UNIDAD TEMÁTICA 1: Técnicas de Diseño de Algoritmos

TRABAJO DE APLICACIÓN 1

Para cada uno de los siguientes algoritmos

1. Verificar si se puede clasificar dentro de alguna de las técnicas de diseño de algoritmos presentadas.
2. En cualquier caso, fundamentar por qué cumple o no cumple con cada uno de los elementos de la técnica.

1- Obtener la cantidad de hojas de un árbol binario

En TNodoArbolBinario cantidadHojas(): entero

COM

```
Si hijoIzq == nulo Y hijoDer == nulo entonces
        devolver 1
fin si

hojasIzquierdo = 0
hojasDerecho = 0

Si hijoIzq != nulo entonces
        hojasIzquierdo = hijoIzq.cantidadHojas()
fin si

Si hijoDer != nulo entonces
        hojasDerecho = hijoDer.cantidadHojas()
fin si

devolver hojasIzquierdo + hojasDerecho
```

FIN

#2 – Obtener el nivel de una clave en un árbol binario de búsqueda

En TNodoArbolBinarioBusqueda nivelDeLaClave (unaClave): de tipo entero

```
COM
de tipo entero resultado
Si unaClave == etiqueta entonces
  devolver 0
Sino
  Si unaClave < etiqueta entonces
      Si hijoIzq != nulo entonces
            resultado = hijoIzq. nivelDeLaClave(unaClave)
      sino
              devolver -1
      fin si
   Sino
      Si hijoDer != nulo entonces
             resultado = hijoDer. nivelDeLaClave(unaClave)
      sino
              devolver -1
      fin si
   fin si
Fin si
Si resultado < 0 entonces resultado = resultado -1
Sino resultado = resultado + 1 fin si
devolver resultado
FIN
```

3 – Obtener las matrices para el armado de un árbol binario de búsqueda óptimo

```
public void encontrarOptimo(int cantElem, int[] FrecExito, int[] FrecNoExito) {

    int i, j, k, kraiz, h;
    int min, PesoSubArboles;

    for (i = 0; i < cantElem + 1; i++) {
      W[i][i] = FrecNoExito[i];
      P[i][i] = W[i][i];
    }

    for (i = 0; i < cantElem; i++) {
      for (j = i + 1; j < cantElem + 1; j++) {
        W[i][j] = W[i][j - 1] + FrecExito[j] + FrecNoExito[j];
      }
    }

    for (i = 0; i < cantElem; i++) {
      j = i + 1;
      P[i][j] = W[i][j] + P[i][i] + P[j][j];
      R[i][j] = j;
    }

    kraiz = 0;
    for (h = 2; h < cantElem + 1; h++) {

      for (i = 0; i < cantElem - h + 1; i++) {
        j = i + h;
        min = Integer.MAX_VALUE;

        for (k = i + 1; k < j + 1; k++) {
          PesoSubArboles = P[i][k - 1] + P[k][j];
          if (PesoSubArboles <= min) {
            min = PesoSubArboles;
            kraiz = k;
          }
        }
}
```

4 – Obtener la agenda óptima

```
De tipo Conjunto de tareas agendaOptima (conjunto de tareas entrada)
COM
        Tarea t;
        Conjunto de tareas salida = crear conjunto
        Mientras no(entrada.vacio()) hacer
                Buscar en entrada la tarea t que termine más temprano
                Eliminar t de entrada y toda otra tarea que se superponga con t
                Insertar t en salida
        Fin mientras
        Devolver salida
FIN
```