Examen de Teoría de la Programación

ESCUELA DE INGENIERÍA INFORMÁTICA – UNIVERSIDAD DE OVIEDO

Final Febrero – Curso 2010-2011 17 de enero de 2011

DNI _____ Nombre _____ Apellidos _____

Titulación: ☐ Gestión ☐ Sistemas

3. (1 punto) Sea la función:

- ***Tipo vector= array [1..1000] de entero;***
- ***fun maximoValor(a:vector;n:entero) dev (x:entero)***

que realiza extrae el máximo valor de los n primeros elementos del vector a.

a) **Realizar la especificación formal de la función con la técnica pre/post.**
b) **Codificar en Java el esqueleto de esta función utilizando un método público: cabecera, precondición y postcondición (no hace falta codificar la funcionalidad del método en sí).**

4. (1,75 puntos) El problema de las "Parejas estables" consiste en que tenemos n hombres y n mujeres, y una matriz nxn que contiene la compatibilidad entre ellos. El problema consistiría en encontrar la manera de emparejarlos a todos, de tal forma que cada hombre acabe con una mujer diferente y la suma de las compatibilidades sea máxima.

a) **Completar el código Java para dar solución a este problema con la técnica de Backtracking (escribirlo en los recuadros preparados a tal efecto).**

```
public class Backtracking {
      int solucion[];
      int compatibilidadActual;
      int solucionOptima[];
      int compatibilidadOptima;
      int matrizValores[][];  // Compatibilidad para cada una de las parejas
      boolean  selecionado[];

    […]

private void ensaya(                              ) {
      for(                                                ){
            if(                                           ){
                  solucion[pos]=est;
                  selecionado[est]=true;
                  compatibilidadActual+=matrizValores[pos][est];
                  if(                                    ){
                        if(compatibilidadActual>compatibilidadOptima){
                              
                              
                              
                  }
                  }
                  else ensaya(                        );
                  
                  
            }
      }
}
```

5. (1,5 puntos) El problema de la mochila consiste en que disponemos de *n* objetos y una "mochila" para transportarlos. Cada objeto *i*= 1,2, ... *n* tiene un peso w_i y un valor v_i. La mochila puede llevar un peso que no sobrepase *W*. En el caso de que un objeto no se pueda meter entero, se fraccionará, quedando la mochila totalmente llena. El objetivo del problema es

maximizar valor de los objetos respetando la limitación de peso. Queremos resolver este problema mediante un método voraz.

a) **(0,5 puntos) Escribir pseudocódigo para la función heurística que permita alcanzar la solución óptima.**
b) **(0,5 puntos) Escribir el código Java del método que permita obtener la solución a este problema mediante un algoritmo voraz (escribir exclusivamente el método que realiza el algoritmo voraz suponer ya cargados los arrays necesarios con los datos de los objetos, no realizar ninguna entrada / salida en el método).**
c) **(0,5 puntos) Qué ocurre si no se pudieran fraccionar los objetos. ¿Seguiría siendo óptimo el heurístico propuesto? Demostrar la respuesta.**

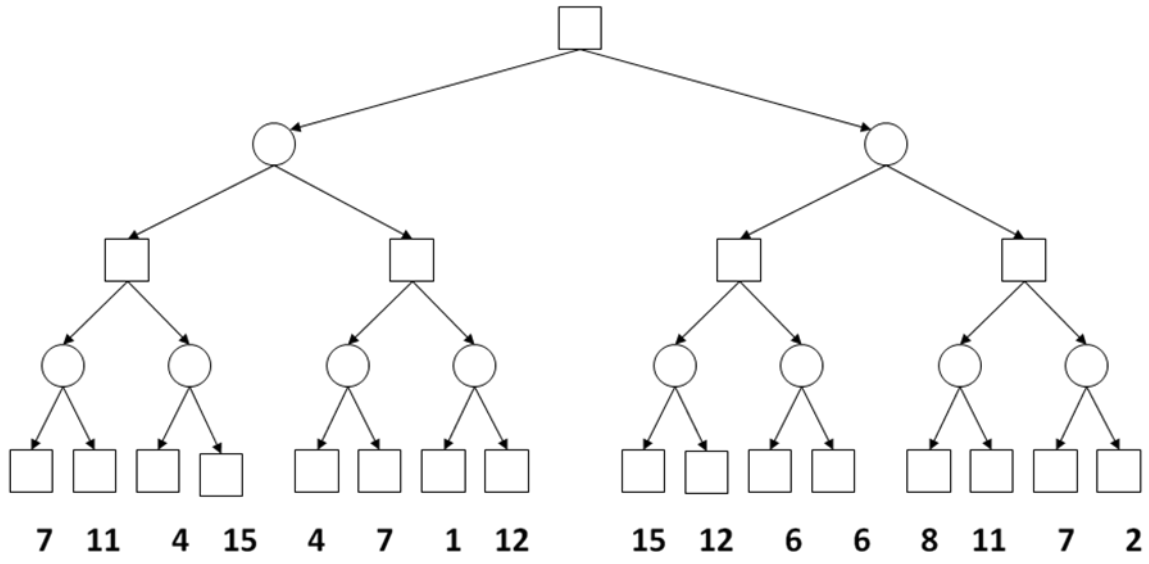
6. (1,5 puntos) Sea el problema de la asignación de tareas a agentes visto en el ejercicio 5. Se dispone de una matriz de costes de ejecución de una tarea j por un agente i.
Dados 4 agentes: a..d y 4 tareas: 1..4. Y la siguiente matriz de costes:

	1	2	3	4
A	34	24	24	26
B	36	39	32	25
C	20	24	32	17
D	22	27	29	19

Nos planteamos resolver el problema mediante la técnica de ramificación y poda.

a) **(0,25 puntos) Explicar cómo se calcula el heurístico de ramificación para el estado donde asignamos la tarea 3 al agente b, cuando ya tenemos asignada la tarea 1 al agente a.**
b) **(0,25 puntos) Explicar cómo se calcula la cota inicial de poda y razonar cuándo se produce el cambio de esta cota.**
c) **(0,5 punto) Representar el árbol de estados después de haber expandido los primeros 3 estados (incluyendo el estado inicial)**
d) **(0,5 puntos) Representar de forma ordenada los estados que quedan en la cola de prioridad en la situación descrita en el punto c).**

7. (1,25 puntos) Desarrollar la poda α-β para conocer que jugada debe realizar el jugador MAX, sobre el siguiente árbol:



a) **Sombrear los nodos que haya que desarrollar**
b) **Escribir las cotas α y β,**
c) **Marcar los cortes e indicar si son de tipo α o β,**
d) **Por último, indicar que jugada debe elegir MAX para situarse en la mejor posición posible.**

Notas: El jugador que realiza el primer movimiento en el árbol es MAX. Los nodos del árbol se desarrollan de izquierda a derecha.