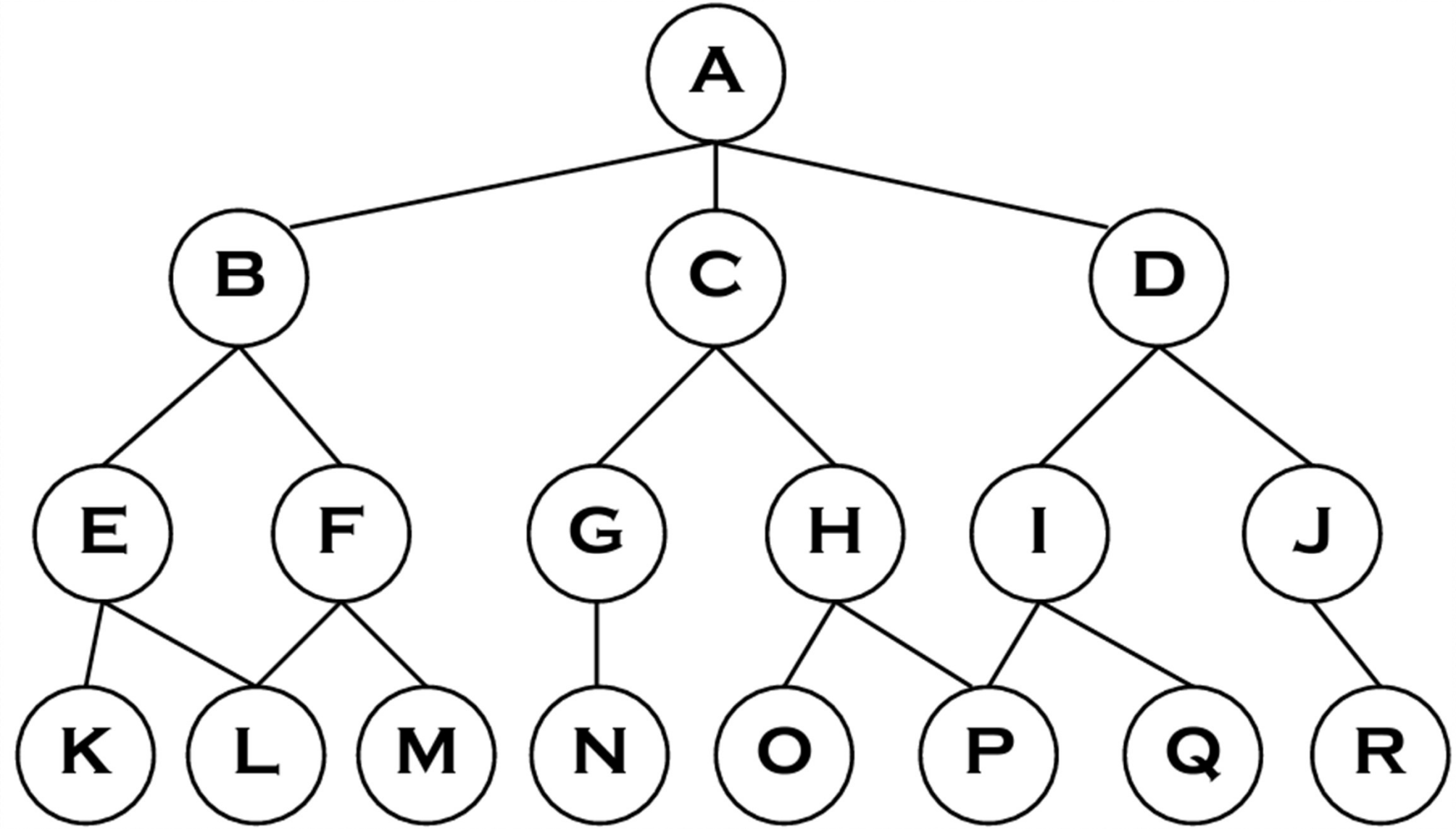
Estructura Abierta (A):	Almacena los nodos en espera de ser examinados
Lista Cerrada (C):	Almacena los nodos ya examinados

- El orden en el cual se remueven los nodos determina el orden del recorrido:
  - *Si A es una **pila** -> recorrido en profundidad*
  - *Si A es una **cola** -> recorrido en anchura*

Algoritmo de búsqueda en profundidad y anchura



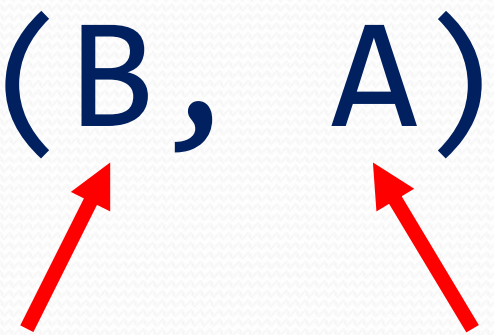
Ejemplo



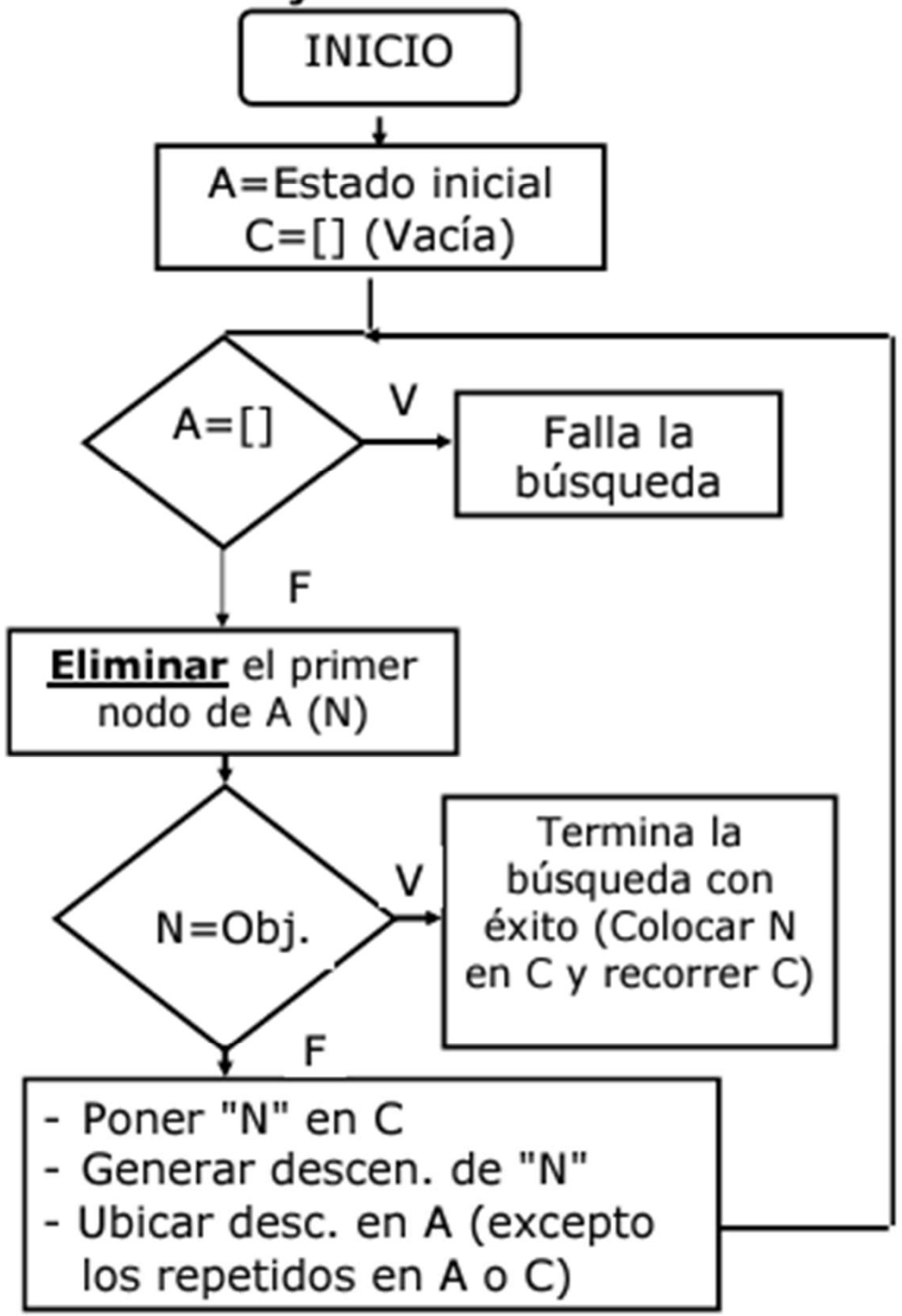
Formato de registro de los nodos visitados

- Sintaxis:
  - *(nodo visitado, nodo antecesor)*

- *Ejemplo:*

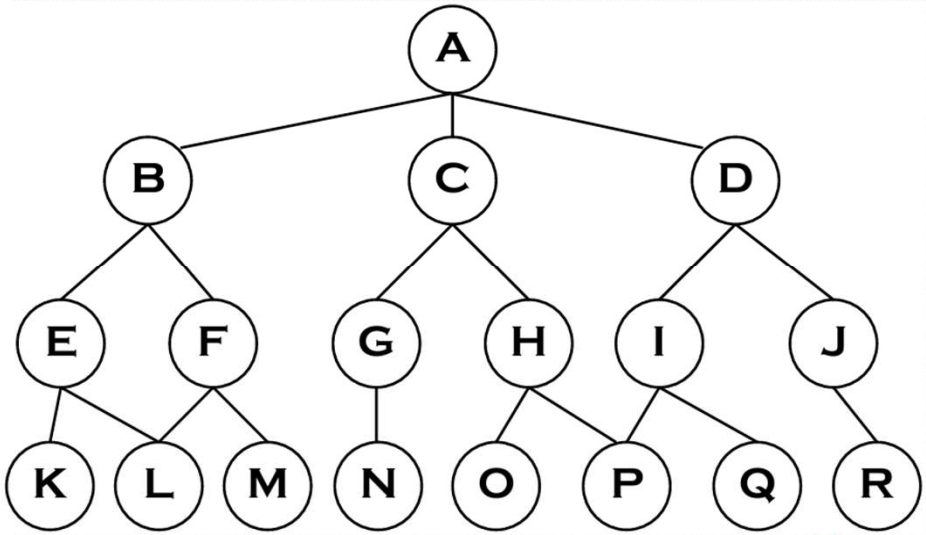


Algoritmo de búsqueda en profundidad y anchura



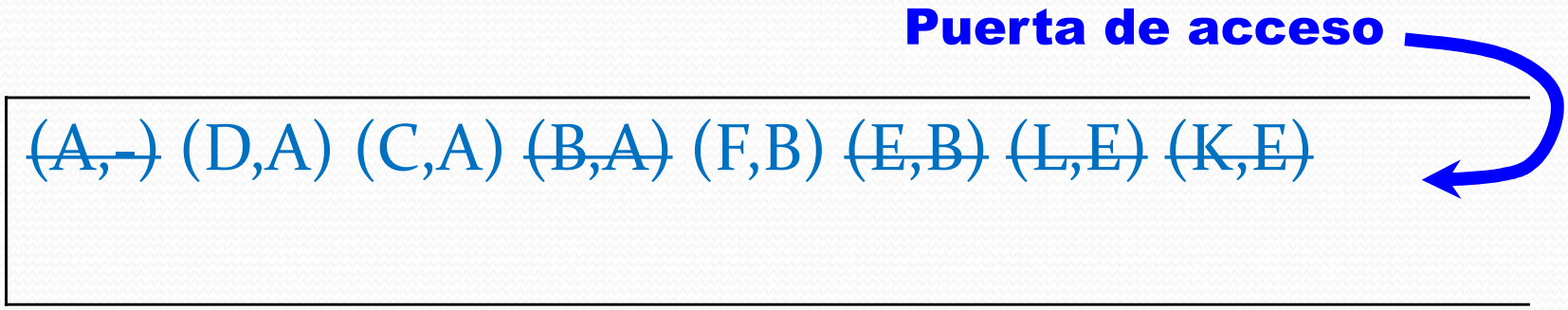
Ejercicio en Profundidad

Utilice este grafo para encontrar el camino del nodo A a L



Resultado de búsqueda del camino en profundidad del nodo A al nodo L

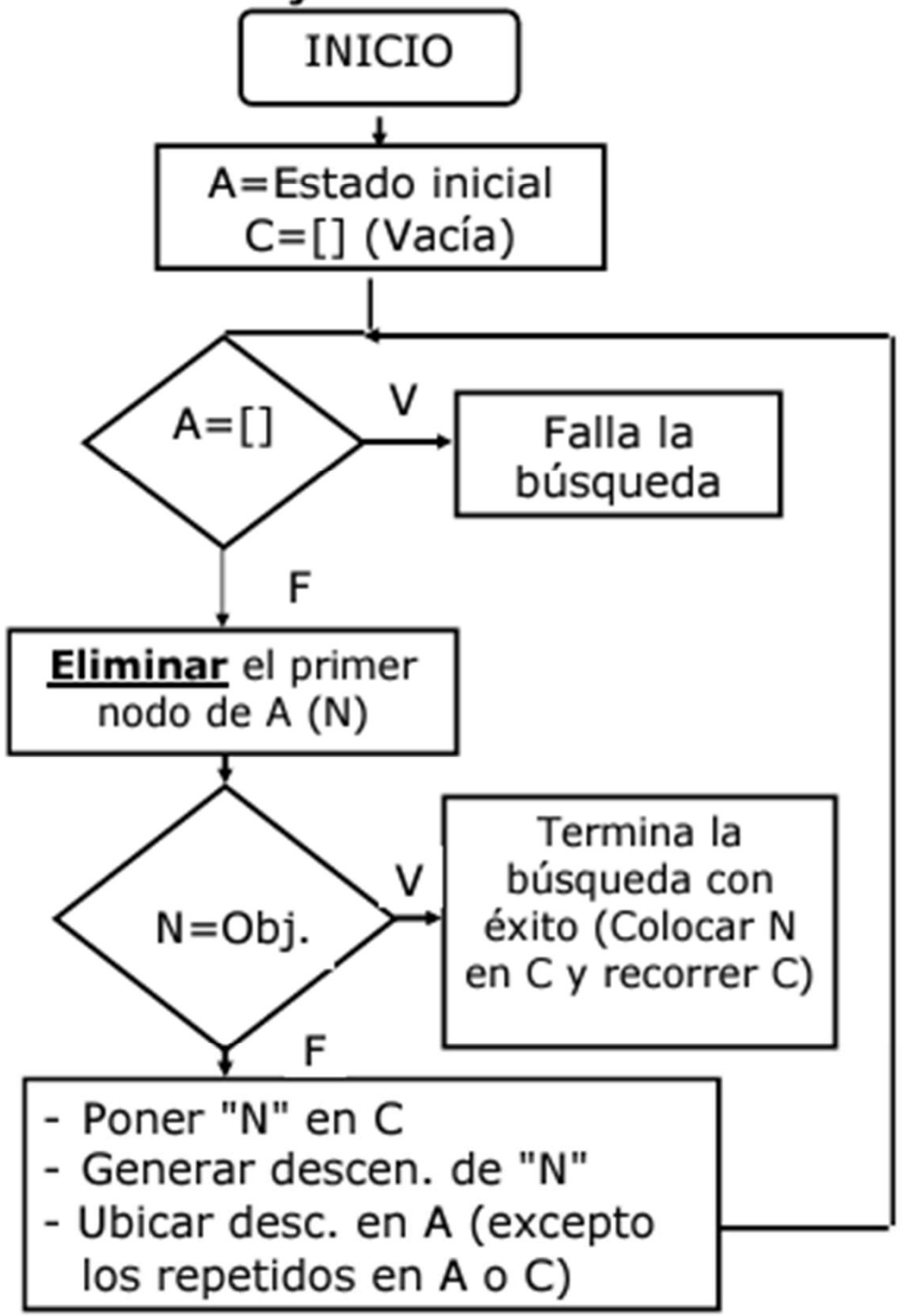
Pila A (espera)



Lista C (visitados)

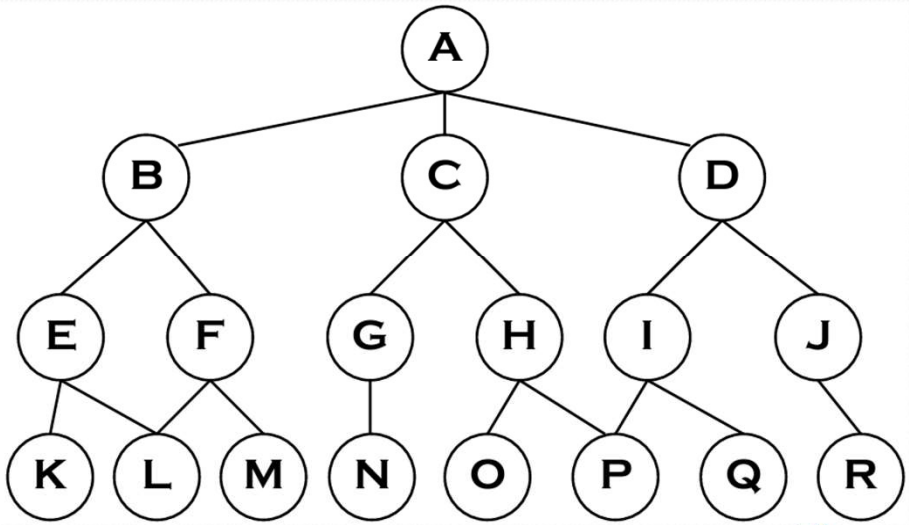
Ruta: A -> B -> E -> L

Algoritmo de búsqueda en profundidad y anchura

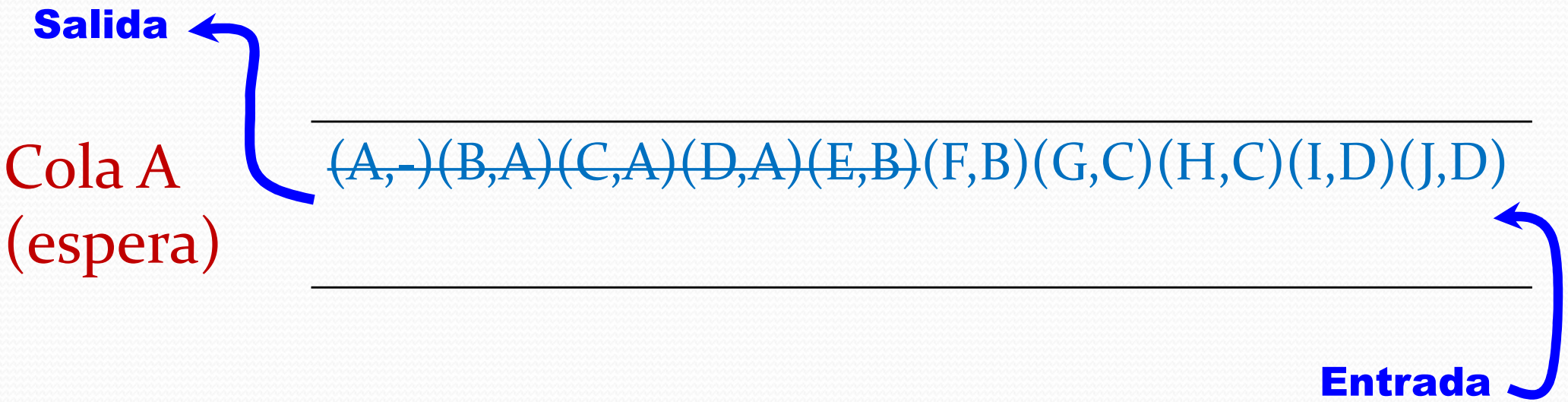


Ejercicio en Anchura

Utilice este grafo para encontrar el camino del nodo A a E

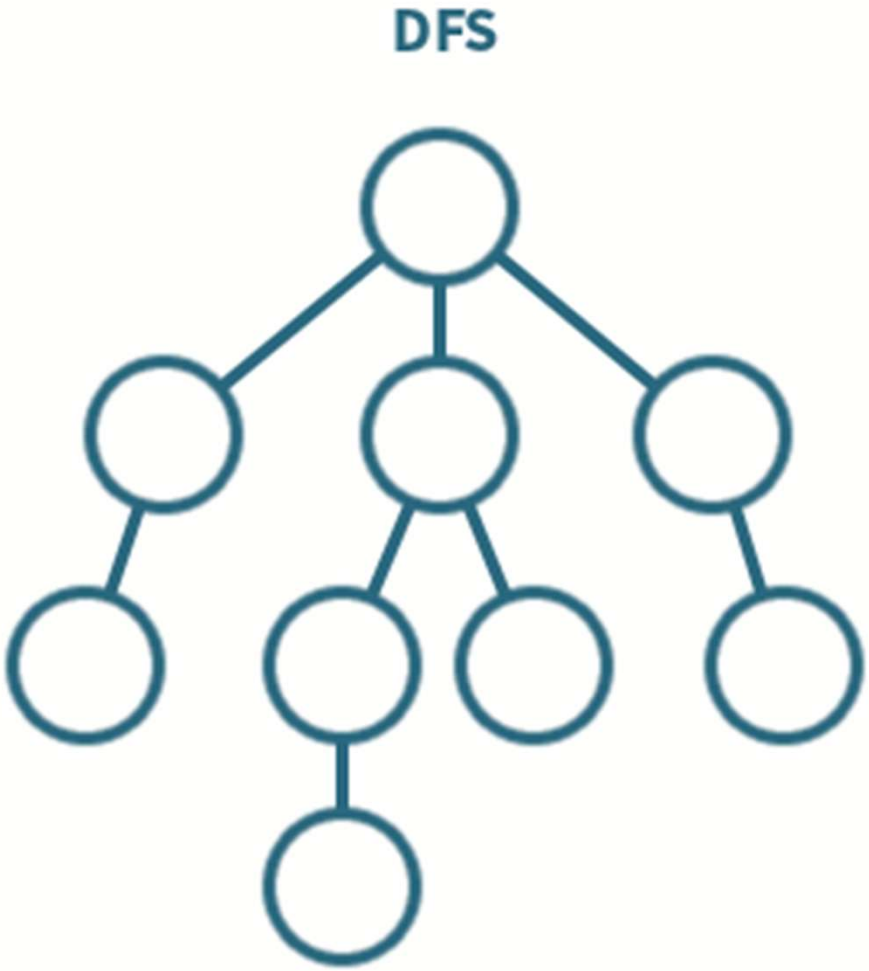


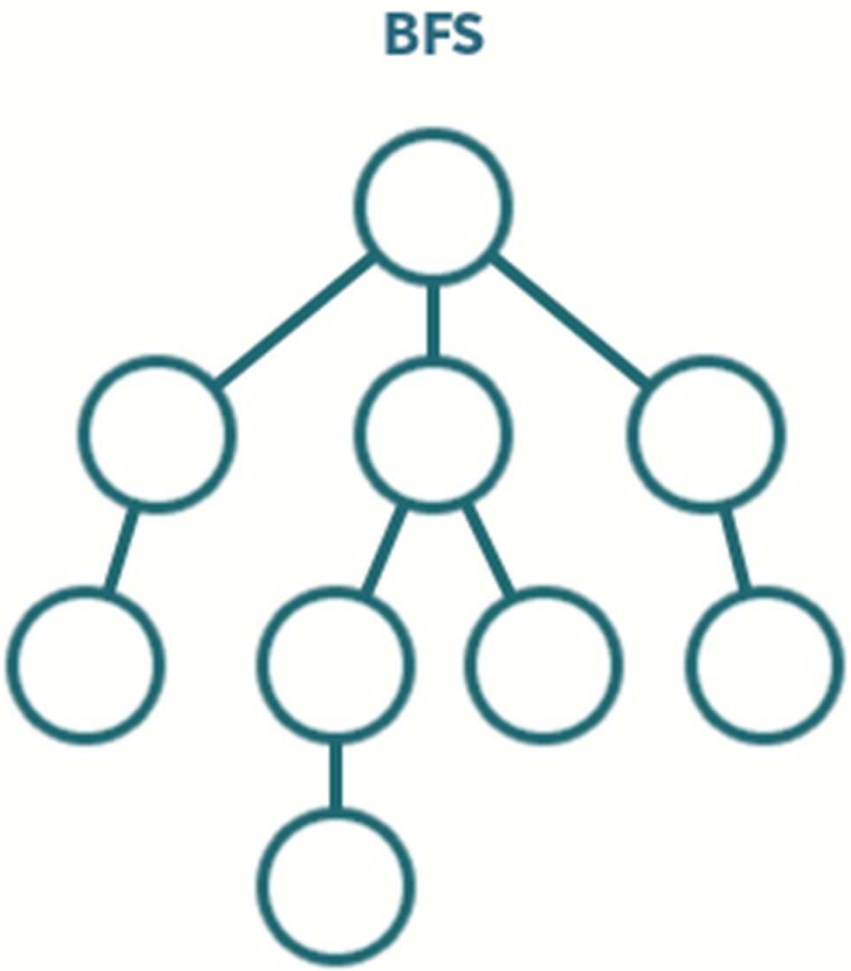
Resultado de búsqueda del camino en anchura del nodo A al nodo E



Ruta: A -> B -> E

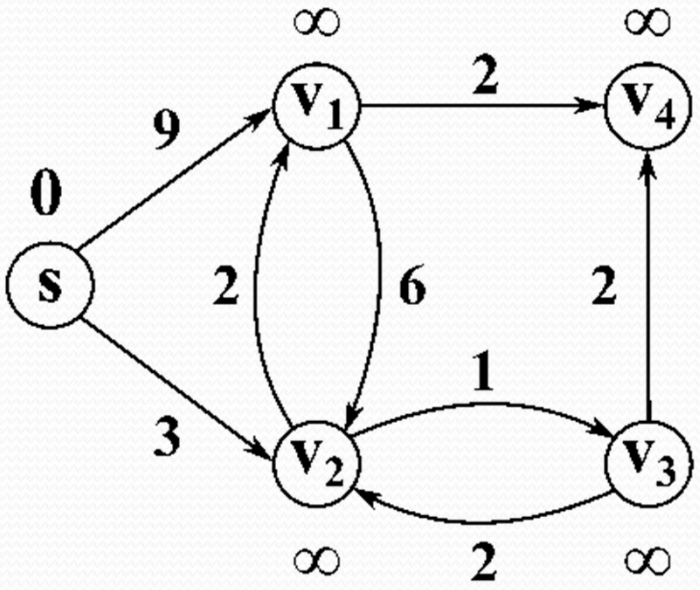
Profundidad vs. Anchura



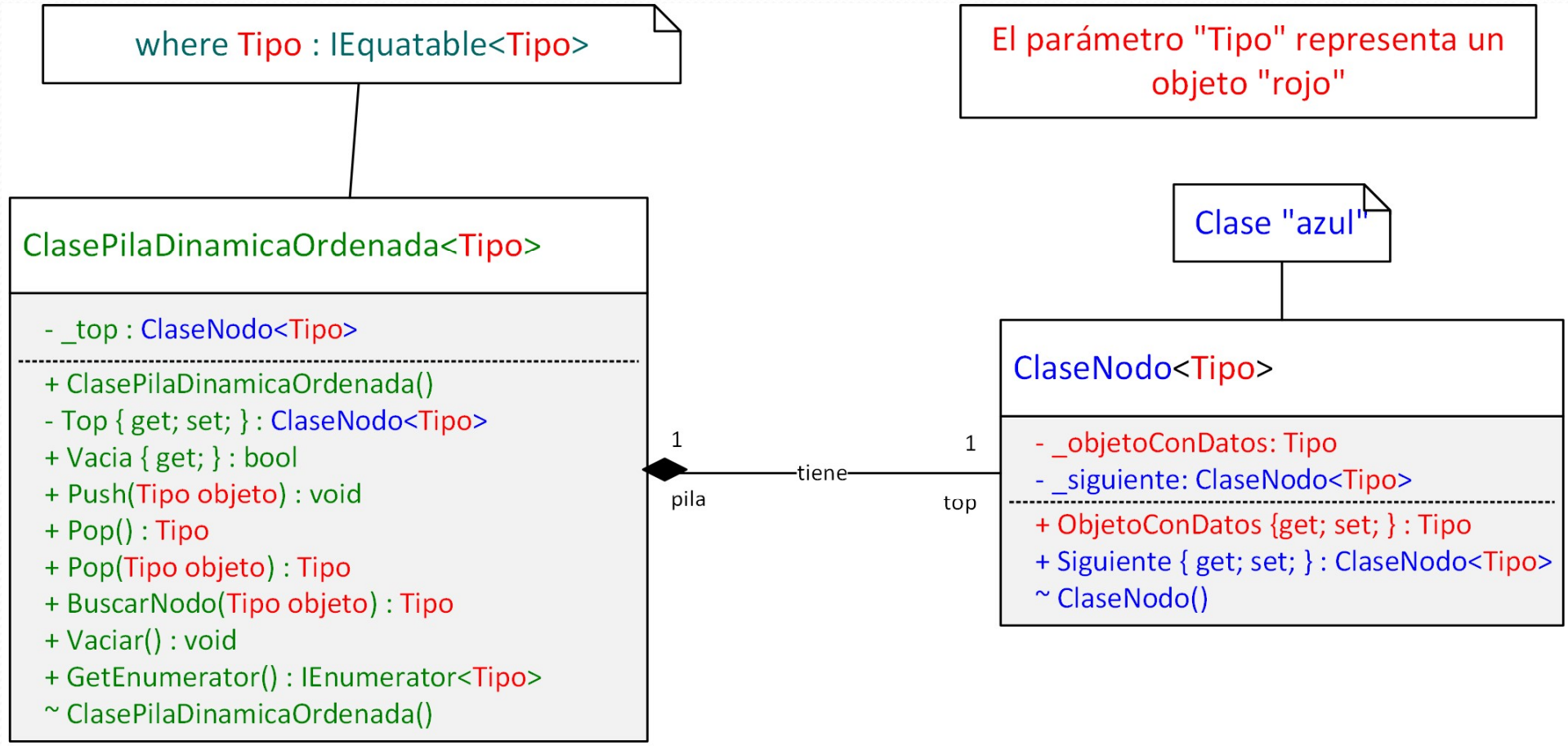


Camino más corto entre nodos

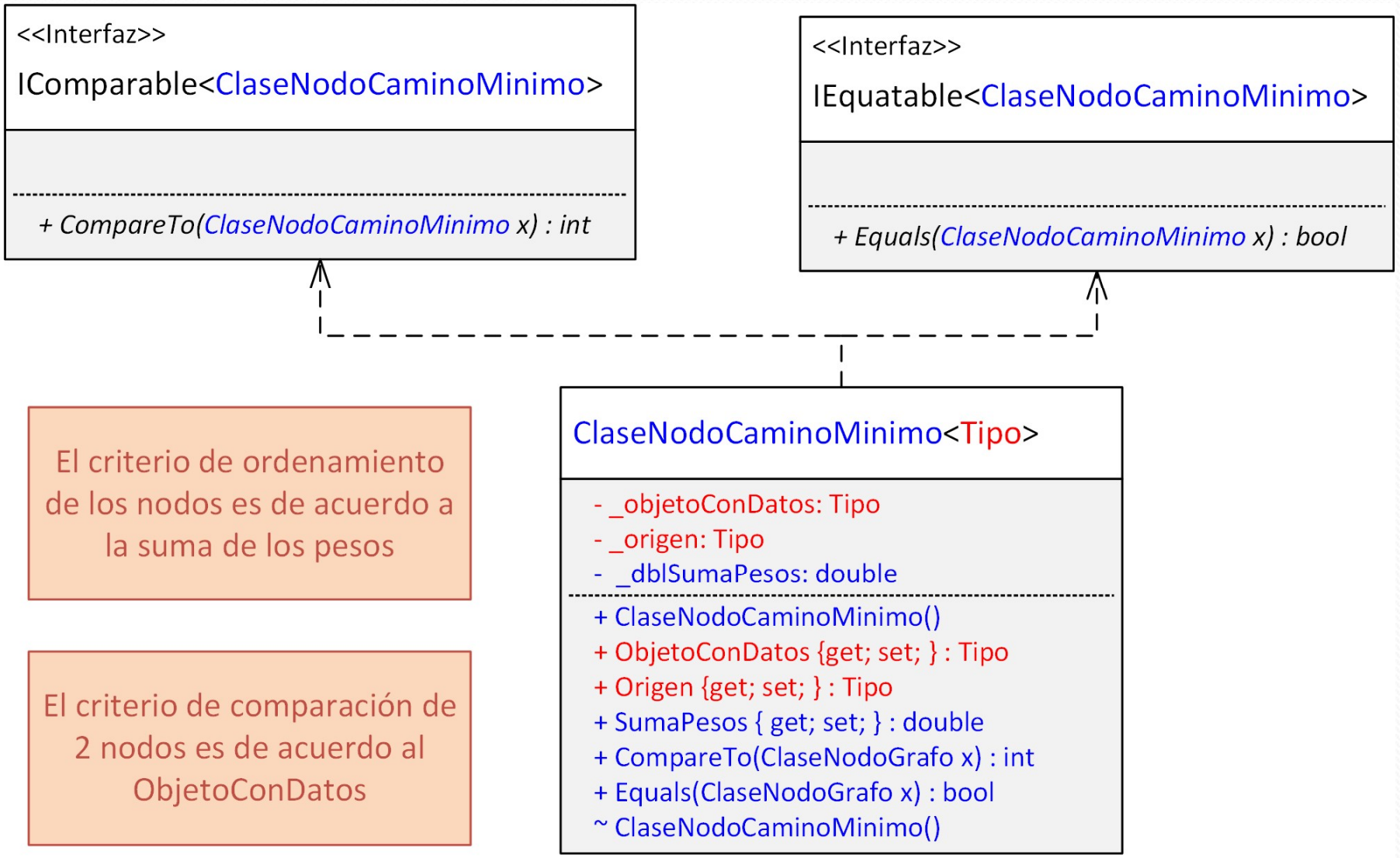
- Controlar nodos en espera
  - *Pila con nodos ordenados por la suma de los pesos*
- Controlar nodos visitados
  - *Nodos visitados*
  - *Nodos que forman la ruta*
  - *Lista doble desordenada*
- *Los arcos tienen peso*



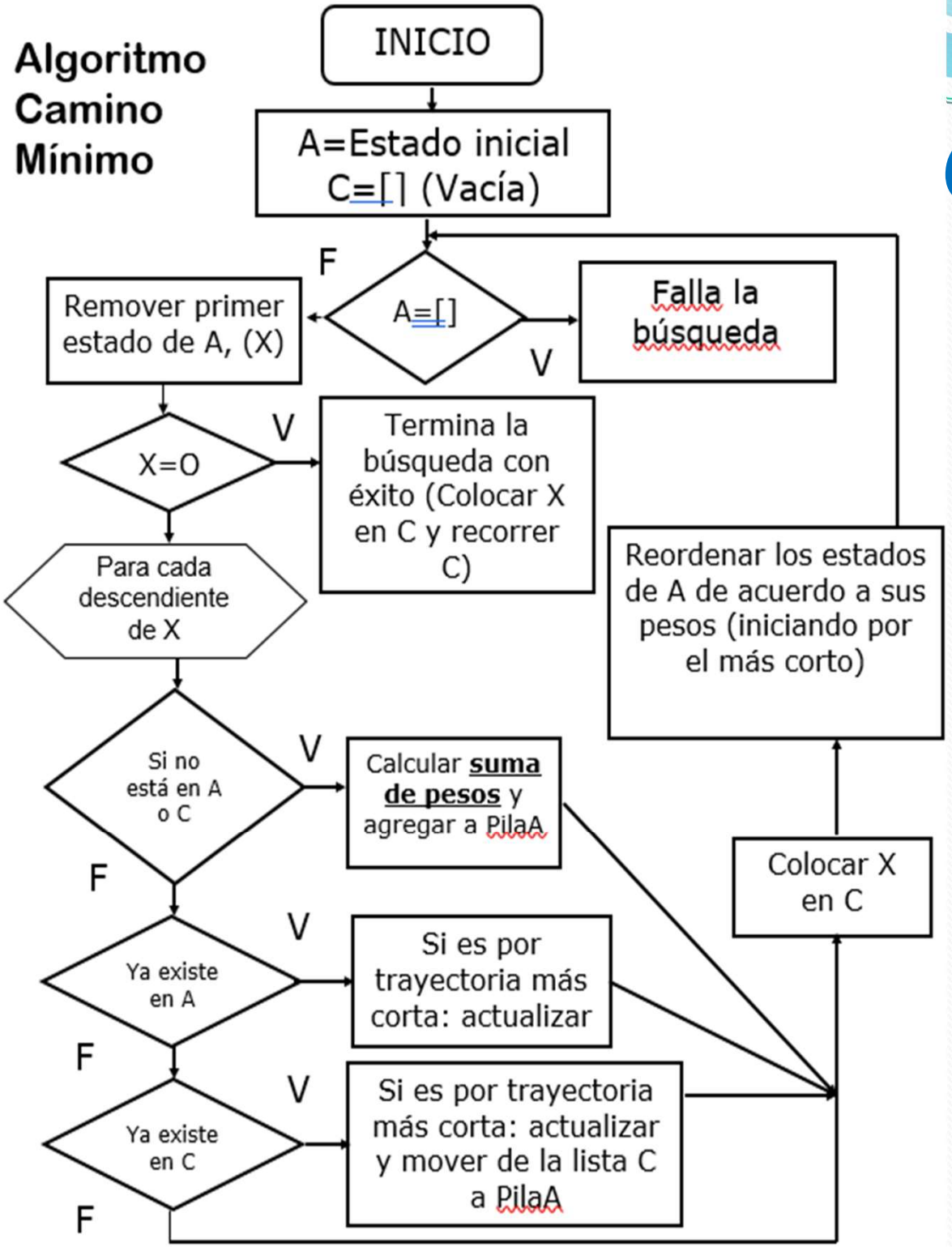
Pila ordenada para controlar nodos en espera de ser visitados



Clase para administrar los nodos visitados por el camino mínimo

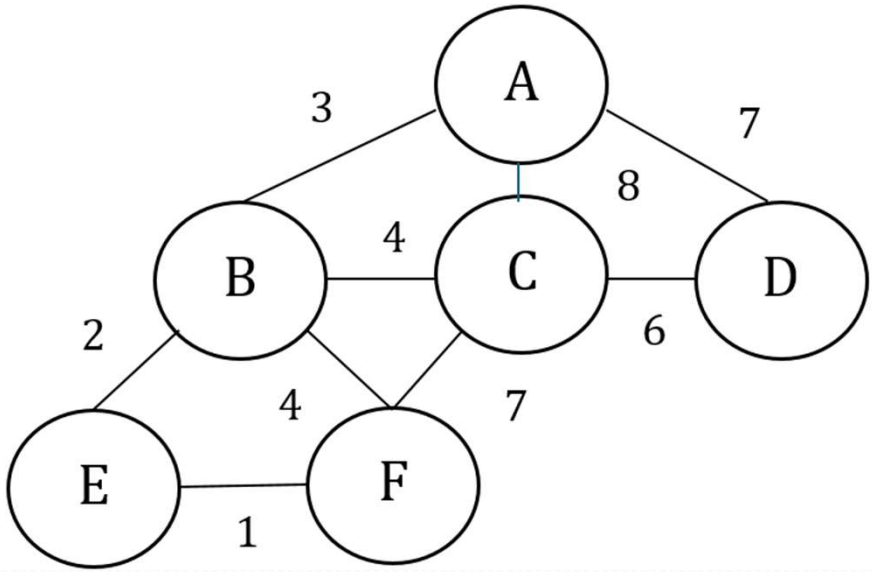


Algoritmo Camino Mínimo



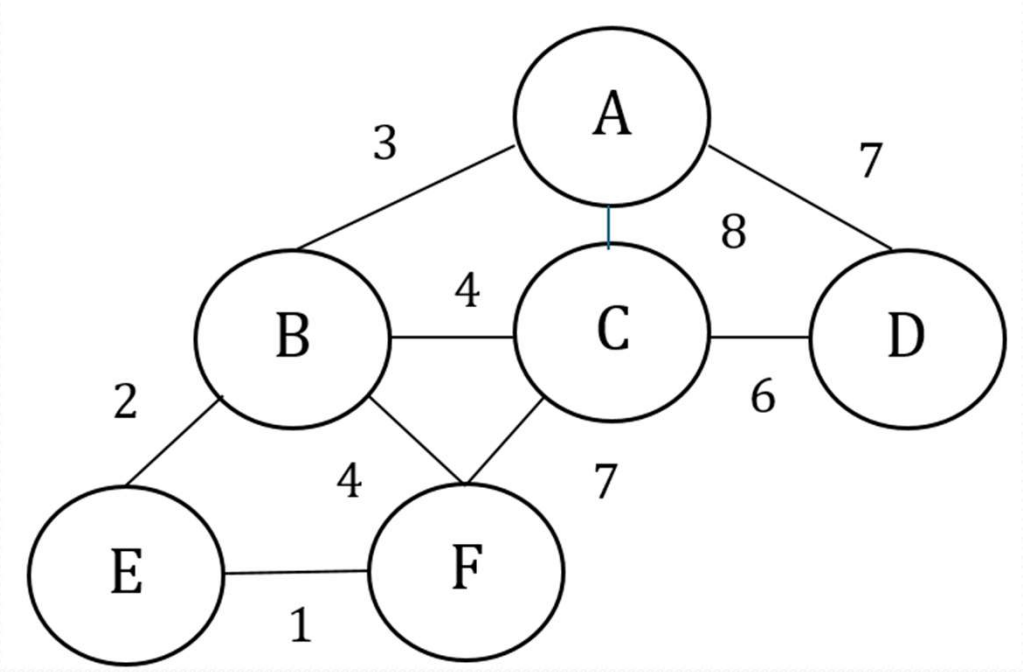
Ejercicio en Camino Mínimo

Utilice este grafo para encontrar el camino más corto del nodo A a F



Algoritmo del camino mínimo

Encontrar el camino mínimo de A a F

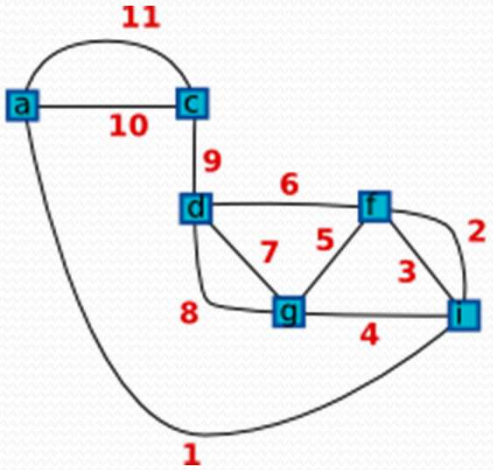


Nodos en espera: ~~(A,,0)~~,~~(C,A,8)~~,(C,B,7),(D,A,7),~~(B,A,3)~~,~~(F,B,7)~~,(F,E,6),~~(E,B,5)~~

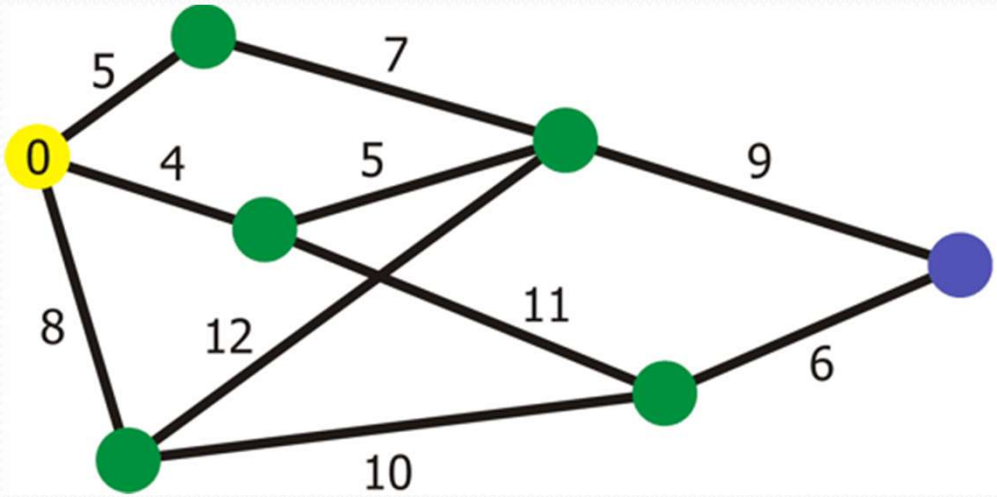
Nodos examinados: (A, -, 0), (B, A, 3), (E, B, 5), (F, E, 6)

Recorrido Camino Mínimo

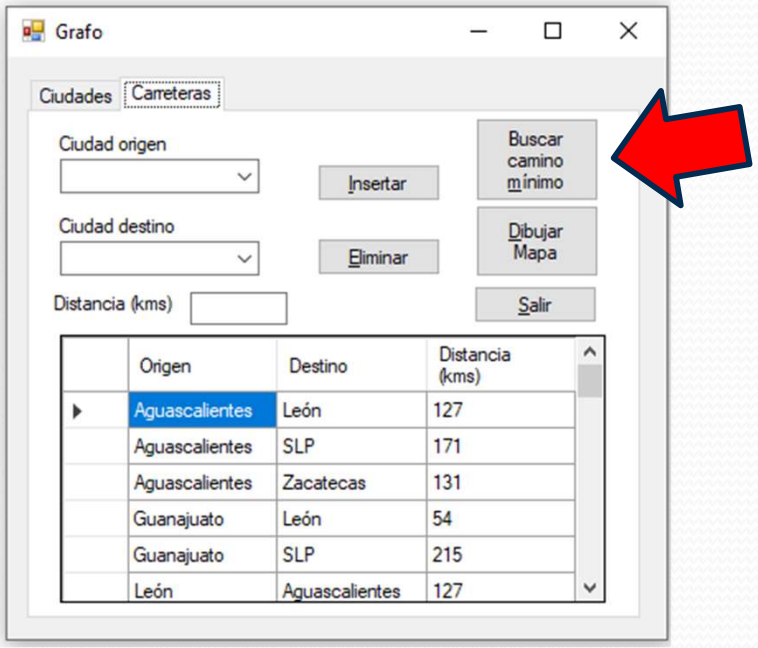
- Descargar:
  - *Pseudocódigo del algoritmo del camino mínimo*
  - *Pseudocódigo para descifrar la ruta al terminar el algoritmo del camino mínimo*
- https://nlaredo.tecnm.mx/takeyas/tallergrafos/



Demo



Grafo Ponderado y No Dirigido

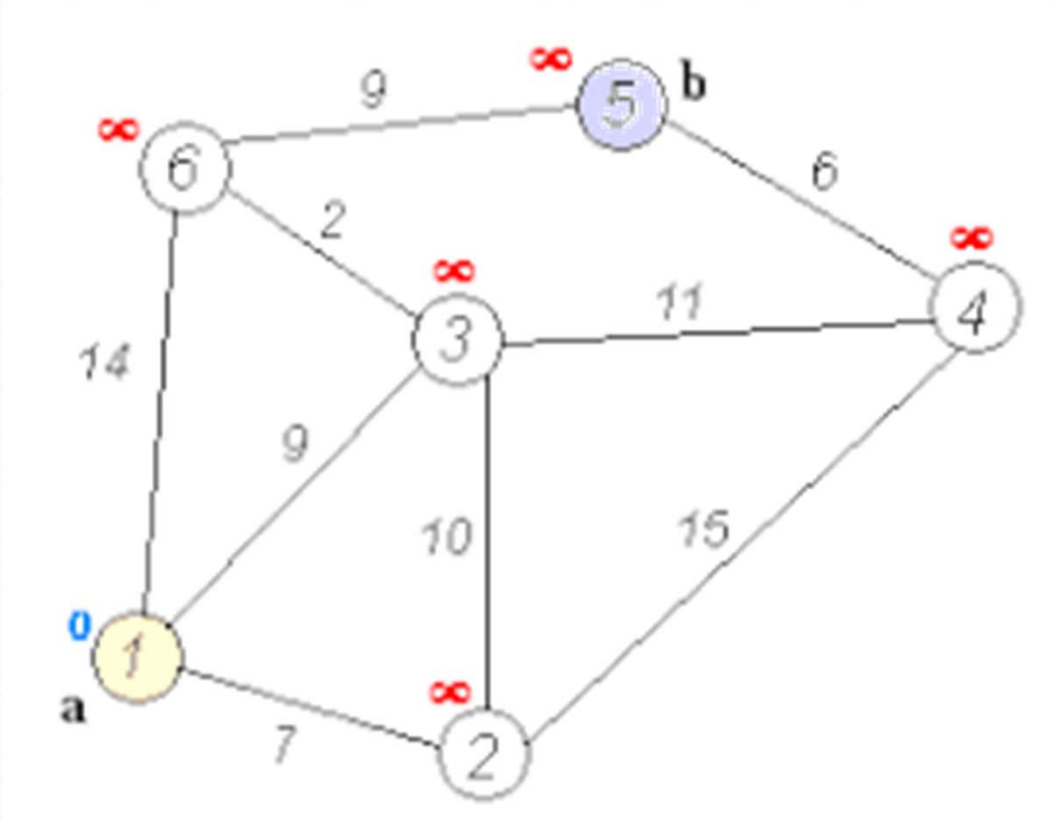


Prog.9.4.- Grafo Ponderado No Dirigido Formas (Mapa carretero)

Botón: Buscar camino mínimo

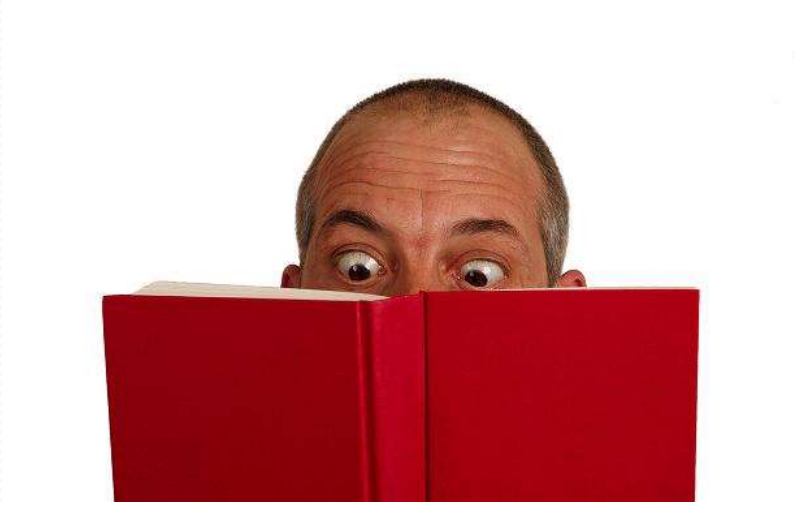
SESIÓN 4

Dibujar el grafo y Algoritmos inteligentes



Lectura

¿Cómo dibujar una estructura de datos utilizando Graphviz y su lenguaje dot?

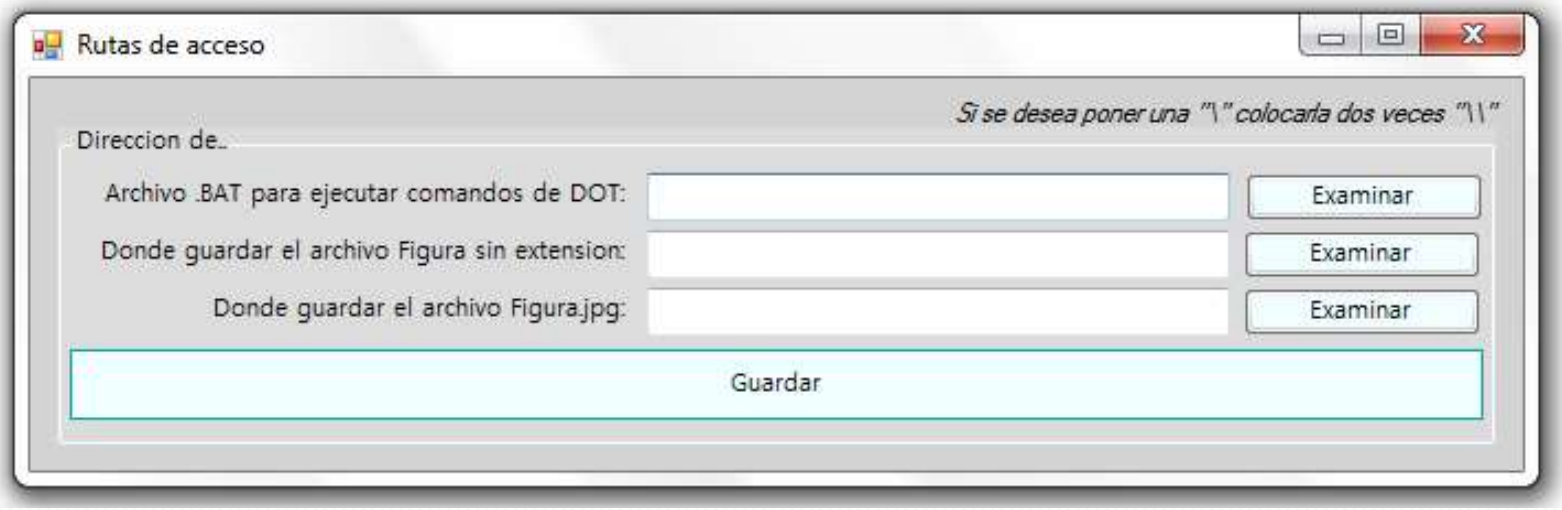


Graphviz y su lenguaje dot

https://graphviz.org/

Sugerencia

- *Sugerencia para seleccionar las carpetas de:*
  - *Archivo BAT*
  - *Archivo con el programa DOT*
  - *Archivo JPG con la figura del ABB*



Otra sugerencia

- *Diseñar un método que verifique la existencia de:*
  - *Carpeta de trabajo para dibujar el ABB*
    - *Crear la carpeta si no existe*
  - *Batch* `Dibujar.Bat`
    - *Crear el archivo* `Dibujar.Bat` *si no existe*
- *Invocar este método desde el método* `Form_Load()`

Código de Dibujar.bat

- *El batch* `Dibujar.bat` *debe ubicarse dentro de la carpeta donde se dibujará la figura*
  - *P. ejem. C:\Datos*

```
@echo off
c:
cd \Datos
del Figura.jpg
\"Program Files (x86)"\Graphviz2.31\bin\DOT -Tjpg -O Figura
```

Asegúrese escribir la ruta completa donde se ubica el programa DOT de GraphViz
(Puede variar dependiendo de la versión de GraphViz y de la carpeta de instalación)

Codificación del método

```
private void VerificarCarpeta()
{
   string strRuta = "\\22100123";

   // Si no existe la carpeta de la ruta
   if(!System.IO.Directory.Exists(strRuta))
   {
       // Crear la carpeta
       System.IO.Directory.CreateDirectory(strRuta);
       MessageBox.Show("Se ha creado la carpeta " + strRuta);
   }

   // Si no existe Dibujar.Bat
   if(!System.IO.File.Exists(strRuta+"\\Dibujar.bat"))
   {
      string strCodigoBat = "c:\ncd " + strRuta + "\ndel Figura.jpg\ndot -Tjpg -O Figura";

      // Se crea el archivo Dibujar.Bat
      System.IO.StreamWriter miArchivoBat = new System.IO.StreamWriter(strRuta + "\\Dibujar.Bat");
      miArchivoBat.Write(strCodigoBat);
      miArchivoBat.Close();
      MessageBox.Show("Se ha creado el archivo Dibujar.Bat\n\n"+strCodigoBat);
   }
}
```

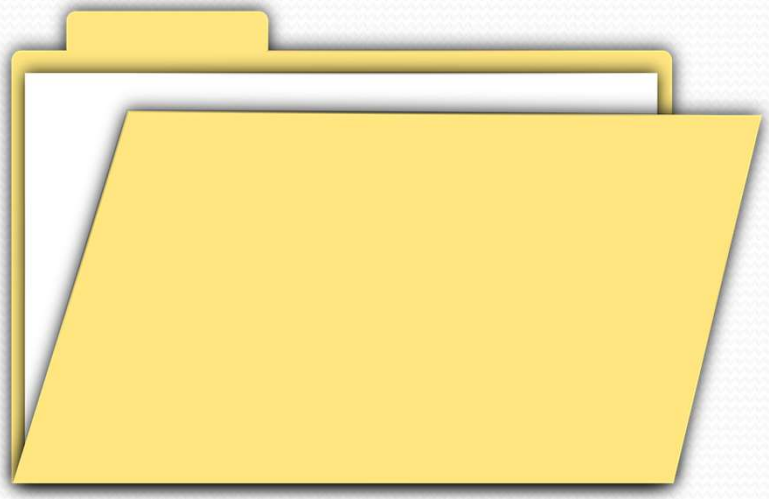
Se recomienda variable global

```
// Declaración de la variable global para la ruta
// de la carpeta de trabajo para dibujar el ABB
static string strRuta = "\\22100123";
private void Form1_Load(object sender, E
{
    VerificarCarpeta();
}

private void VerificarCarpeta()
{
    // Si no existe la carpeta de la ruta
    if (!System.IO.Directory.Exists(strRuta))
    {
        // Crear la carpeta
        System.IO.Directory.CreateDirectory(strRuta);
        MessageBox.Show("Se ha creado la carpeta " + strRuta);
    }

    // Si no existe Dibujar.Bat
    if (!System.IO.File.Exists(strRuta + "\\Dibujar.bat"))
    {
        string strCodigoBat = "c:\ncd " + strRuta + "\ndel Figura.jpg
          \ndot -Tjpg -O Figura";

        // Se crea el archivo Dibujar.Bat
        System.IO.StreamWriter miArchivoBat = new
          System.IO.StreamWriter(strRuta + "\\Dibujar.Bat");
        miArchivoBat.Write(strCodigoBat);
        miArchivoBat.Close();
        MessageBox.Show("Se ha creado el archivo Dibujar.Bat\n\n" +
          strCodigoBat);
    }
}
```

Variable global

Recomendación

- *Declarar una variable global de tipo string que almacene la ruta de la carpeta que guarde los archivos de la figura del ABB*



Código del botón para dibujar el ABB

```
private void btnDibujarABB_Click(object sender, EventArgs e)
{
   if (!ArbolBinarioBusqueda.EstaVacio())  // Si no está vacío ..
        DibujarFigura();  // Invoca el método para dibujar la figura
   else
   MessageBox.Show("Árbol Binario de Búsqueda vacío“);
}
```

Código del método DibujarFigura()

```
private void DibujarFigura()
{
// Se invoca el método para crear el archivo Figura que contiene el
programa en lenguaje DOT que crea el archivo Figura.jpg con la imagen del
ABB
    string strProgramaDot=CrearArchivoDot();

// Ejecución del proceso por lotes DIBUJAR.bat para crear la imagen
Figura.jpg
    InvocaDibujar();

    MessageBox.Show(strProgramaDot);
// Ejecución del método para crear una nueva forma y mostrar en ella la
Figura.jpg con la imagen del ABB
    MostrarImagenJPG();
}
```

Código del método independiente CrearArchivoDot()

```
private string CrearArchivoDot()
{
   // Recorrido del ABB para crear la línea con los comandos para el
archivo en lenguaje DOT
   string strProgramaDot = "";
   ArbolBinarioBusqueda.CrearArchivoDot(ref strProgramaDot);

  // Creación del archivo \Figura que contiene el programa en lenguaje
DOT)
  System.IO.StreamWriter miArchivoDot = new
System.IO.StreamWriter("c:\\Datos\\Figura");
  miArchivoDot.WriteLine(strProgramaDot);
  miArchivoDot.Close();

  return (strProgramaDot);
}
```

Código del método CrearArchivoDot() de la ClaseABB

```
class ClaseABB
{
  private void RecorrerNodos(ClaseNodoArbolBinarioBusqueda NodoActual, ref string strProgramaDot)  {
      if (NodoActual != null) {
          if (NodoActual.HijoIzq != null)
              strProgramaDot = strProgramaDot + "\n" + NodoActual.Dato.ToString() + "->" +
NodoActual.HijoIzq.Dato.ToString() + ";";

          if (NodoActual.HijoDer != null)
              strProgramaDot = strProgramaDot + "\n" + NodoActual.Dato.ToString() + "->" +
NodoActual.HijoDer.Dato.ToString() + ";";

          // Llamada recursiva para recorrer el subárbol izquierdo
          RecorrerNodos(NodoActual.HijoIzq, ref strProgramaDot);

          // Llamada recursiva para recorrer el subárbol derecho
          RecorrerNodos(NodoActual.HijoDer, ref strProgramaDot);
      }
}

public void CrearArchivoDot(ref string strProgramaDot) {
   if (!EstaVacio())  // Si no está vacío ...
   {
     strProgramaDot = strProgramaDot + "digraph Figura {";
     strProgramaDot = strProgramaDot + "\nRaíz->" + Raiz.Dato.ToString() + ";";
     RecorrerNodos(Raiz, ref strProgramaDot);
     strProgramaDot = strProgramaDot + "\n}";
   }
 }
}
```

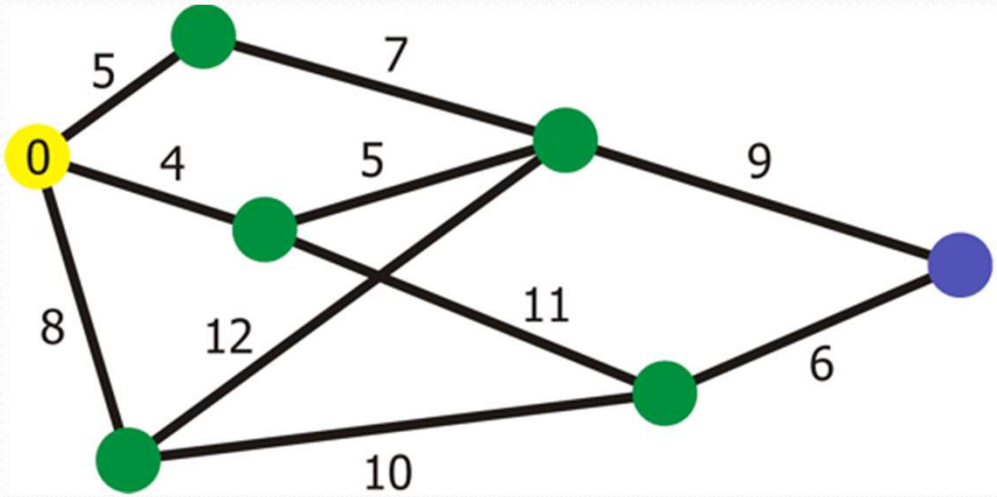
Código del método InvocaDibujar()

```
private void InvocaDibujar()
{
   // El archivo por lotes DIBUJAR.bat contiene el siguiente código:
   //    @echo off
   //    del \Figura.jpg
   //    \“Program Files (x86)"\GraphViz2.31\bin\DOT -Tjpg -O \Figura

   // Ejecuta DIBUJAR.bat ubicado en la carpeta c:\Datos
   System.Diagnostics.Process.Start("c:\\Datos\\DIBUJAR.bat");
}
```

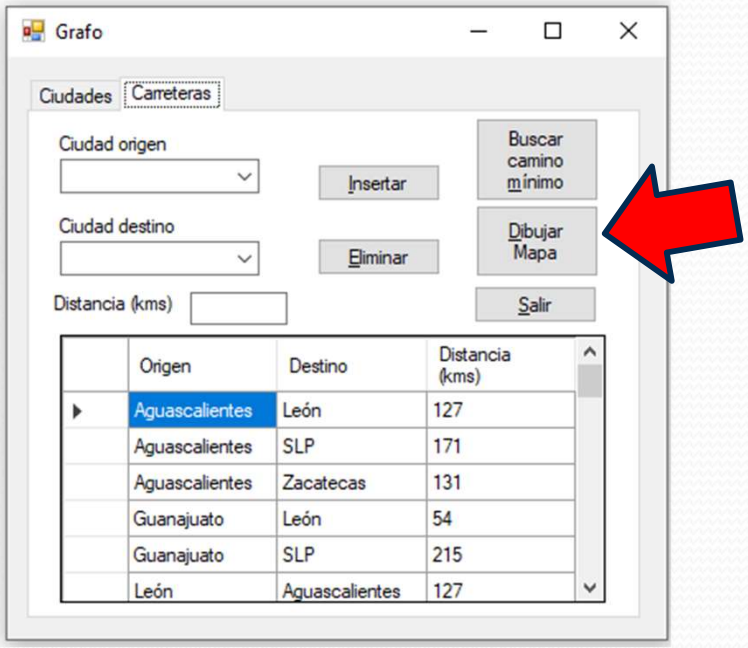
Código del método MostrarImagenJPG()

```
private void MostrarImagenJPG()    {

  picABB.Width = 990;  // Define el ancho del pictureBox
  picABB.Height = 545;  // Define la altura del pictureBox
  picABB.SizeMode = PictureBoxSizeMode.Zoom;

  // Abre el archivo Figura.jpg en modo de sólo lectura
  System.IO.FileStream miArchivoJPG; // Declaración del archivo Figura.jpg

  try { // Intenta abrir el archivo
    miArchivoJPG = new System.IO.FileStream("c:\\Datos\\Figura.jpg", System.IO.FileMode.Open,
System.IO.FileAccess.Read);
  }
  catch (Exception x) { // En caso de error ...
     MessageBox.Show("No se pudo abrir el archivo FIGURA.jpg");
     return;
  }

  // Carga la imagen del ArbolBinarioBusqueda.jpg en el pictureBox

  try {  // Intenta cargar la imagen en el pictureBox
     picABB.Image = System.Drawing.Bitmap.FromStream(miArchivoJPG);
  }
  catch (Exception x) { // En caso de error ...
     MessageBox.Show(“Error al cargar la imagen del archivo FIGURA.jpg");
     miArchivoJPG.Close();  // Cierra el archivo
     return;
  }

  miArchivoJPG.Close();
  picABB.Refresh(); // Refresca la imagen
}
```

Demo



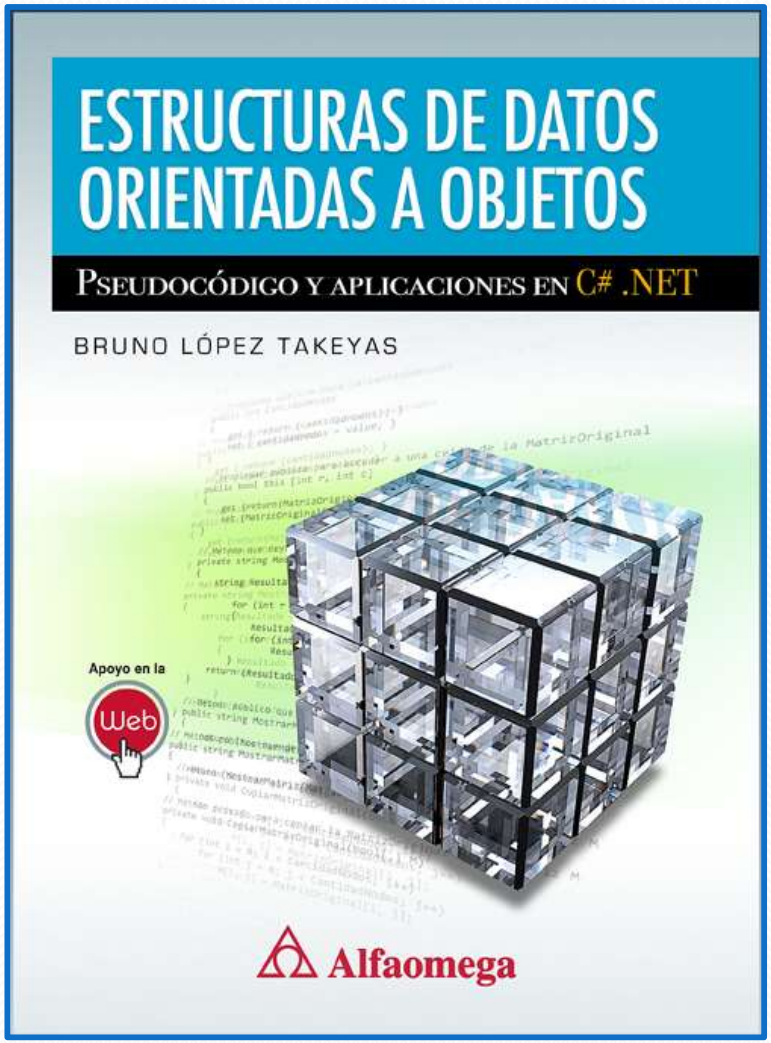
Grafo Ponderado y No Dirigido



Prog.9.4.- Grafo Ponderado No Dirigido Formas (Mapa carretero)

Botón: Dibujar Mapa

Lectura

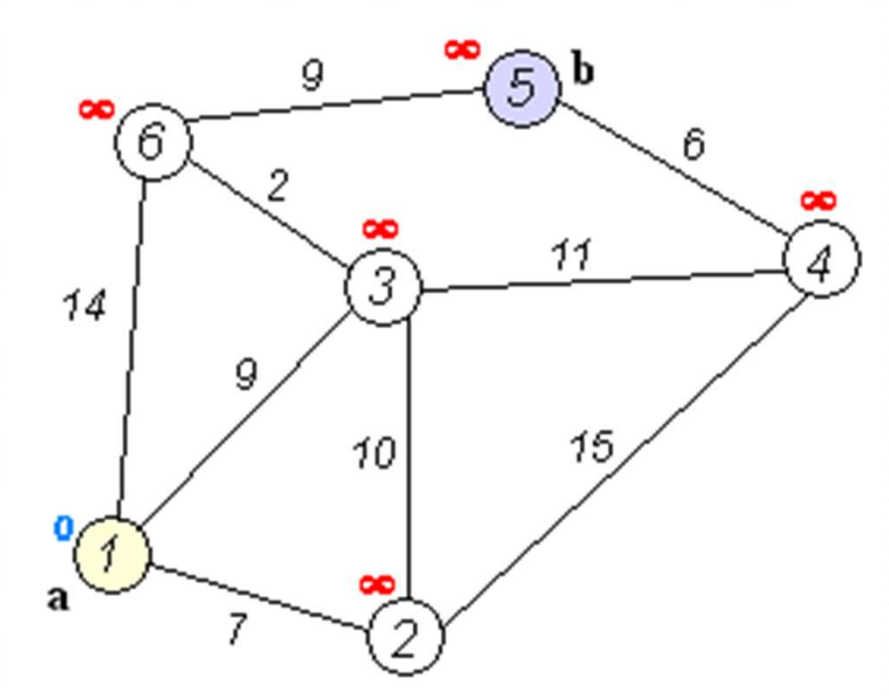


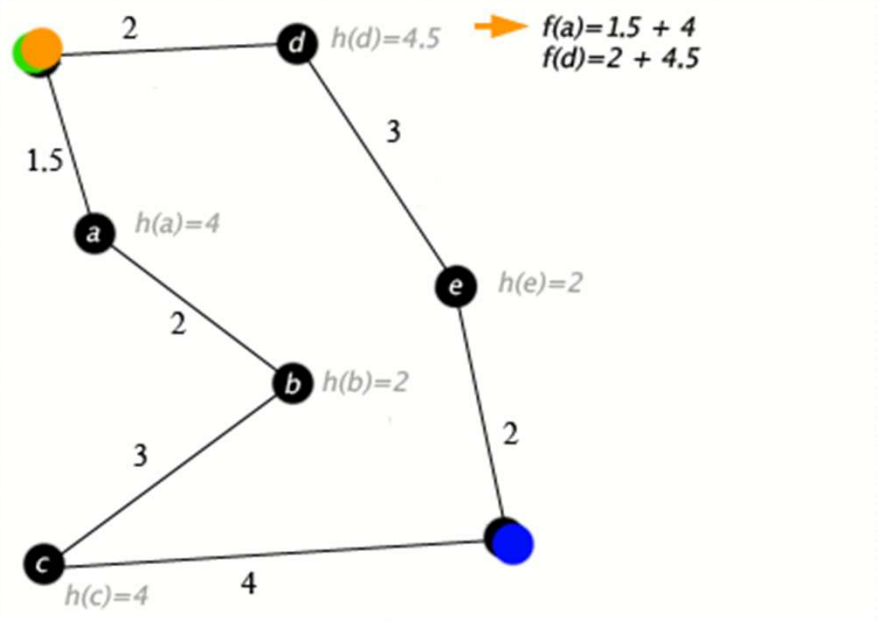
Para reforzar el tema de ***Grafos*** *se recomienda la lectura de:*

Capítulo 9.- Grafos

Algoritmos Inteligentes con Grafos

- *Algoritmo Best-First*
- *Algoritmo A**





Algoritmo Best-First

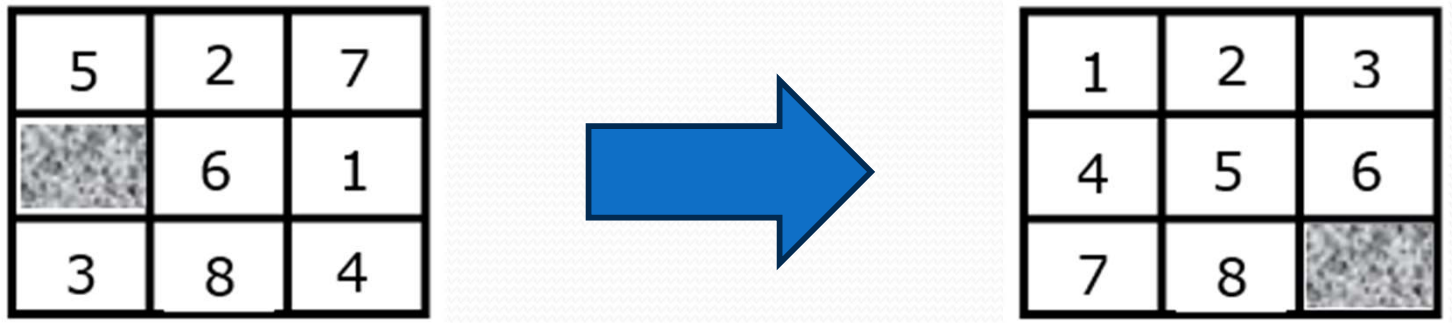
- *Guiado por valor heurístico colocado en el nodo (peso en el nodo – no en el arco)*
- *Combina ventajas de los recorridos en profundidad y en anchura*
- *No es determinista como profundidad y anchura (no sigue el mismo patrón de recorrido)*
- *Puede cambiar de ruta por una que parezca más prometedora*

¿Qué significa el valor heurístico?

- *Cada nodo tiene un peso (heurística)*
- *La heurística depende del problema analizado*
- *La heurística representa un criterio, regla o método que permite evaluar un nodo y determinar cuál es la mejor opción durante un recorrido*
- *Se requiere disponer de información o conocimiento del problema para determinar la heurística*

Ejemplo del uso de heurística

- *Juego de rompecabezas de números*
- *Cantidad de celdas fuera de su posición*
- *El mejor nodo tiene la heurística menor*

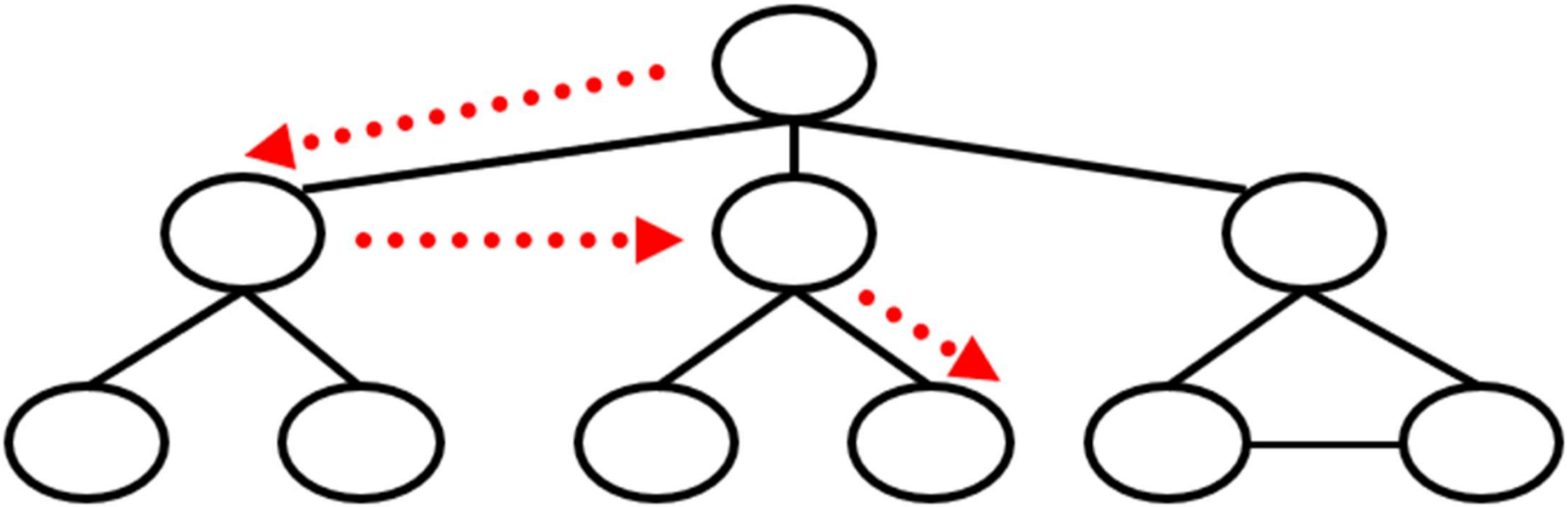


Nodo inicial
Heurística = 7

Nodo final
Heurística = 0

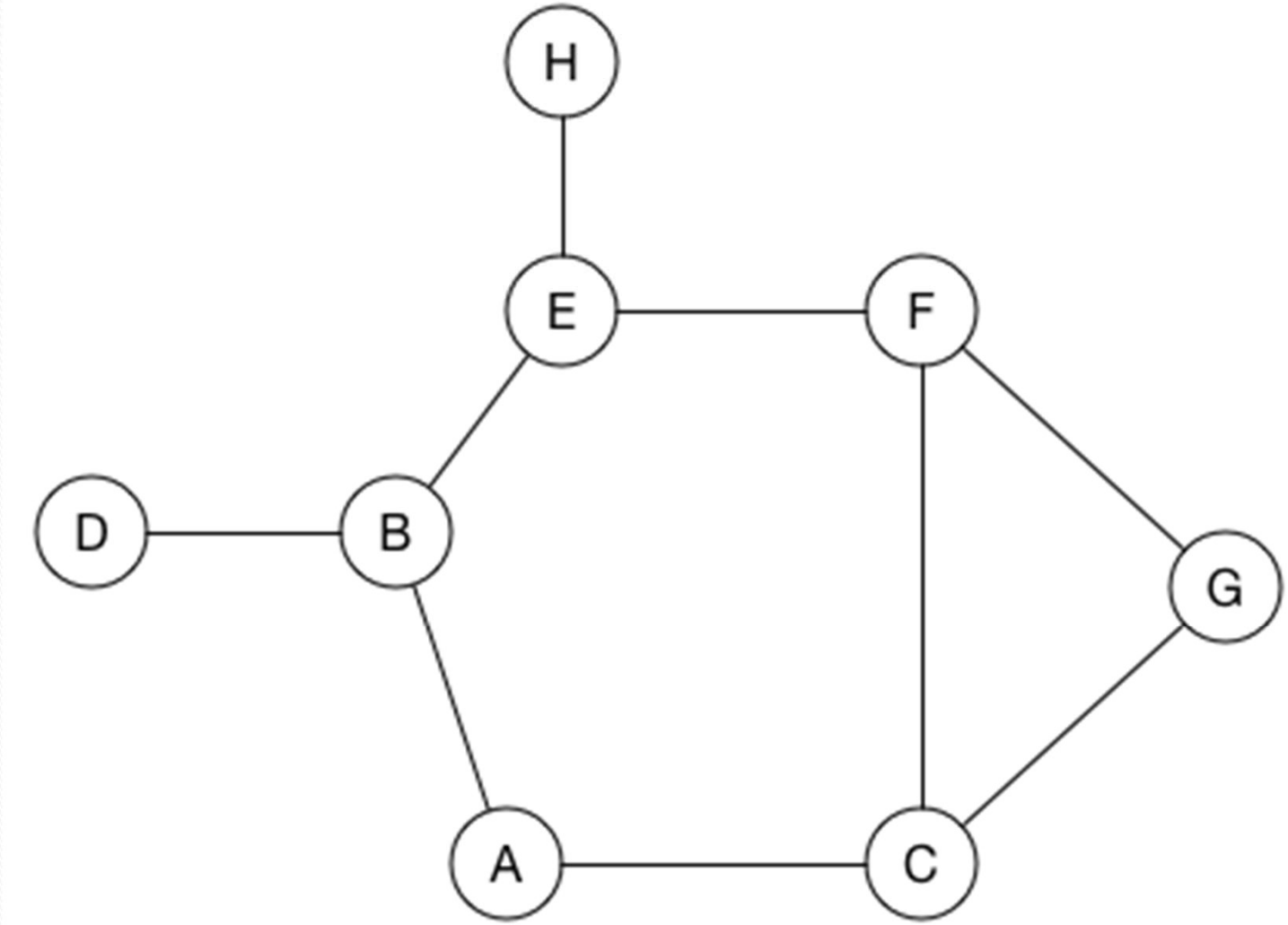
Ejemplo de recorrido Best First

- *Cambia la trayectoria según la heurística*

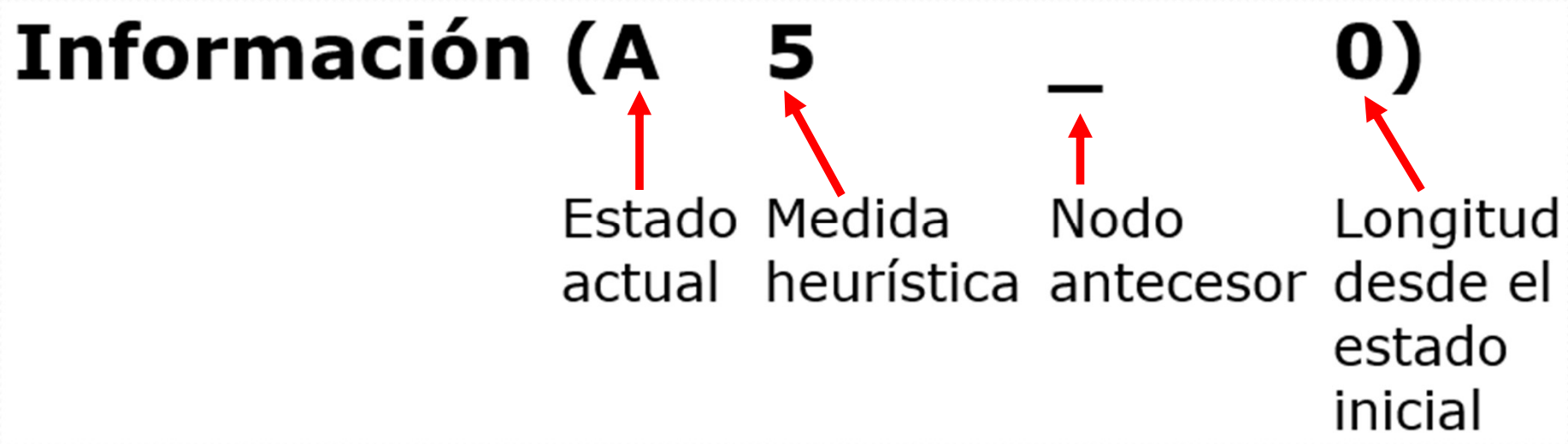


Ejemplo de recorrido Best First

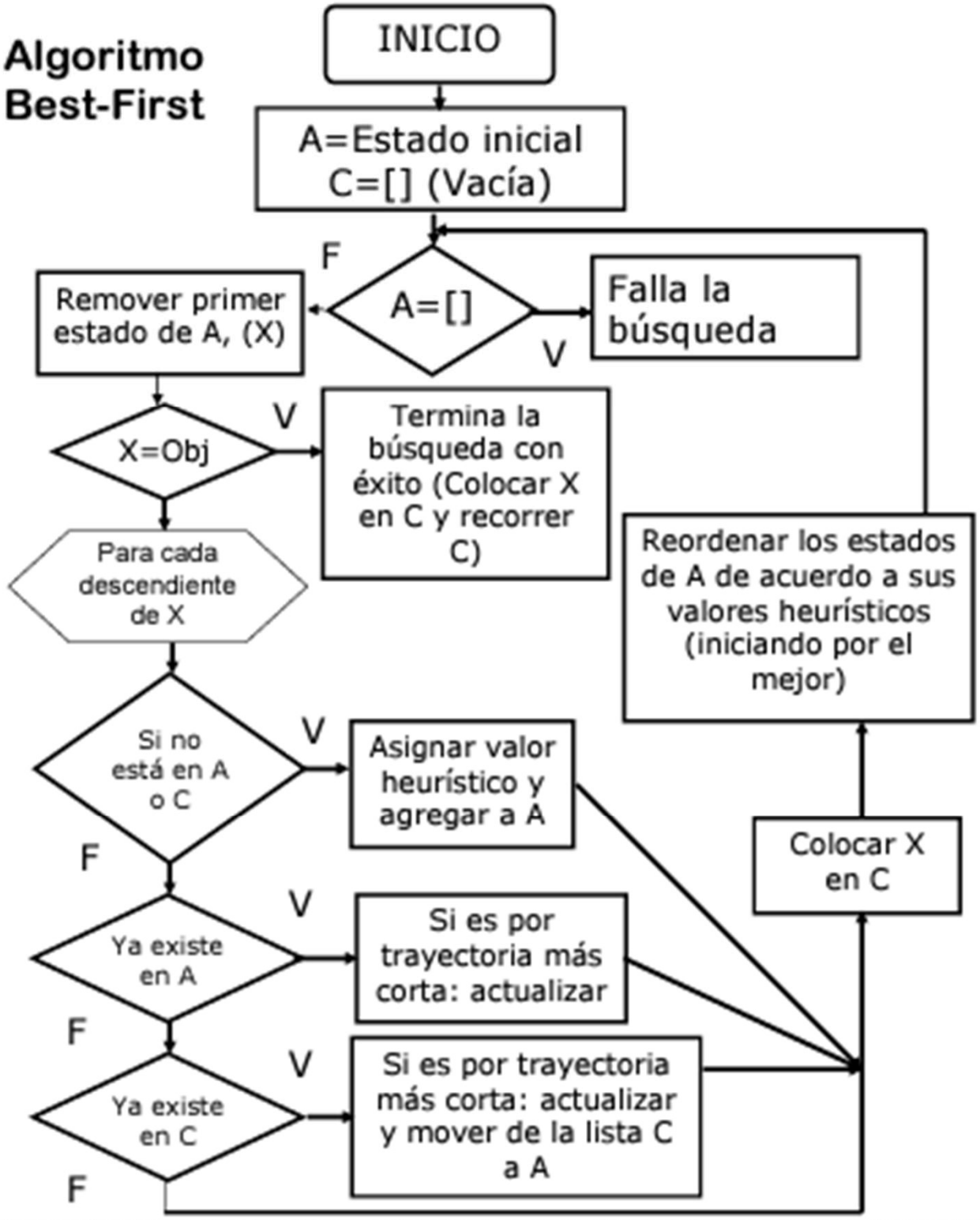
- *Cambia la trayectoria según la heurística*



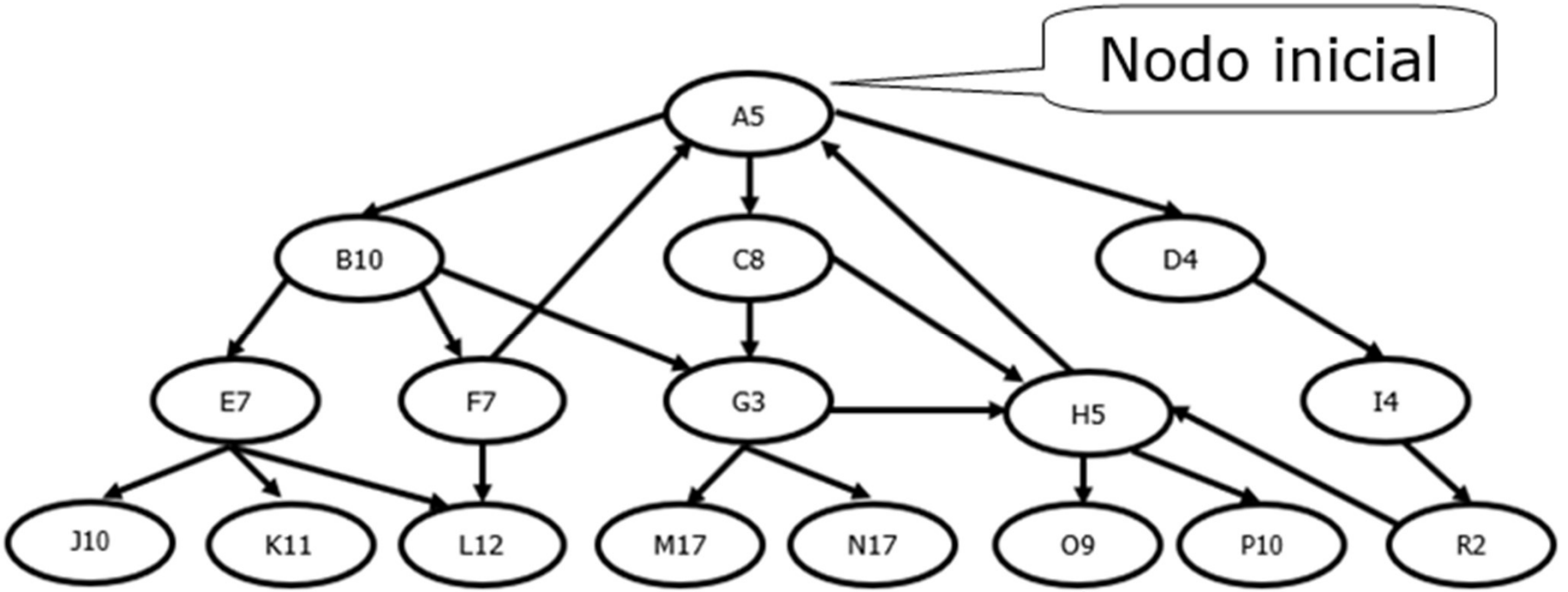
Formato de registro de los nodos visitados



Algoritmo Best-First



Ejercicio Algoritmo Best First



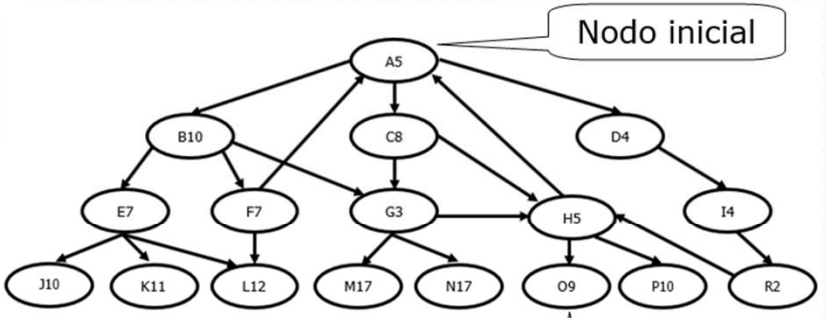
Resuelva este ejercicio aplicando minimización (heurística menor)

Algoritmo Best-First

Algoritmo

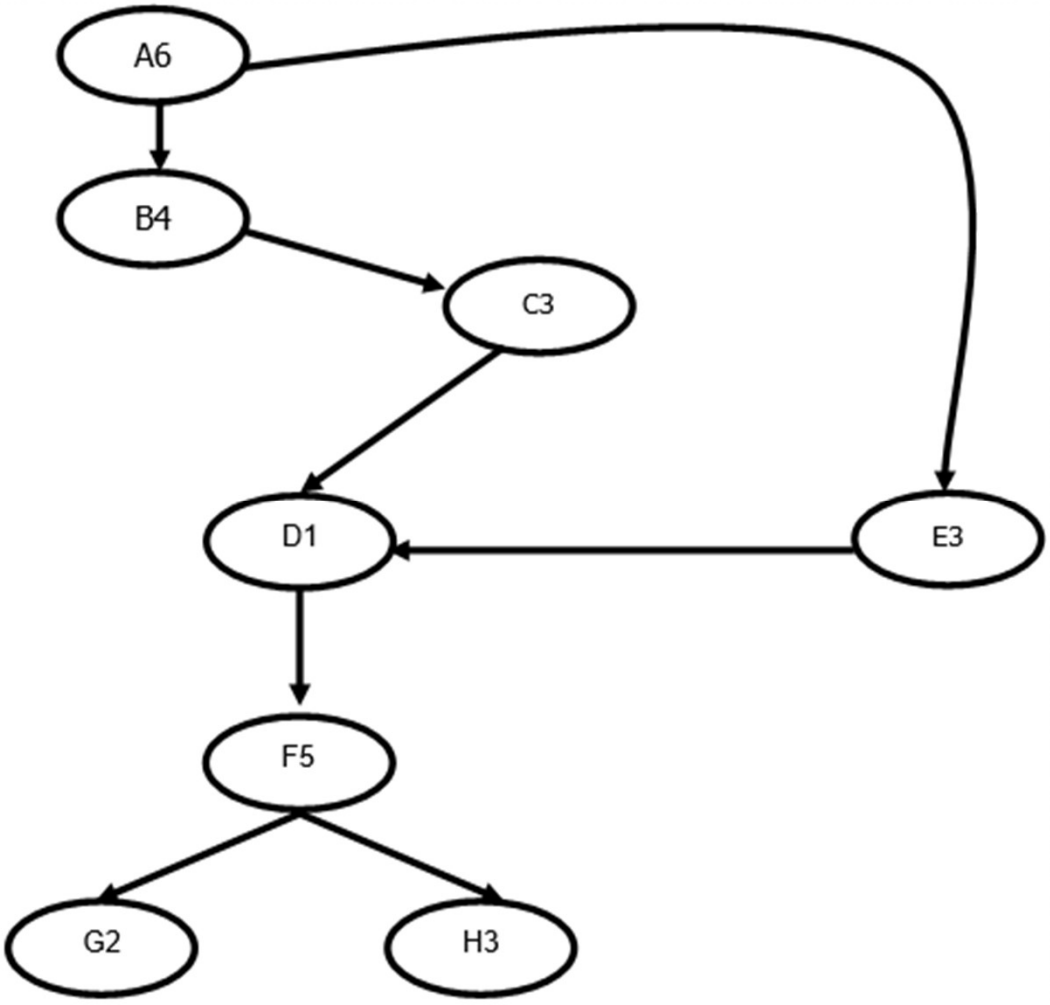
Best First

Utilice este grafo para encontrar el camino del nodo A a O



Resuelva este ejercicio aplicando minimización (heurística menor)

Estrategia Algoritmo Best First

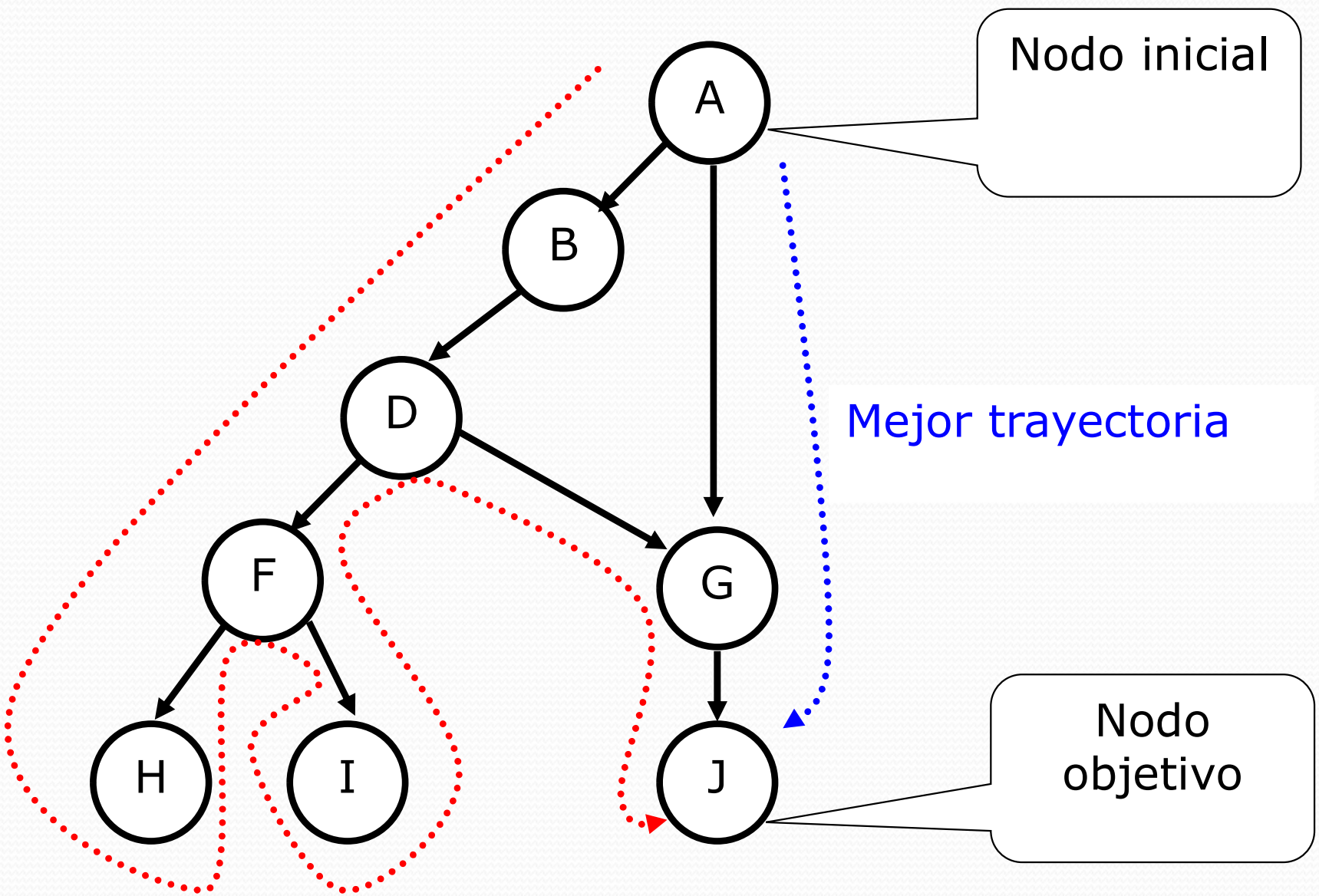


Cuando el nodo alcanzado esta previamente almacenado en la lista C, no importa que sus nodos descendientes ya estén generados, solo se actualiza la mejor trayectoria.

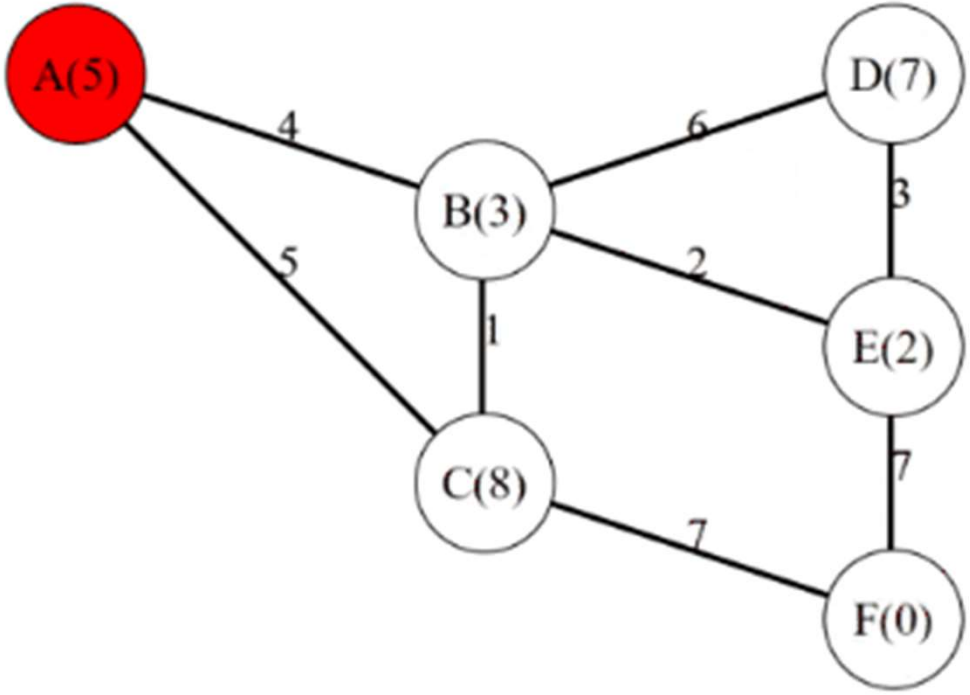
Estrategia Algoritmo Best First

Si un nuevo nodo generado ya está previamente almacenado en la estructura A, no debe ser agregado nuevamente; sin embargo, si la longitud asociada es menor, significa que se alcanzó el mismo estado por una mejor trayectoria, por lo que debe ser considerado.

Estrategia Algoritmo Best First



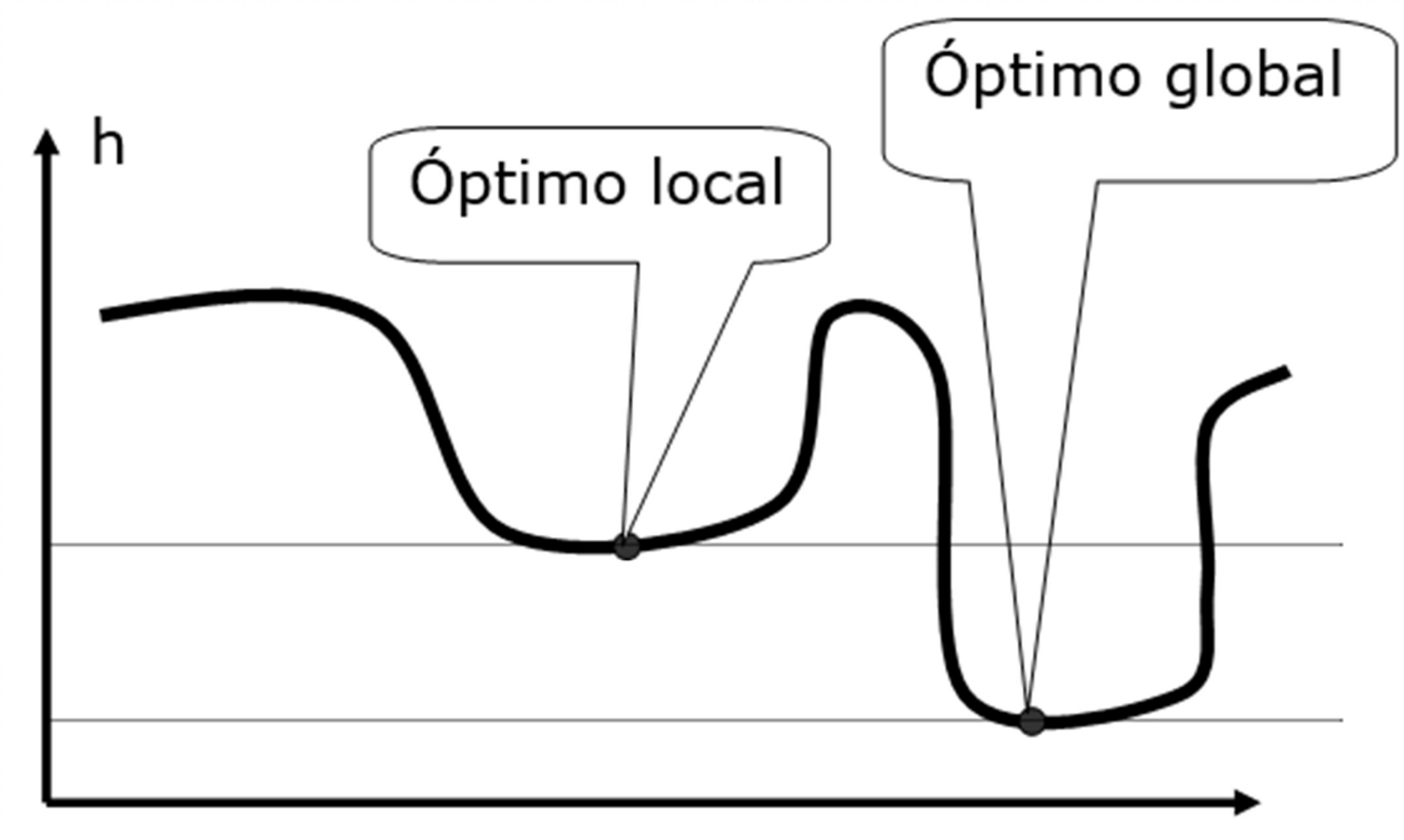
Estrategia Algoritmo Best First



Estrategia Algoritmo Best First

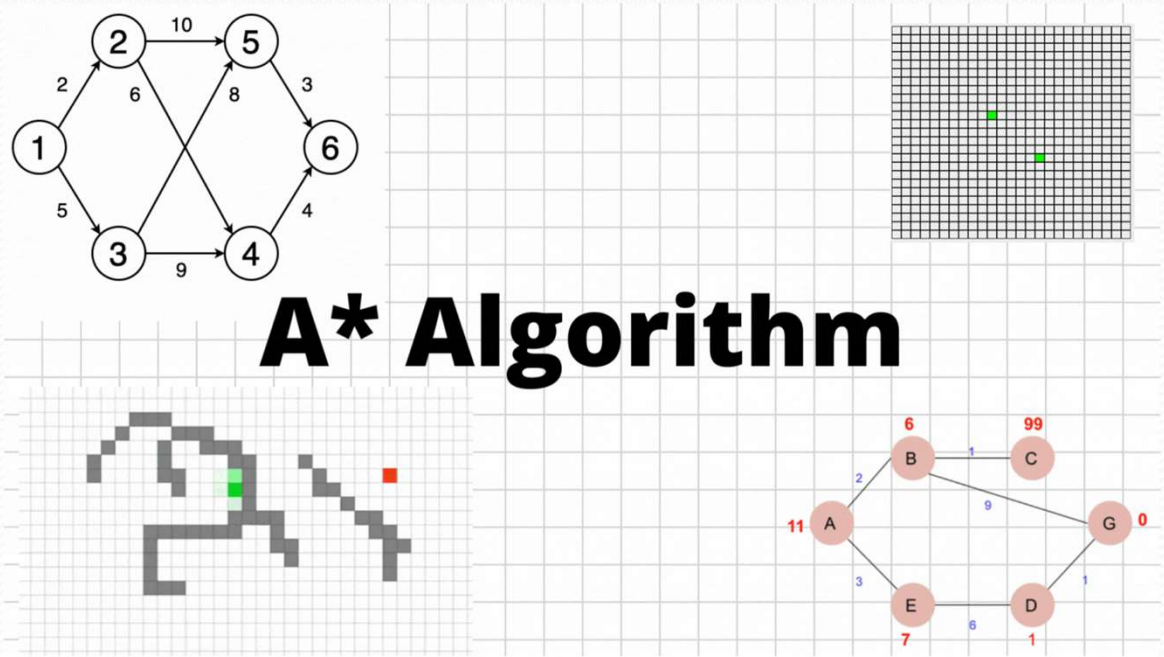
- *Si un nuevo nodo generado ya está previamente almacenado en la lista C, significa que se volvió a llegar a un estado que ya se había analizado.*
- *Sin embargo, si la trayectoria por la cual se volvió a alcanzar este estado es mejor, se debe considerar al formar la trayectoria solución, sin importar que sus nodos descendientes ya estén generados*

Estrategia Algoritmo Best First



Algoritmo A*

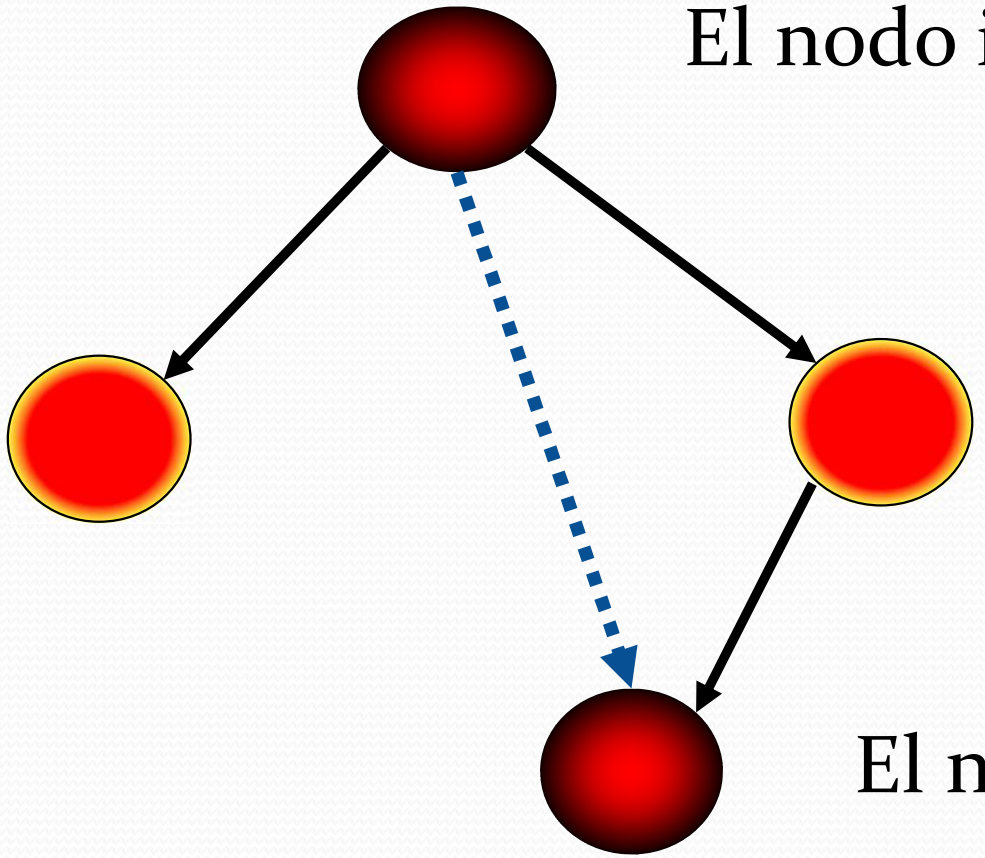
- *Algoritmo que encuentra el camino más corto entre nodos*
- *Utiliza información de los pesos*



Información requerida

- *El algoritmo A* requiere:*

El conjunto de reglas (distancia en línea recta)

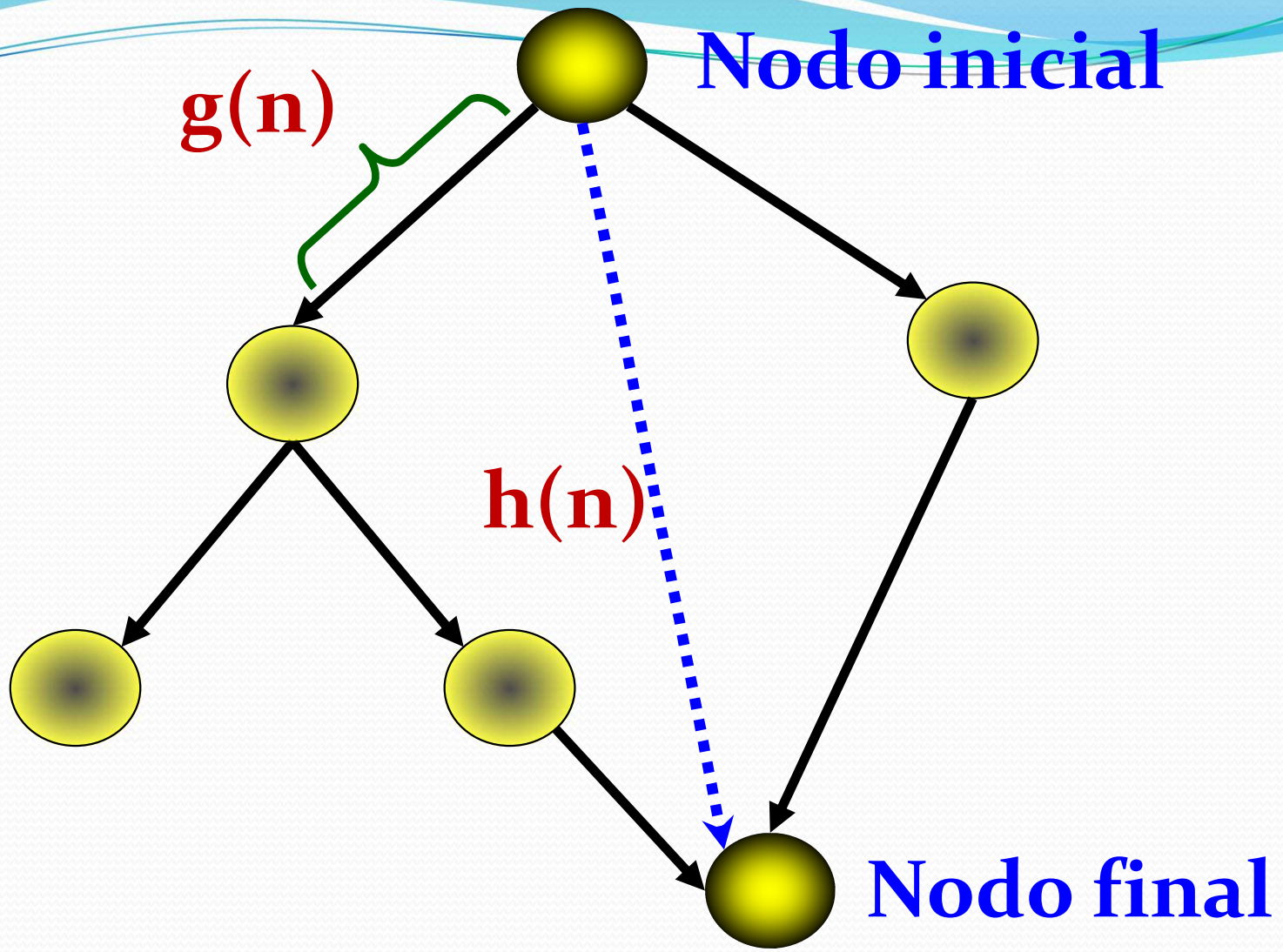


Algoritmo A*

- *Es un algoritmo de búsqueda preferente por lo mejor (best-first) en el que se utiliza f como función heurística y una función h aceptable.*
- *Es un algoritmo genérico de búsqueda.*
- *Basa su comportamiento en una función de evaluación.*

Función de evaluación

- *Esta función se encuentra compuesta por la siguiente combinación:*
- *La búsqueda por costo uniforme, reduce al mínimo el costo de la ruta, g(n).*
- *La búsqueda avara permite reducir al mínimo el costo de la meta, h(n).*
  - $f(n) = g(n) + h(n)$

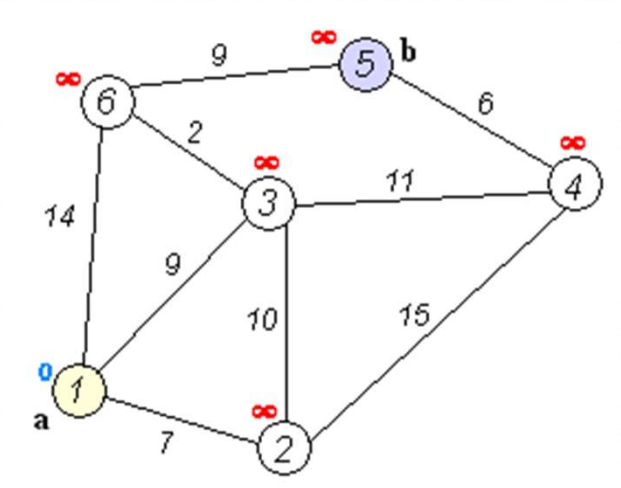


f(n) = costo estimado de la solución más barata, pasando por n.

Estrategia del algoritmo A*

Esta función tiene dos principios:

1. *Lo más corto es lo mas rápido (h)*
2. *Para asegurarnos que es la mejor opción hay que agregar subestimaciones (g)*



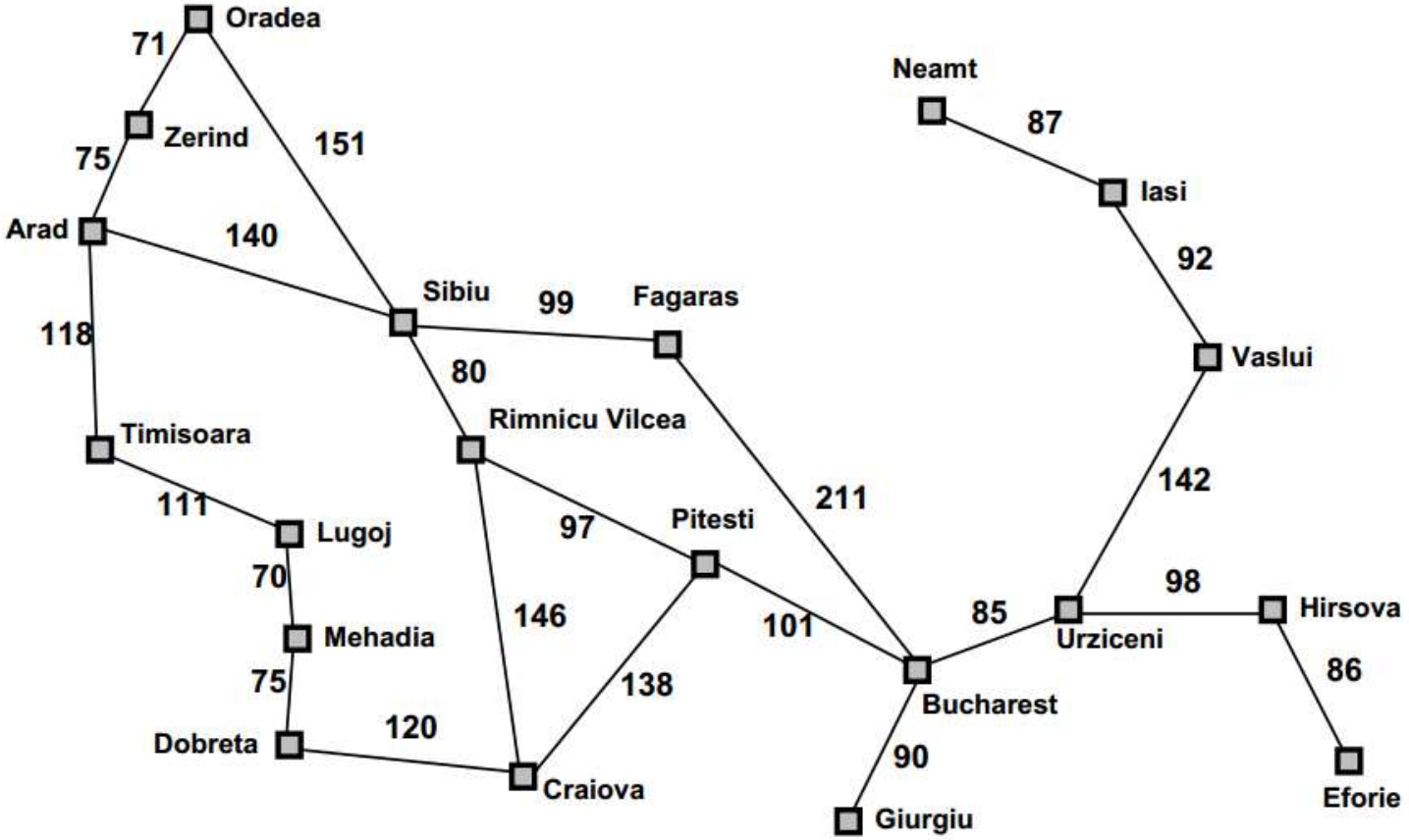
Estructuras utilizadas

Estructura abierta: *contiene los nodos que podrían formar parte del camino*

Lista cerrada: *contiene los nodos que ya han sido examinados y que no hace falta volver a examinar*

Ejemplo

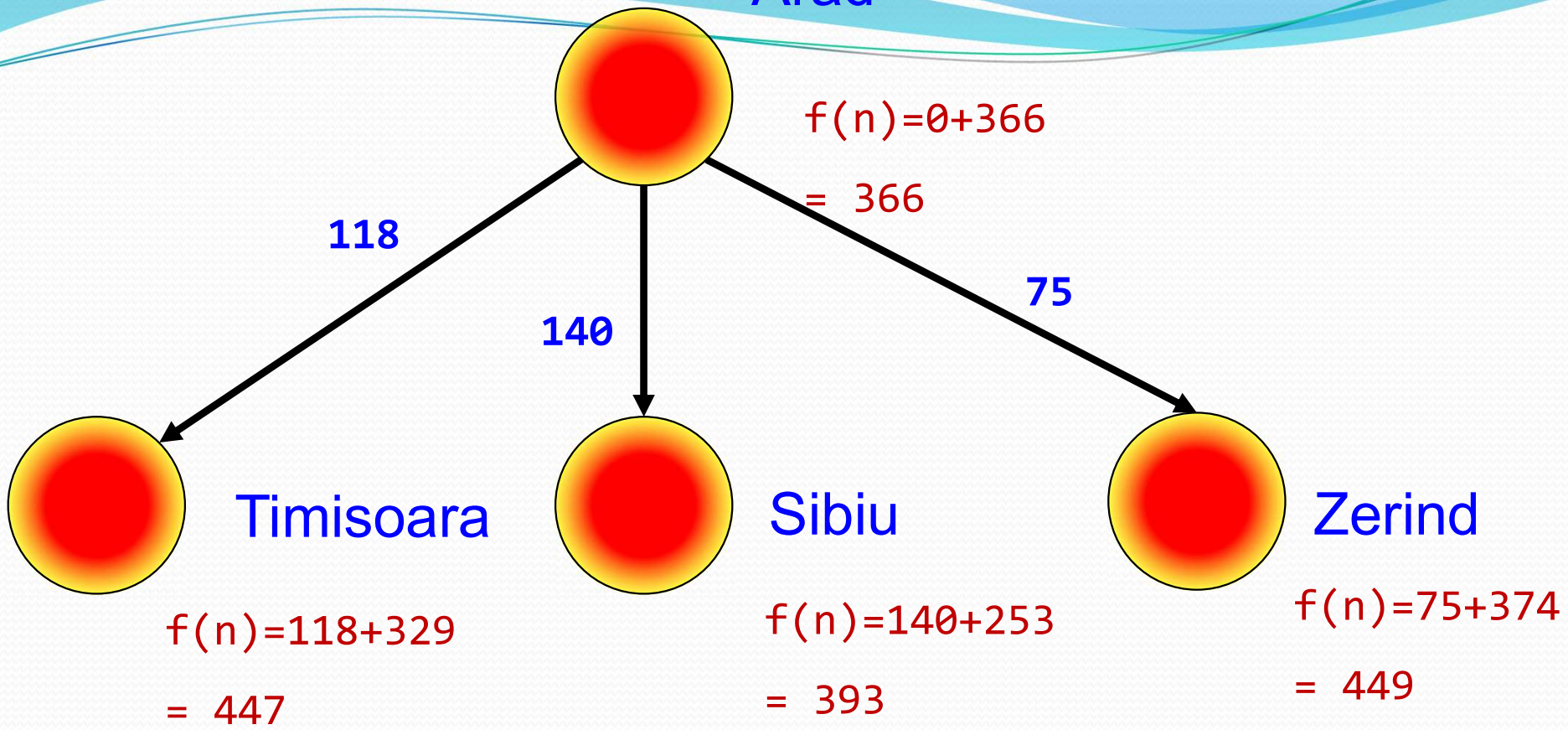
Mapa de Rumania (distancia en kms)

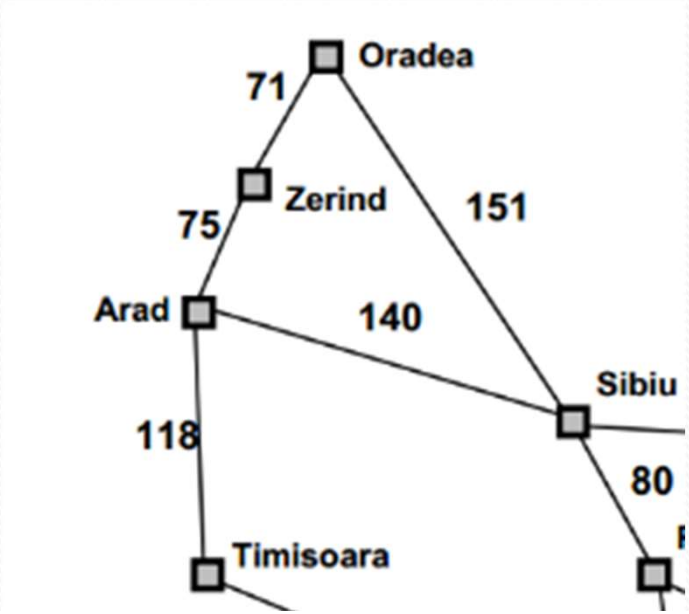


Distancia en línea recta a Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

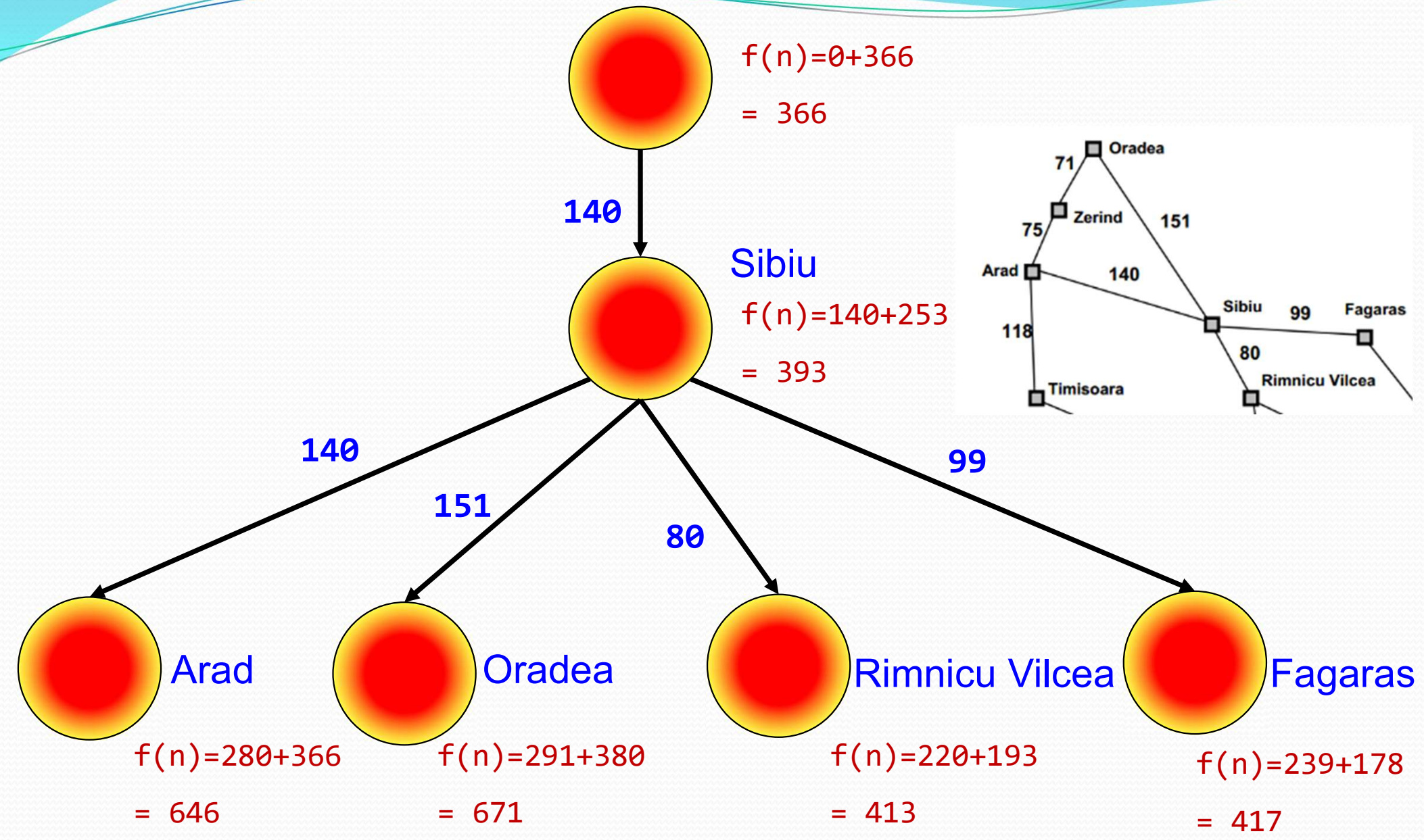
Distancia en línea recta a Bucarest

Arad	366
Bucharest	0
Craiova	160
Fagaras	178
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Zerind	374

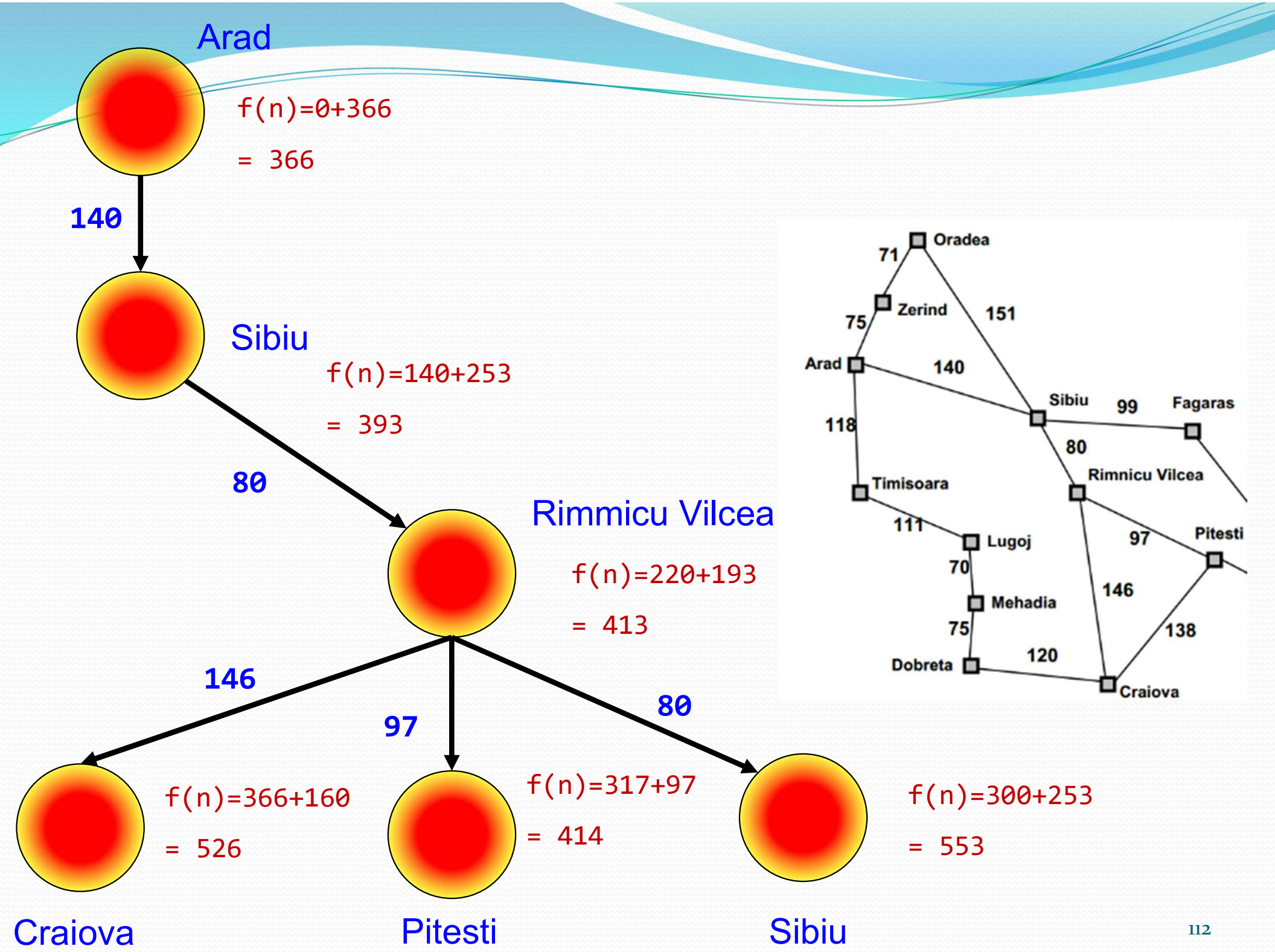

Arad
f(n)=0+366
= 366
118
140
75
Timisoara
Sibiu
Zerind
f(n)=118+329
= 447
f(n)=140+253
= 393
f(n)=75+374
= 449


Oradea
71
Zerind
75
151
Arad
140
Sibiu
118
80
Timisoara

Arad


f(n)=0+366
= 366
140
Sibiu
f(n)=140+253
= 393
140
151
80
99
Arad
f(n)=280+366
= 646
Oradea
f(n)=291+380
= 671
Rimnicu Vilcea
f(n)=220+193
= 413
Fagaras
f(n)=239+178
= 417

Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Timisoara
Rimnicu Vilcea


Arad
f(n)=0+366
= 366
140
Sibiu
f(n)=140+253
= 393
80
Rimmicu Vilcea
f(n)=220+193
= 413
146
97
80
f(n)=366+160
= 526
f(n)=317+97
= 414
f(n)=300+253
= 553
Craiova
Pitesti
Sibiu
Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
70
146
Mehadia
75
138
120
Dobreta
Craiova

f(n)=0+366

= 366

Arad

140

Sibiu

f(n)=140+253

= 393

80

Rimmicu Vilcea

f(n)=220+193

= 413

97

Pitesti

f(n)=317+98

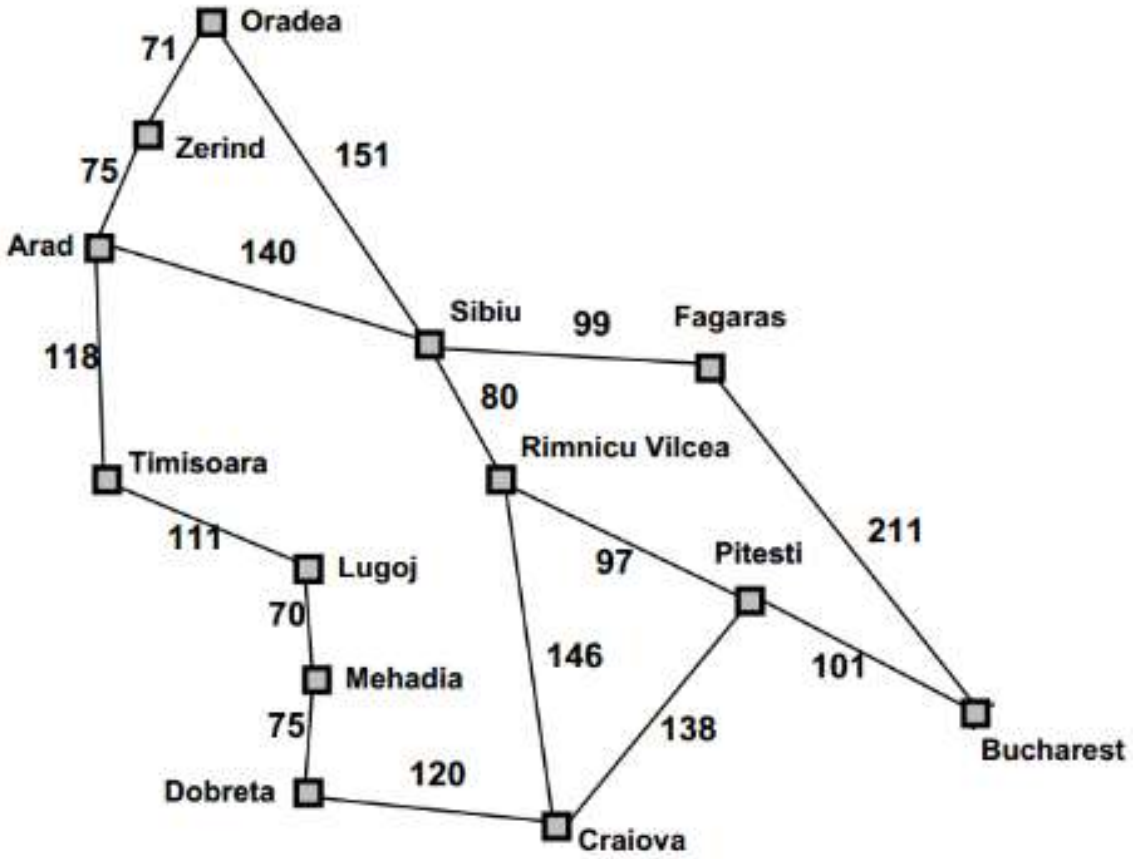
= 415

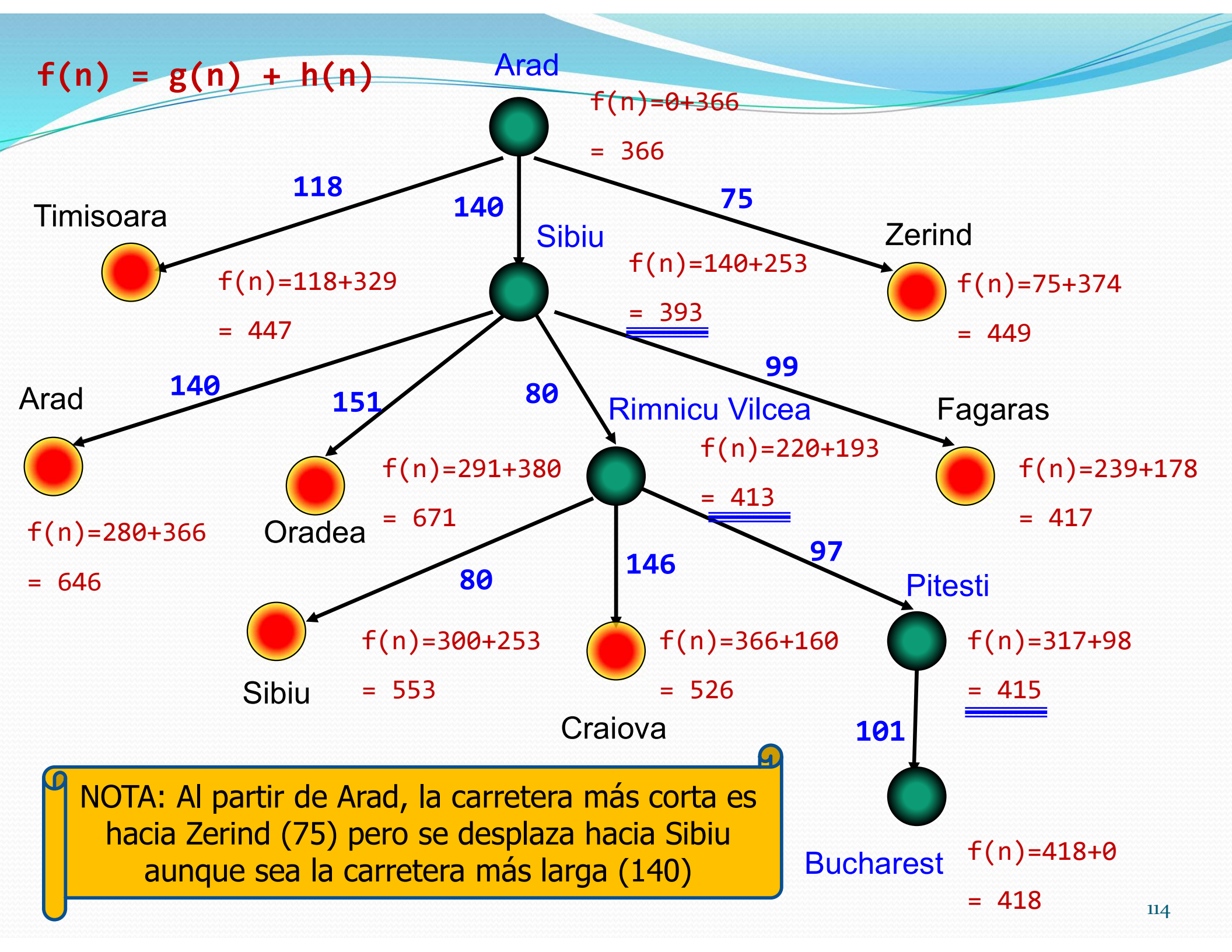
101

Bucharest

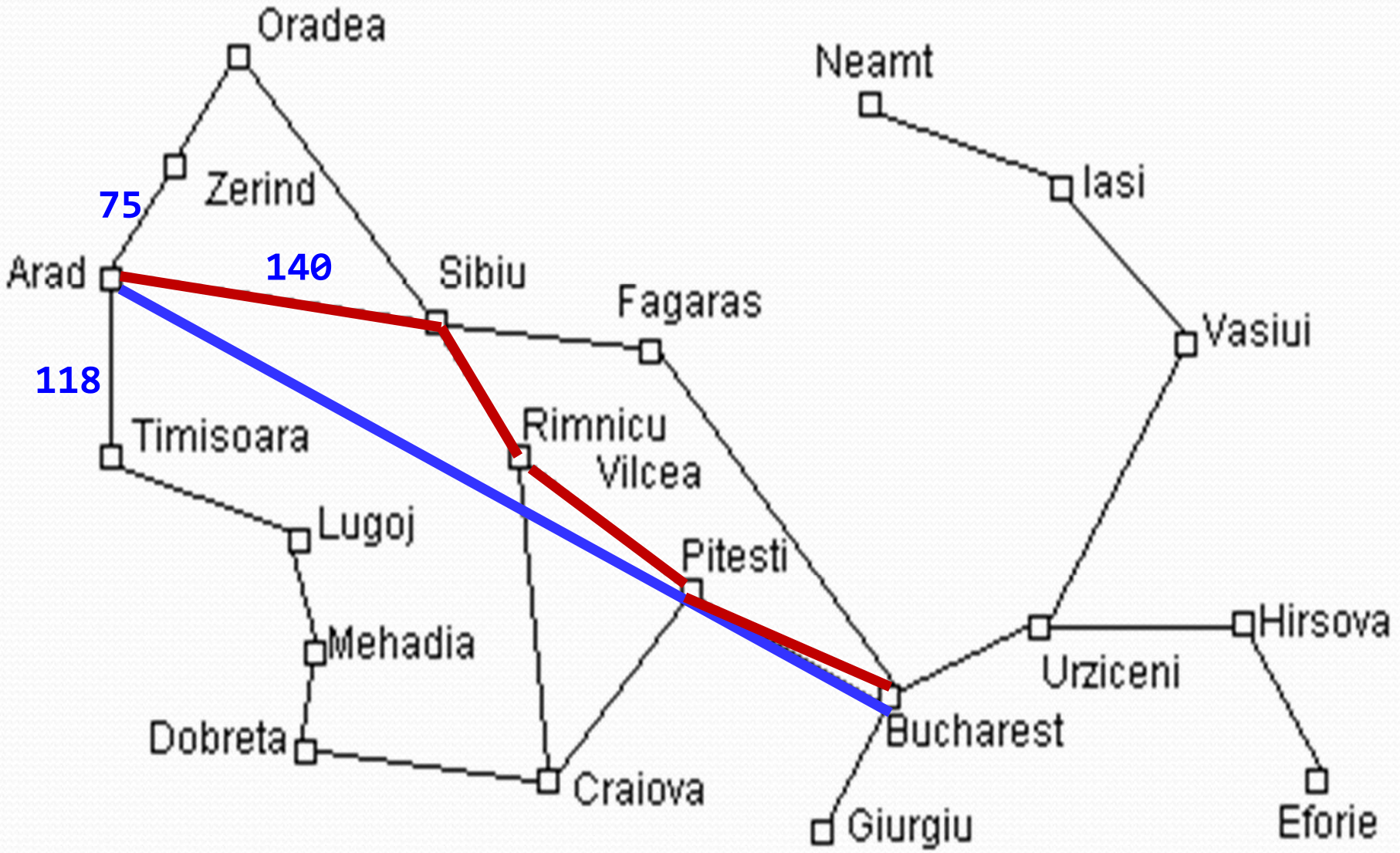
f(n)=418+0

= 418



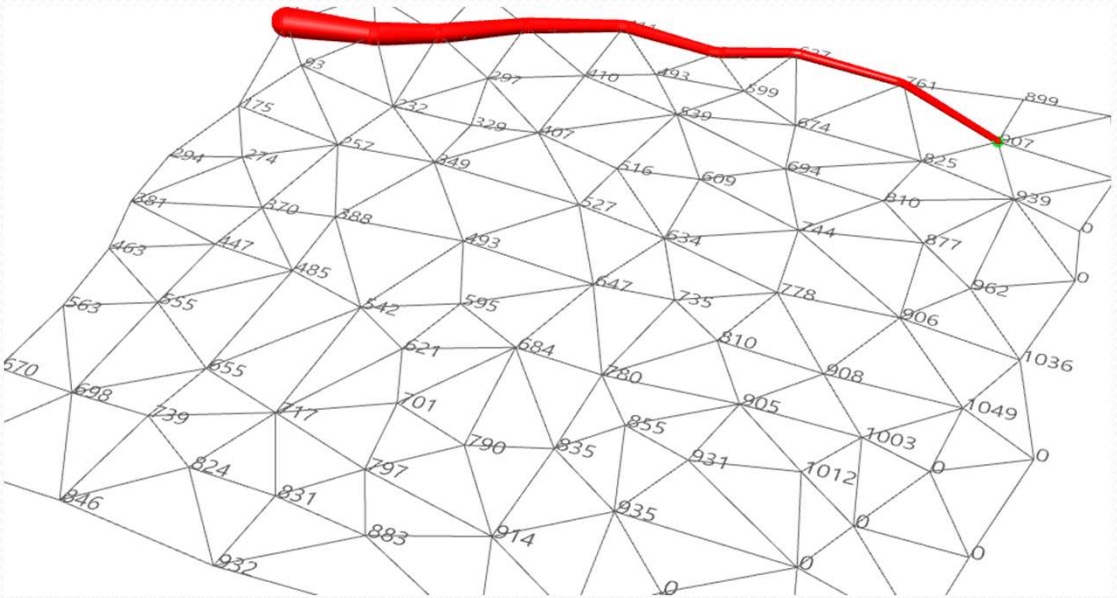

f(n) = g(n) + h(n)
Arad
f(n)=0+366
= 366
118
Timisoara
f(n)=118+329
= 447
140
Sibiu
f(n)=140+253
= 393
75
Zerind
f(n)=75+374
= 449
Arad
140
f(n)=280+366
= 646
151
Oradea
f(n)=291+380
= 671
80
Rimnicu Vilcea
f(n)=220+193
= 413
99
Fagaras
f(n)=239+178
= 417
80
Sibiu
f(n)=300+253
= 553
146
Craiova
f(n)=366+160
= 526
97
Pitesti
f(n)=317+98
= 415
101
Bucharest
f(n)=418+0
= 418
NOTA: Al partir de Arad, la carretera más corta es hacia Zerind (75) pero se desplaza hacia Sibiu aunque sea la carretera más larga (140)

Conclusión

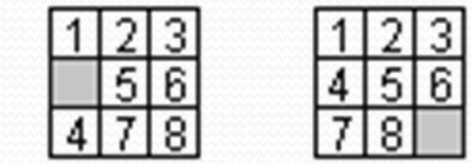

Oradea
Neamt
Zerind
Iasi
75
Arad
140
Sibiu
Fagaras
Vasiui
118
Timisoara
Rimnicu
Vilcea
Lugoj
Pitesti
Hirsova
Mehadia
Urziceni
Dobreta
Bucharest
Craiova
Giurgiu
Eforie

Aplicaciones del algoritmo A*

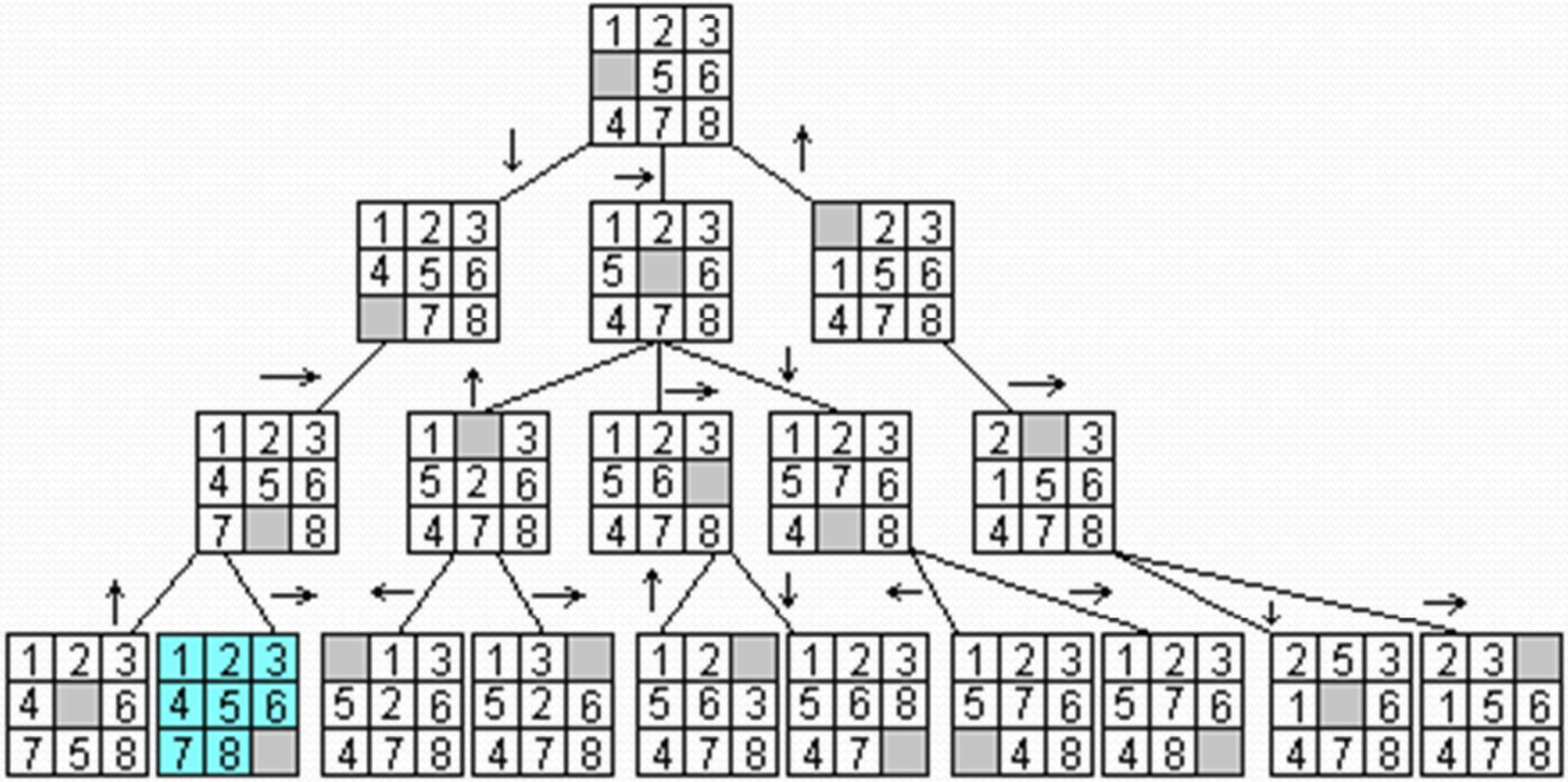
- Mapas
- Juegos
- Robótica



Juegos

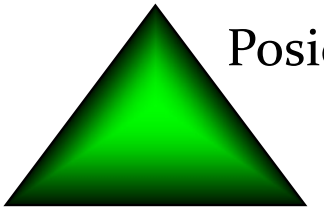


OP1=Despl.blanco a la derecha
OP2=Despl.blanco a la izquierda
OP3=Despl.blanco hacia abajo
OP4=Despl.blanco hacia arriba

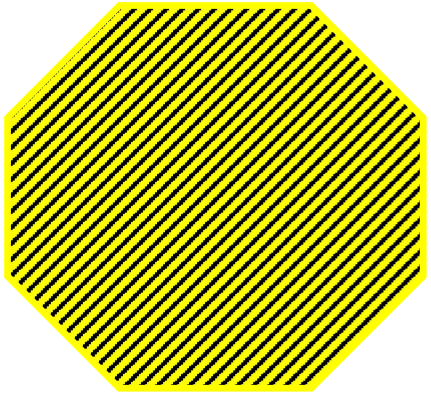


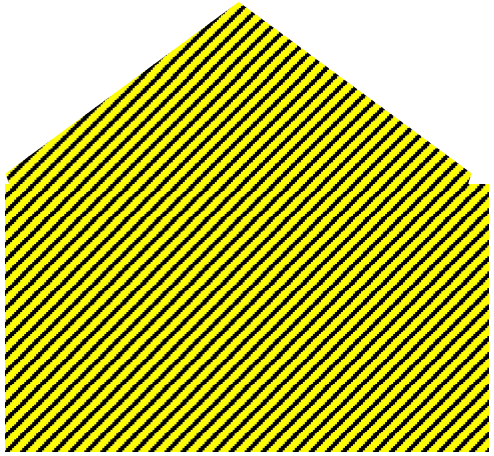
Resolución del 8-puzzle por Búsqueda preferente por profundidad

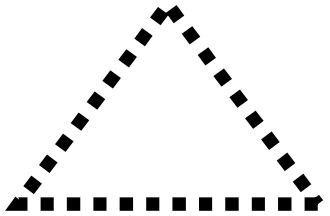
Robótica



Posición inicial

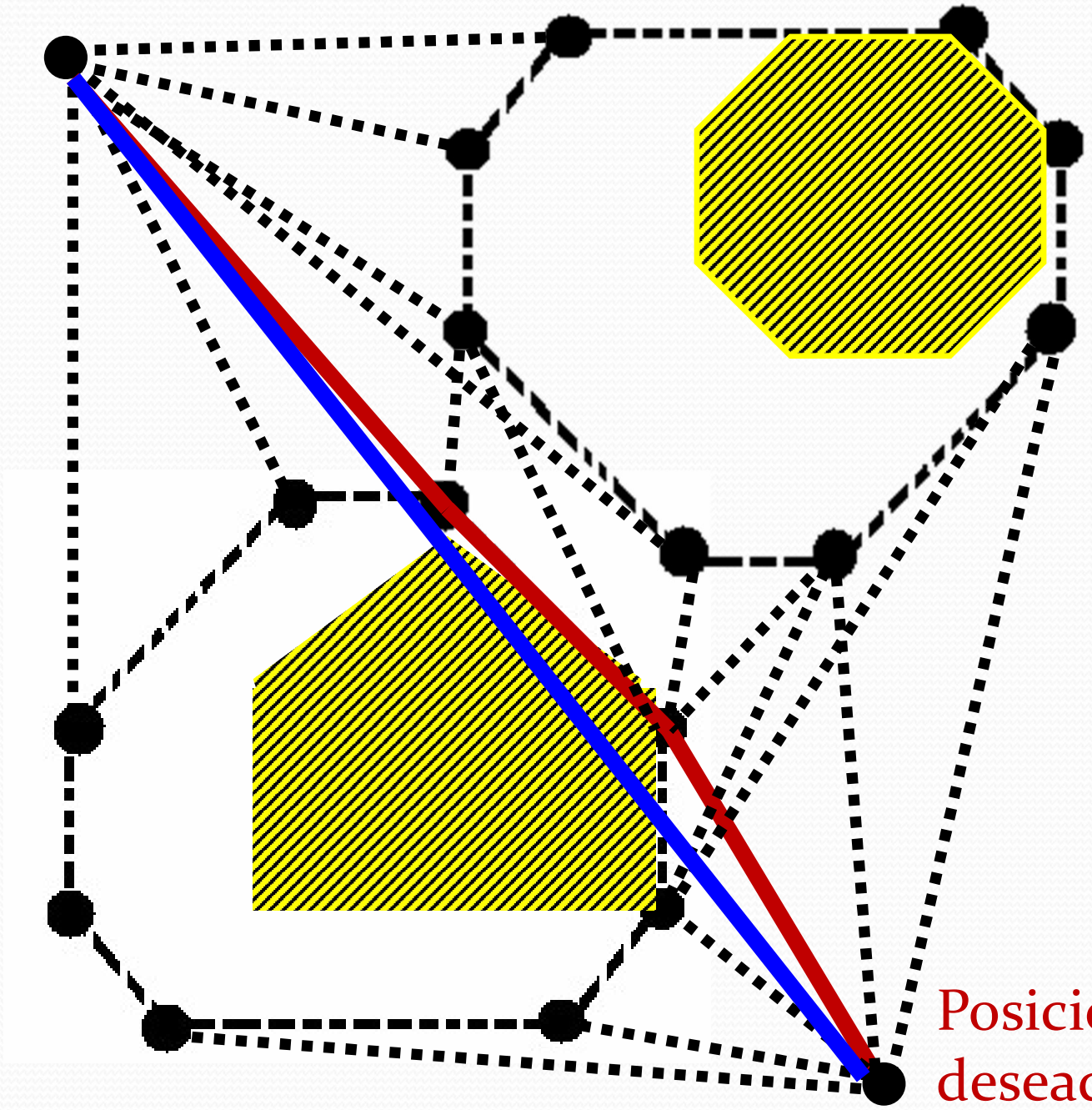






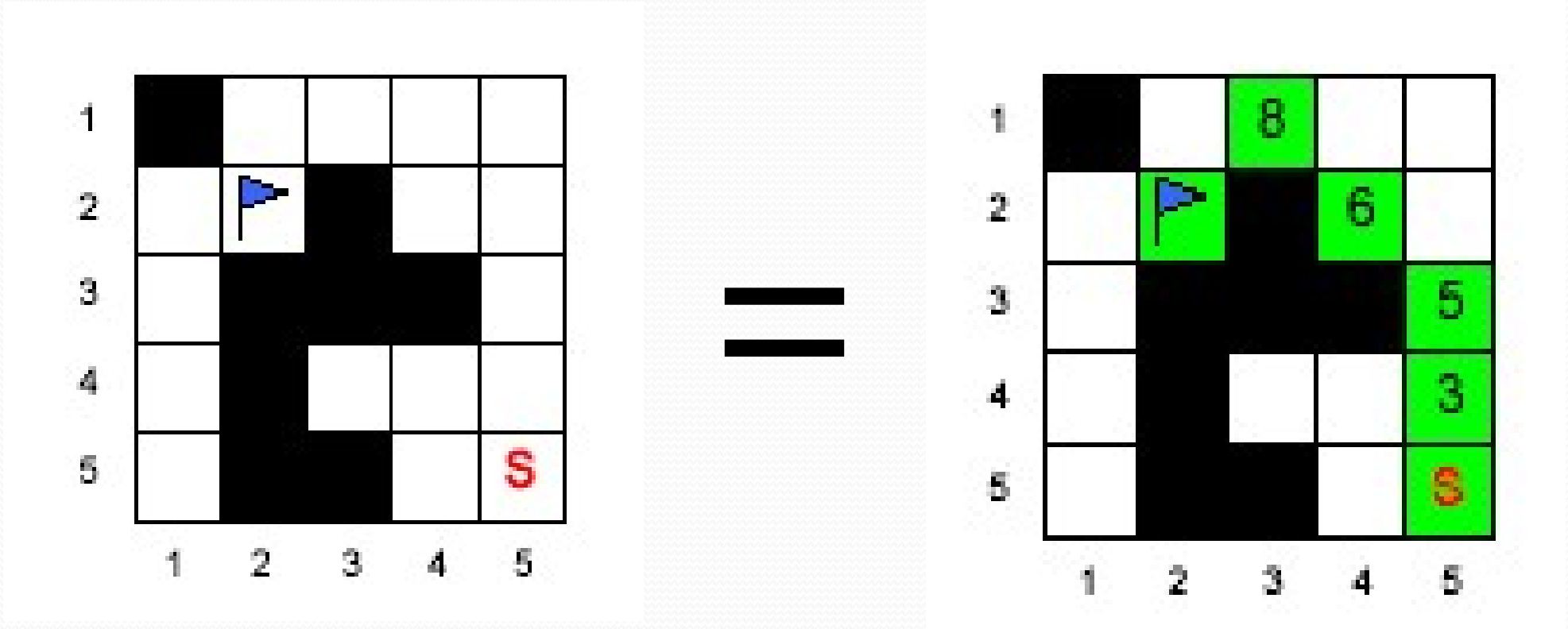
Posición deseada

Posición inicial

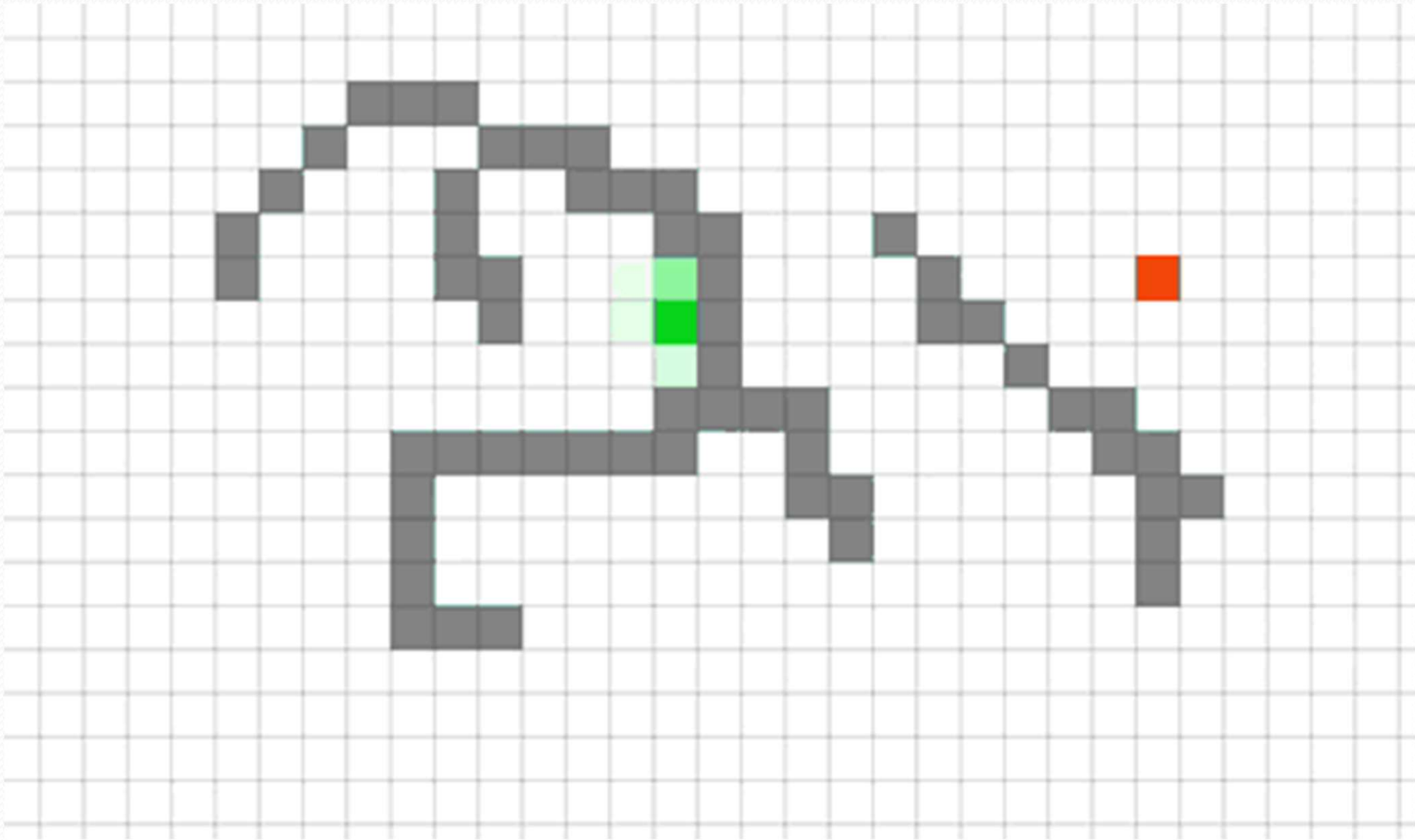


Posición deseada

Búsqueda del camino más corto

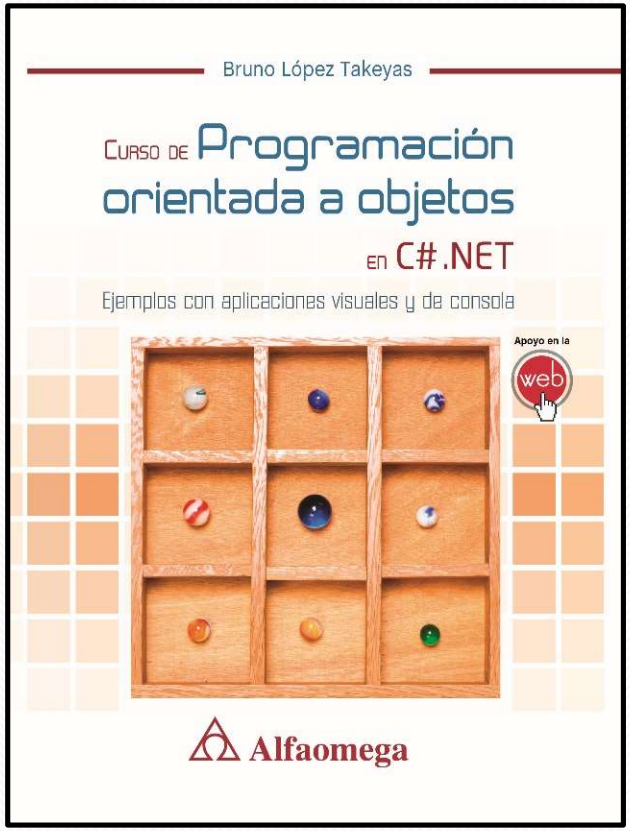


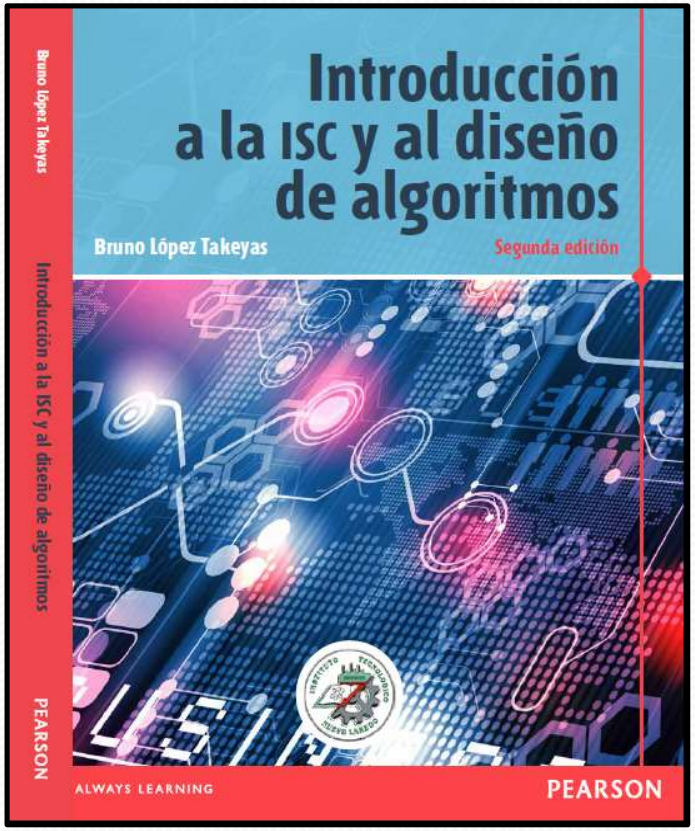
Evitar obstáculos mediante A*

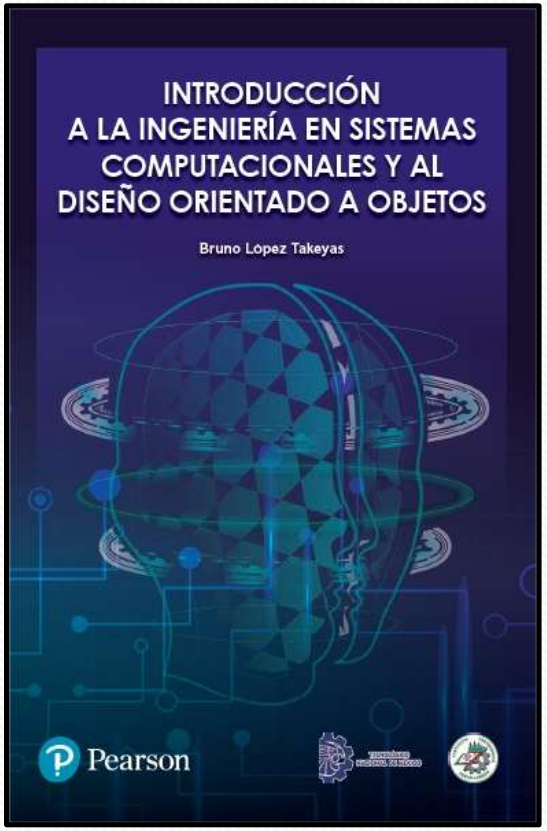


Otros libros del autor

https://nlaredo.tecnm.mx/takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas