REPASO DE ÁRBOLES Y GRAFOS

Árboles binarios

- Un árbol binario es una estructura en la que cada nodo puede tener de 0 a 2 hijos (izquierdo y derecho)
- Los nodos que no tienen hijos se llaman hojas o nodos externos
- Comúnmente se utilizan cómo Árboles de búsqueda o cómo Montículos binarios
- La estructura básica del nodo se compone de un valor y dos punteros a sus hijos
- El punto de inicio de un árbol es el nodo raíz
- Otra forma especializada de árboles binarios son los árboles balanceados o AVL, en los cuales la profundidad de una de las ramas solo puede exceder en uno a la profundidad de la otra rama. Este tipo de árbol se reestructura cada vez que se inserta un nodo

Árboles binarios

- Existen cuatro métodos básicos de recorrido de árboles binarios, agrupados en dos categorías:
  - *Profundidad Primero*
    - PreOrden
      - *Visitar raíz, recorrer rama izquierda, recorrer rama derecha*
    - InOrden
      - *Recorrer rama izquierda, visitar raíz, recorrer rama derecha*
    - PostOrden
      - *Recorrer rama izquierda, recorrer rama derecha, visitar raíz*
  - *Anchura primero*
    - Recorrido por niveles
      - *Visitar primer nivel (raíz), visitar segundo nivel, continuar hasta llegar a las hojas*

Ejemplo en VB.NET

Nodo de árbol binario

```
Public Class Nodo
    Public Property Derecha As Nodo
    Public Property Izquierda As Nodo
    Public Property Valor As Integer


    Public Sub New()
    End Sub

    Public Sub New(valor As Integer)
        Me.Valor = valor
    End Sub

End Class
```

Nodo de árbol AVL

```
Public Class AVL
    Public Property Valor As Integer
    'FB es la altura del subarbol izquierdo menos la altura del subarbol derecho
    Public Property FB As Integer
    Public Property Izquierda As AVL
    Public Property Derecha As AVL
    Public Property Borrado As Boolean

    Public Sub New()
        FB = 0
        Borrado = 0
    End Sub
```

Ejemplo en VB.NET

```vb
Private Sub RecorrerPreOrden(nodo As AVL)
    If nodo Is Nothing Then
        Return
    End If

    Console.Write(nodo.Valor & " - ")
    RecorrerPreOrden(nodo.Izquierda)
    RecorrerPreOrden(nodo.Derecha)
End Sub
```

```vb
Private Sub RecorrerInOrden(nodo As AVL)
    If nodo Is Nothing Then
        Return
    End If

    RecorrerInOrden(nodo.Izquierda)
    Console.Write(nodo.Valor & " - ")
    RecorrerInOrden(nodo.Derecha)
End Sub
```

```vb
Private Sub RecorrerPostOrden(nodo As AVL)
    If nodo Is Nothing Then
        Return
    End If

    RecorrerPostOrden(nodo.Izquierda)
    RecorrerPostOrden(nodo.Derecha)
    Console.Write(nodo.Valor & " - ")
End Sub
```

Ejemplo en VB.NET

Recorrido por niveles con colas

```
Public Sub RecorrerPorNiveles(ByRef raiz As Nodo)
    Dim cola As New Queue(Of Nodo)
    Dim nodo As Nodo
    cola.Enqueue(raiz)
    While cola.Count > 0
        nodo = cola.Dequeue
        Console.Write(nodo.Valor.ToString & " - ")

        If nodo.Izquierda IsNot Nothing Then
            cola.Enqueue(nodo.Izquierda)
        End If

        If nodo.Derecha IsNot Nothing Then
            cola.Enqueue(nodo.Derecha)
        End If
    End While
    Console.WriteLine()
End Sub
```

Ejemplo en VB.NET

Recorrido por niveles con listas

```vb
Private Sub PorNivel(raiz As Nodo)
    ' Se invoca la función recursiva enviando una lista de solo un elemento
    RecorrerPorNivel(New List(Of Nodo)({raiz}))
End Sub

''' <summary> En el primer llamado solo tiene el nodo raiz, entonces crea ur
Private Sub RecorrerPorNivel(nodos As List(Of Nodo))
    Dim siguienteNivel As New List(Of Nodo)
    For Each nodo As Nodo In nodos
        Console.Write(nodo.Valor.ToString & " - ")

        If nodo.Izquierda IsNot Nothing Then
            siguienteNivel.Add(nodo.Izquierda)
        End If

        If nodo.Derecha IsNot Nothing Then
            siguienteNivel.Add(nodo.Derecha)
        End If
    Next
    Console.WriteLine()
    If siguienteNivel.Count > 0 Then
        RecorrerPorNivel(siguienteNivel)
    End If
End Sub
```

Grafos

- Un grafos es una estructura en donde los nodos o vértices tienen relaciones aleatorias entre sí
- Las relaciones se conocen cómo aristas o arcos y opcionalmente pueden tener dirección y peso
  - *En un grafo no dirigido las aristas no tienen dirección*
  - *En un grafo dirigido las aristas si tienen dirección*
- Si un nodo se relaciona consigo mismo se dice que tiene una relación recursiva
- Los grafos tienen diferentes aplicaciones en la computación y la matemática y existe toda una teoría sobre grafos
  - *Pueden utilizarse para representar mapas y recorridos*
  - *En programación pueden usarse para representar el flujo de un programa y a partir de ahí calcular la complejidad ciclomática del mismo*

Grafos

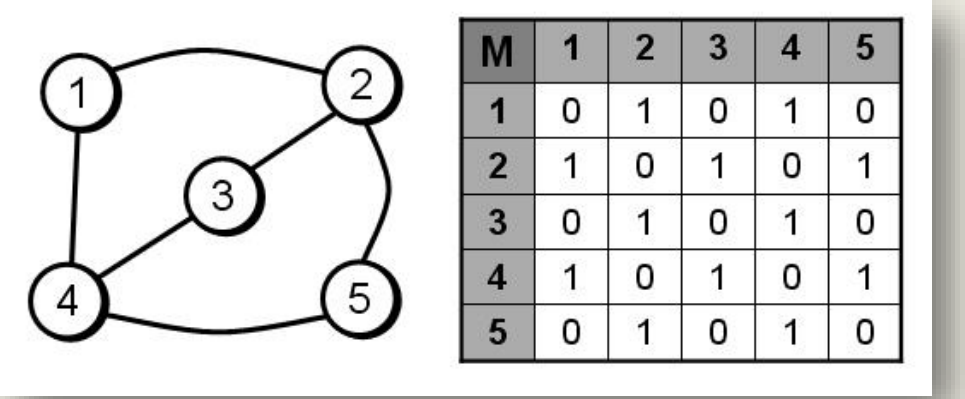
- Propiedades
  - *Adyacencia*
    - Dos vértices son adyacentes si una arista los unes
    - Dos aristas son adyacentes si tienen un vértice en común
  - *Incidencia*
    - Una arista es *incidente* a un vértice si ésta lo une a otro
  - *Ponderación*
    - Corresponde a una función que a cada arista le asocia un valor (costo, peso, longitud, etc.), para aumentar la expresividad del modelo. Esto se usa mucho para problemas de optimización, como el del vendedor viajero o del camino más corto.
  - *Etiquetado*
    - Distinción que se hace a los vértices y/o aristas mediante una marca que los hace unívocamente distinguibles del resto.

Representación de grafos

- Los grafos pueden representarse de varias maneras
  - *Matriz de adyacencia (MA)*
    - Las filas y columnas hacen referencia a los vértices y las posiciones hacen referencia a las aristas
    - Por ejemplo MA(1, 2) = 4 indica que hay una arista de peso 4 entre el vértice 1 y el 2
  - *Lista de adyacencia (LA)*
    - Se utiliza un vector de tamaño *n* (un elemento por cada vértice) donde **LA**[i] almacena la referencia a una lista de los vértices adyacentes a i. En una red esta lista almacenará también la longitud de la arista que va desde i al vértice adyacente
  - *Listas de nodos*
    - Se utiliza un objeto nodo que tiene una lista de los nodos conectados a el
    - Es muy similar a un árbol. De hecho los árboles son un subtipo de grafos

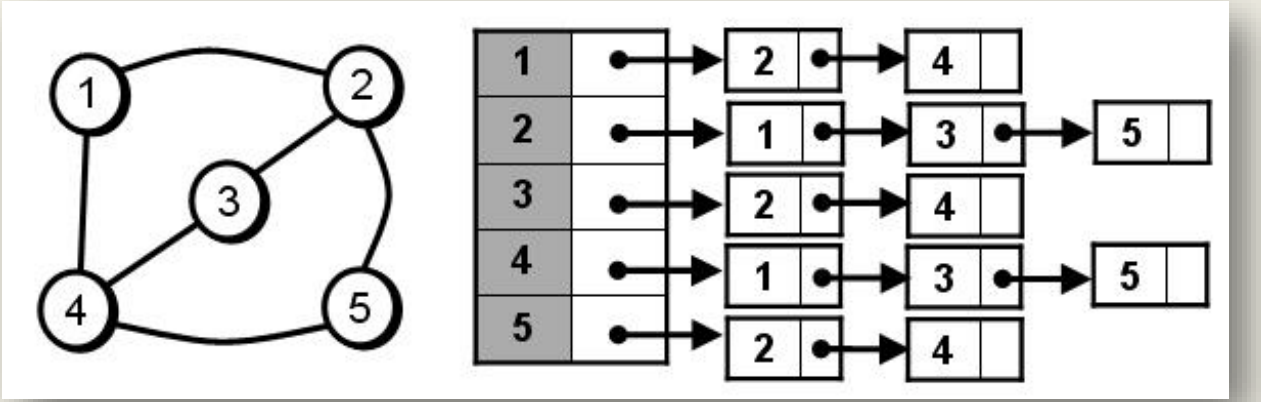
Representación de grafos

Matriz de adyacencia



M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Lista de adyacencia



Algoritmos usados en grafos

- Recorridos
  - *Los recorridos de grafos deben considerar que hay relaciones cíclicas o recursivas por lo que usualmente emplean una lista para almacenar los valores que ya visitaron*
  - *Recorrido en profundidad*
    - Similar al recorrido en PostOrden de los árboles binarios
  - *Recorrido de anchura*
    - Similar al recorrido de anchura de los árboles binarios

Algoritmos usados en grafos

Recorrido en profundidad

```
Public Shared Sub RecorridoEnProfundidad(nodo As Nodo,
                                         ByRef visitados As List(Of Integer))
    For Each hijo In nodo.Hijos
        If Not visitados.Contains(hijo.Valor) Then
            visitados.Add(nodo.Valor)
            RecorridoEnProfundidad(hijo, visitados)
        End If
    Next
    Console.Write(nodo.Valor & " - ")
End Sub
```

Recorrido en anchura

```
Public Shared Sub RecorridoPorAnchura(raiz As Nodo,
                                      ByRef visitados As List(Of Integer))
    Dim cola As New Queue(Of Nodo)
    Dim nodo As Nodo
    cola.Enqueue(raiz)
    visitados.Add(raiz.Valor)
    While cola.Count > 0
        nodo = cola.Dequeue
        Console.Write(nodo.Valor.ToString & " - ")
        For Each hijo In nodo.Hijos
            If Not visitados.Contains(hijo.Valor) Then
                visitados.Add(hijo.Valor)
                cola.Enqueue(hijo)
            End If
        Next
    End While
End Sub
```

LABORATORIO

Laboratorio árboles

- Vamos a crear una clase nodo para probar los algoritmos de recorrido de árboles binarios
- Por comodidad vamos a definir las clases y métodos dentro del archivo Module1.vb
- La clase nodo se define de la siguiente manera:

```
Public Class Nodo
    Public Property Derecha As Nodo
    Public Property Izquierda As Nodo
    Public Property Valor As Integer


    Public Sub New()
    End Sub

    Public Sub New(val As Integer)
        Valor = val
    End Sub

    Public Sub New(val As Integer, ByRef izq As Nodo, ByRef der As Nodo)
        Valor = val
        Izquierda = izq
        Derecha = der
    End Sub

End Class
```

Laboratorio árboles

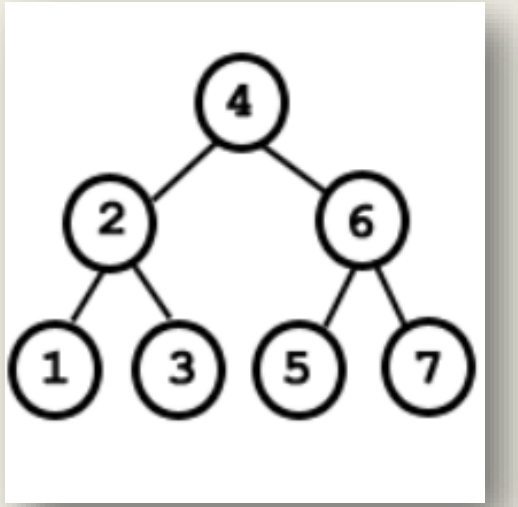
- A continuación vamos a definir los métodos de recorrido en profundidad y amplitud

```
Public Sub PorNiveles(raiz As Nodo)
    Dim cola As New Queue(Of Nodo)
    Dim nodo As Nodo
    cola.Enqueue(raiz)
    While cola.Count > 0
        nodo = cola.Dequeue
        Console.Write(nodo.Valor.ToString & " - ")

        If nodo.Izquierda IsNot Nothing Then
            cola.Enqueue(nodo.Izquierda)
        End If

        If nodo.Derecha IsNot Nothing Then
            cola.Enqueue(nodo.Derecha)
        End If
    End While
End Sub
```

```
Public Sub PreOrden(nodo As Nodo)
    If nodo Is Nothing Then Return

    Console.Write(nodo.Valor & " - ")
    PreOrden(nodo.Izquierda)
    PreOrden(nodo.Derecha)
End Sub

Public Sub InOrden(nodo As Nodo)
    If nodo Is Nothing Then Return

    InOrden(nodo.Izquierda)
    Console.Write(nodo.Valor & " - ")
    InOrden(nodo.Derecha)
End Sub

Public Sub PostOrden(nodo As Nodo)
    If nodo Is Nothing Then Return

    PostOrden(nodo.Izquierda)
    PostOrden(nodo.Derecha)
    Console.Write(nodo.Valor & " - ")
End Sub
```

Laboratorio árboles

- Vamos a replicar la siguiente estructura de árbol
- El método de creación crea las relaciones y devuelve la raíz



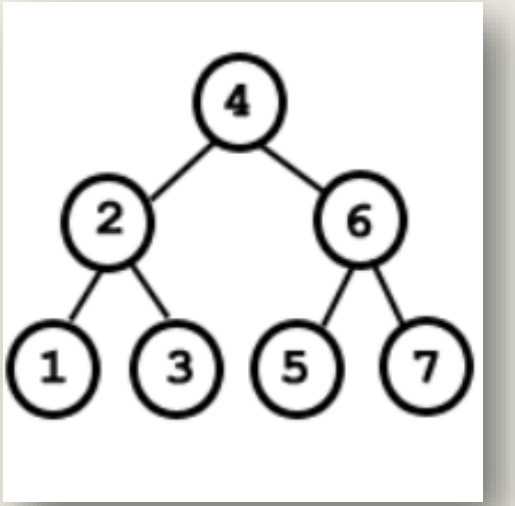

```
Private Function CrearArbol() As Nodo
    Dim nodo1 As New Nodo(1)
    Dim nodo3 As New Nodo(3)
    Dim nodo5 As New Nodo(5)
    Dim nodo7 As New Nodo(7)
    Dim nodo2 As New Nodo(2, nodo1, nodo3)
    Dim nodo6 As New Nodo(6, nodo5, nodo7)
    Dim nodo4 As New Nodo(4, nodo2, nodo6)
    Return nodo4
End Function
```

Laboratorio árboles

- Finalmente escribimos el código de la prueba y verificamos las salidas esperadas

```
Sub Main()
    Dim raiz As Nodo = CrearArbol()

    Console.WriteLine("PreOrden:")
    PreOrden(raiz)
    Console.WriteLine()
    Console.WriteLine("InOrden:")
    InOrden(raiz)
    Console.WriteLine()
    Console.WriteLine("PostOrden:")
    PostOrden(raiz)
    Console.WriteLine()
    Console.WriteLine("PorNiveles:")
    PorNiveles(raiz)
    Console.ReadLine()
End Sub
```

```
file:///C:/Users/Administrator/Documents/
PreOrden:
4 - 2 - 1 - 3 - 6 - 5 - 7 -
InOrden:
1 - 2 - 3 - 4 - 5 - 6 - 7 -
PostOrden:
1 - 3 - 2 - 5 - 7 - 6 - 4 -
PorNiveles:
4 - 2 - 6 - 1 - 3 - 5 - 7 -
```



Laboratorio de grafos

- Este ejemplo va a ser más estructurado, vamos a comenzar creando una clase nodo

```
Public Class Nodo

    Public Property Valor As Integer

    Public Property Hijos As List(Of Nodo)

    Public Sub New(ByVal valor As Integer)
        Me.Valor = valor
        Me.Hijos = New List(Of Nodo)
    End Sub

    Public Sub New(valor As Integer, hijos As IList(Of Nodo))
        Me.Valor = valor
        Me.Hijos = hijos.ToList
    End Sub

End Class
```

Laboratorio de grafos

- Seguidamente vamos a crear la clase Grafo en la cuál agregaremos los métodos de recorridos

```
Public Class Grafo
    Public Shared Sub RecorridoPorAnchura(raiz As Nodo,
                                          ByRef visitados As List(Of Integer))
        Dim cola As New Queue(Of Nodo)
        Dim nodo As Nodo
        cola.Enqueue(raiz)
        visitados.Add(raiz.Valor)
        While cola.Count > 0
            nodo = cola.Dequeue
            Console.Write(nodo.Valor.ToString & " - ")
            For Each hijo In nodo.Hijos
                If Not visitados.Contains(hijo.Valor) Then
                    visitados.Add(hijo.Valor)
                    cola.Enqueue(hijo)
                End If
            Next
        End While
    End Sub

    Public Shared Sub RecorridoEnProfundidad(nodo As Nodo,
                                             ByRef visitados As List(Of Integer))
        For Each hijo In nodo.Hijos
            If Not visitados.Contains(hijo.Valor) Then
                visitados.Add(nodo.Valor)
                RecorridoEnProfundidad(hijo, visitados)
            End If
        Next
        Console.Write(nodo.Valor & " - ")
    End Sub
End Class
```

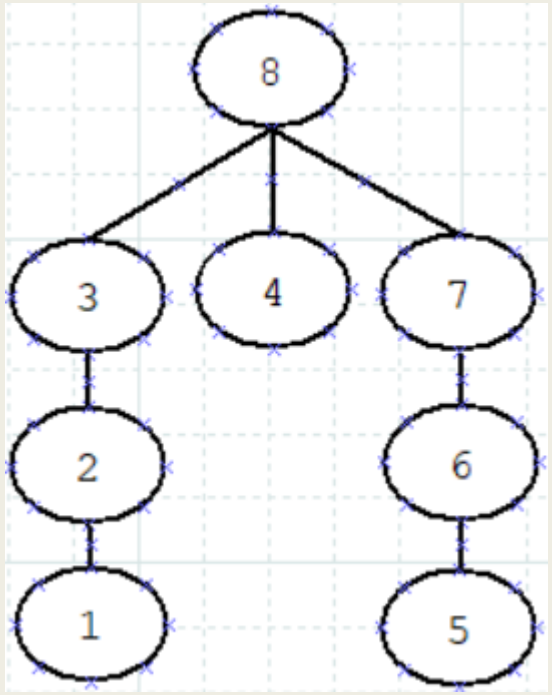
Laboratorio de grafos

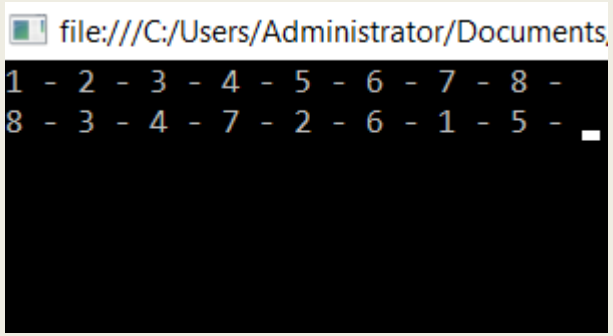
- Ahora en el módulo principal vamos a crear la estructura del grafo y a invocar los métodos de recorrido

```
Module Module1

    Sub Main()
        Dim raiz As Nodo = CrearGrafo()
        Grafo.RecorridoEnProfundidad(raiz, New List(Of Integer))
        Console.WriteLine()
        Grafo.RecorridoPorAnchura(raiz, New List(Of Integer))
        Console.ReadLine()
    End Sub

    Private Function CrearGrafo() As Nodo
        Dim nodo1 As New Nodo(1)
        Dim nodo2 As New Nodo(2, {nodo1})
        Dim nodo3 As New Nodo(3, {nodo2})
        Dim nodo4 As New Nodo(4)
        Dim nodo5 As New Nodo(5)
        Dim nodo6 As New Nodo(6, {nodo5})
        Dim nodo7 As New Nodo(7, {nodo6})
        Dim nodo8 As New Nodo(8, {nodo3, nodo4, nodo7})
        Return nodo8
    End Function

End Module
```

Laboratorio de grafos

- Tomando como base el siguiente ejemplo, las salidas deberían ser las siguientes:





Referencias

- https://es.wikipedia.org/wiki/%C3%81rbol_binario#Recorridos_en_amplitud_(o_por_niveles)
- http://www.bibliadelprogramador.com/2014/04/algoritmos-de-busqueda-en-anchura-bfs-y.html
- http://www.hci.uniovi.es/Products/DSTool/grafos/grafos-queSon.html