**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**ESTRUCTURAS DE DATOS**

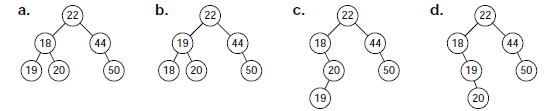
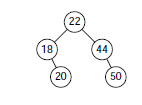
**EVALUACIÓN FINAL - I TÉRMINO 2018**

***Nombre: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Matrícula: \_\_\_\_\_\_\_\_\_\_\_***

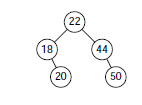
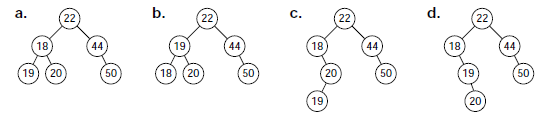
**TEMA 1. (4 PUNTOS)**

**Elija la opción correcta y justifique su respuesta.**

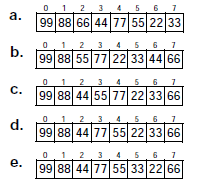
1. A partir del siguiente Árbol Binario de búsqueda, Cuál sería el resultado si se inserta el número 19:

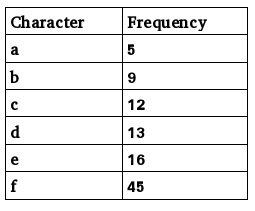


1. A partir del siguiente Árbol AVL, Cuál sería el resultado si se inserta el número 19:



1. Cuál de los arreglos mostrados NO representa un Heap





1. Cuál es la altura del árbol de huffman que se construye a partir de la siguiente tabla de frecuencias.
2. 4
3. 5
4. 6
5. 7

**TEMA 2. (4 PUNTOS)**

Considere las siguientes estructuras:

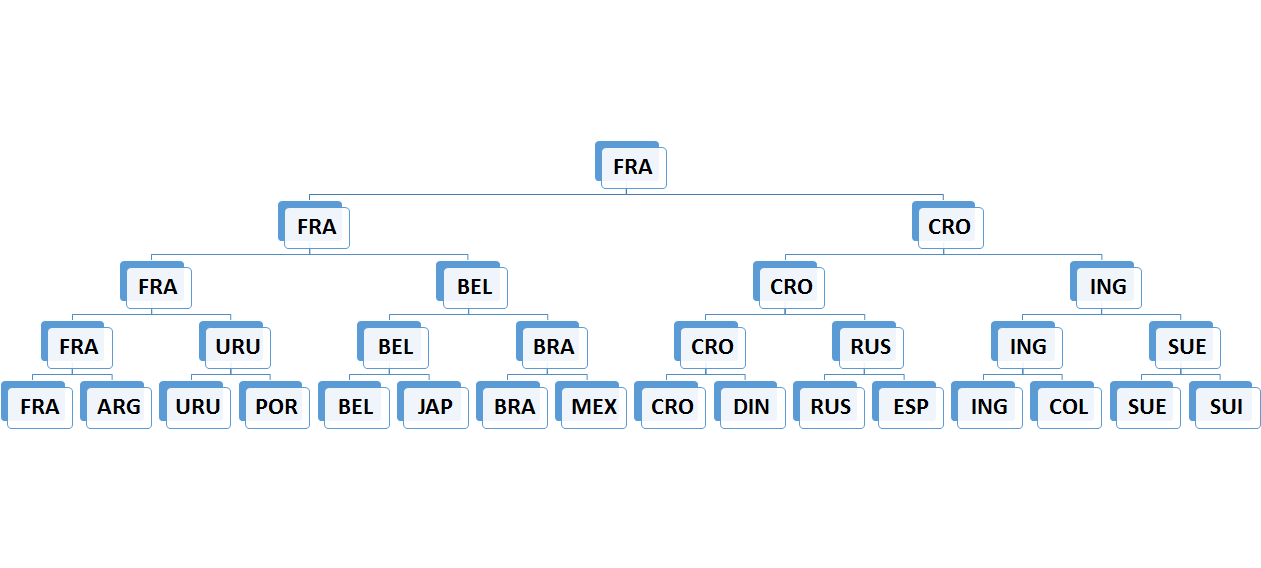
|  |  |
| --- | --- |
| Árbol Multicaminos | Grafo Dirigido sin ciclos |
| Árbol Binario | Grafo Dirigido con ciclos |
| Árbol Binario de Búsqueda / AVL | Grafo No Dirigido con ciclos |
| Heap | Grafo No Dirigido sin ciclos |

Seleccione y justifique la estructura de datos más apropiada para cada uno de los siguientes problemas:

1. Se desea modelar una red de sensores y conocer qué sensores son críticos para mantener toda la red conectada.
2. En un programa de diseño de vehículos se desea almacenar los componentes de un auto agrupados por jerarquías. Por ejemplo: auto contiene carroceria, carrocería contiene capot y puerta, puerta contiene ventana.
3. Se desea modelar la probabilidad de lluvia en base a mediciones indirectas de condiciones climáticas.
4. Crear un programa que muestre los estudiantes ordenados de mayor a menor según su calificación final.

**TEMA 3. (12 PUNTOS)**

Los enfrentamientos desde octavos de final hasta la final de la copa mundial de fútbol 2018 pueden ser modelados con un árbol binario. El equipo ganador de cada enfrentamiento se encuentra en el lado izquierdo del árbol, mientras que el equipo perdedor se encuentra del lado derecho del árbol.



Además, usted cuenta con un mapa de fases que indica el nivel del árbol según una fase de la competencia:

fases = {“campeón” : 0 , “final” : 1 , “semifinal” : 2 , “cuartos” : 3 , “octavos” : 4}

A usted se le solicita implementar los siguientes métodos en la clase árbol binario:

1. **equiposDerrotadosFases** que recibe una fase, el mapa de fases, y retorna todos las selecciones que han perdido partidos en fases anteriores.

List<E> equiposDerrotadosFases (String fase, HashMap<String, Integer> fases)

Ejemplo:

equiposDerrotadosFases(“semifinal”, fases)

retorna -> [“URU”,”ARG”,”POR”,”BRA”,”JAP”,”MEX”,”RUS”,”DIN”,”ESP”,”SUE”,”COL”,”SUI”]

1. **equiposEliminados** que recibe un nombre de selección y retorna una lista con las selecciones que ha derrotado la selección que recibe la función.

List<E> equiposEliminados(String seleccion)

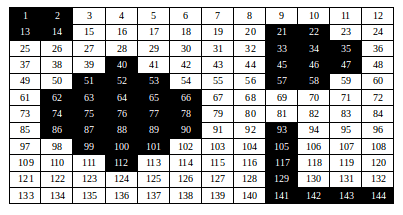
Ejemplo:

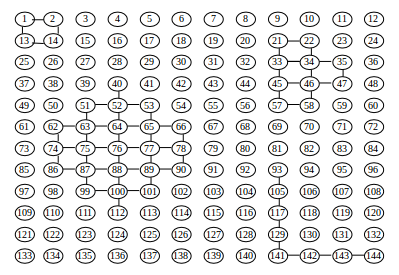
equiposEliminados(“CRO”)

retorna -> [“ING”,”RUS”,”DIN”]

**TEMA 4. (10 PUNTOS)**

Una imagen binaria (solo blanco y negro) puede representarse por medio de un grafo, donde cada vértice representa un píxel y cada arco implica que dos píxeles vecinos tienen color negro (se ignoran pixeles negros aislados). Por ejemplo, la imagen binaria mostrada a continuación se representa por la siguiente matriz:





Implemente una función que recibe un grafo que representa una imagen binaria y que retorna la cantidad de objetos que contiene la imagen.

int objetos(Grafo<int> grafo)

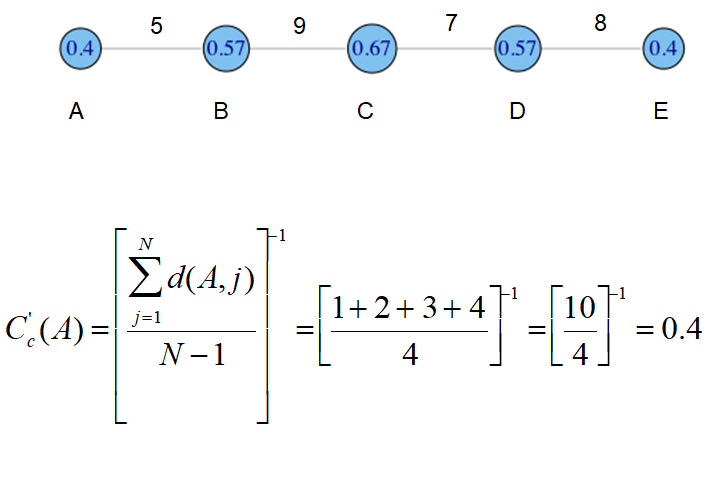
Ejemplo:

objetos(grafo)

retorna ->4

**TEMA 5. (15 PUNTOS)**

La centralidad es una medida de los vértices que permite determinar la importancia de cada vértice dentro del grafo. Una de las métricas para calcular la centralidad es el closeness centrality. El cálculo del closeness centrality consiste en calcular el promedio de la longitud de los caminos más cortos desde un vértice al resto de vértices del grafo.



N es el número de vértices del grafo.

1. Implementar el algoritmo de dijkstra

private void dijkstra(E inicio)

1. Implementar el método centralidad que retorna un mapa cuyas claves son la data de los vértices y el valor es la centralidad.

public HashMap<E,Double> centralidad()