ALGORITMOS Y ESTRUCTURAS DE DATOS I

FINAL Y SEGUNDO PARCIAL

Teoría - cuestiones

INGENIERÍA INFORMÁTICA 25 de JUNIO de 2001 ***UNIVERSITAT DE VALÈNCIA***

Tiempo: 1 hora **Se recomienda leer atentamente los enunciados antes de contestar.** **No se permiten ni libros ni apuntes**

A

La puntuación para preguntas con múltiples opciones es:

- **Pregunta correcta: 1 punto**
- **Pregunta incorrecta: -0,25 puntos**
- **Pregunta en blanco: 0 puntos**

Las preguntas en las que se marquen varias opciones serán consideradas incorrectas.
El examen se responderá en la misma hoja en el espacio reservado para ello.

Cualquier respuesta fuera de estas hojas será ignorada.

1.- Representa el árbol binario de búsqueda que contiene valores enteros tras las siguientes operaciones:
a.- Insertar (30) / Insertar (25) / Insertar (18) / Insertar (22) / Insertar (14) / Insertar (28) / Insertar (19)
b.- Eliminar (30)
c.- Insertar (30) / Insertar(25)
d.- Eliminar (25)
(Utilizar la parte de atrás del folio.)

2.- ¿Es posible representar un árbol *k*-ario con un máximo de 'n' nodos mediante matrices dispersas?
a.- Sí.
b.- No.
c.- Sólo si transformamos el árbol general en árbol binario siguiendo las reglas vistas en clase.
d.- Sólo si el número de enlaces en cada nivel es mayor que el número de nodos.

3.- Supongamos un grafo con 100 nodos y 30 arcos, en el que en cada nodo se guarda un valor entero (2 *bytes*) y en cada arco un valor real (4 *bytes*.) ¿Cuánta memoria ocupará una representación mediante...

...matrices de adyacencia de un máximo de 150 nodos?	...listas de adyacencia de un máximo de 150 nodos?

4.- ¿Es posible utilizar una cola para mantener una cola de prioridad?
a.- No, porque las operaciones básicas sobre colas no lo permiten.
b.- Sí, pero sería conveniente modificar el método de *Encolar*.
c.- Sí, pero sería conveniente modificar todos los métodos sobre colas.
d.- Sólo sería posible si tenemos una representación de la cola estática (como los *heaps*.)

5.- ¿Qué hace la función *f* sobre *q* si consideramos que *q* es una cola.

f (IniciarCola) -> IniciarCola

$$f(Encolar(q, x)) = \begin{cases} Encolar(q, x) \text{ si } q \text{ es Vacía} \\ Encolar(PrimeroCola(q), f(Desencolar(Encolar(q, x)))) \end{cases}$$

6.- Entre los diferentes métodos de ordenación de vectores, en el peor de los casos:
a.- El mejor siempre es el *Quick-Sort*, porque utiliza la técnica de divide y vencerás y su coste es logarítmico.
b.- El mejor es el *Heap-Sort*, que en el peor de los casos sigue teniendo un coste 'n*lg (n)'.
c.- Depende del número de elementos. Si 'n' es pequeño, el mejor algoritmo es el de inserción.
d.- El *Heap-Sort* no se puede emplear para ordenar vectores. Sólo puede ordenar montículos.

7.- Dada una especificación formal de un tipo abstracto de datos:
a.- Según lo que especifique el T.A.D., habrá que hacer una implementación estática o dinámica.
b.- El T.A.D. no especifica nada acerca de la implementación.
c.- El T.A.D. especifica exactamente las operaciones y la representación interna de los datos.
d.- El T.A.D. sólo especifica la representación. Las estructuras de datos especifican las operaciones y los axiomas.

8.- Las pilas es posible representarlas mediante listas dinámicas doblemente enlazadas circulares. Respecto de esta representación:
a.- Es mejor que la lista simple porque evita casos especiales.
b.- No aporta nada, pero consume menos memoria que la representación con listas simples.
c.- No aporta nada, y además consume mayor cantidad de memoria.
d.- No se puede representar una pila dinámica mediante listas doblemente enlazadas.

9.- Supongamos un *heap* de máximos. ¿Es posible su representación mediante estructuras dinámicas?
a.- Sí, pero las operaciones son más complejas.
b.- No, porque es imposible implementar el método *EliminarMaximo*.
c.- Sí, y el método *EliminarMaximo* es más sencilla.
d.- No, porque es imposible implementar el método *Subir*.

7.- Sea el siguiente programa en C++:

```
int f(int & x);

int main(void)
{
    int x = 1; int y = 2;

    y = y + f(x);
    y = y + x;
    cout << y;
    return 0;
}

int f(int & x)
{
    x = x + 2;
    return(x - 2);
}
```

Escribe qué muestra por pantalla.

8.- Dado el siguiente vector:

```
typedef float Datos[5][3];
Datos dato;
```

Y sabiendo que la variable dato comienza en la posición de memoria 100 y que el tamaño de un float es 6 bytes, calcular en qué posición de memoria está el elemento dato[3][1].

9.- Dado el siguiente programa en C++:

```
int main(void)
{
    int pos; int i; int valor;
    ifstream fich;

    cout << "Dime una posiciÛn:";
    cin >> pos;

    fich.open("datos.txt");
    if(!fich)
        cerr << "Error abriendo fichero" << endl;
    for(i = 0; i < pos; i++)
        fich >> valor;
    fich >> valor;
    cout << "El valor en la posiciÛn " << pos << " es:" << valor << endl;
    fich.close();
    return 0;
}
```

El acceso que se está realizando sobre el fichero "datos.txt", ¿es secuencial o directo?. Razona tu respuesta.

P.1. Dada la siguiente clase "Heap" que representa un montículo de máximos representada de forma dinámica:

```
Class Heap
{
   public:
      Heap (void);
      Heap (const Heap &);
      ~Heap (void);
      bool Insertar (Valor);
      bool EliminarMaximo (void);
      bool ConsultarMaximo (Valor &);

   private:
      typedef Heap * Puntero;

      Valor info;
      Puntero izdo, dcho;

      bool esvacio;

      void Subir (Puntero);
      void Bajar (Puntero);
};
```

Implementa los métodos 'Subir' y 'Bajar' y aquellos que consideres oportunos. El parámetro de tipo 'Puntero' que se le pasa a los métodos, es un puntero que apunta al nodo que contiene el valor que debemos comprobar si ha de subir o bajar.

P.2. Implementa un nuevo método 'Retroceder' sobre una lista con punto de interés representada:

a.- Mediante una lista simplemente ligada.

b.- Mediante un vector de un máximo de 100 elementos.

Escribe en cada caso la parte privada de la clase.

P.3. Sea el siguiente programa en C++

```
int f (int a)
{
   int b, c;

   if (a <= 1)
      c = a;
   else
   {
      b = a % 2;
      a = a / 2;
      c = f (a);
      c = c * 10 + b;
   }
   return c;
}

int main (void)
{
   int x;

   x = f (5);
   cout << x << endl;

   return 0;
}
```

Realiza una traza del programa y di cuál será el valor final de x