Examen final – 11 de mayo de 2017

Apellidos, nombre _____ NIF: _____

Pregunta 1 (1 p.)

Responde a las siguientes preguntas (Nota: Resuelve las operaciones intermedias):

a) (0,5 p.) Si la complejidad de un algoritmo es $O(3^n)$, y dicho algoritmo toma 4 segundos para n=2, calcula el tiempo que tardará para n=4

b) (0,5 p.) Considere un algoritmo con complejidad O(logn). Si para t = 2 segundos el método pudiera resolver un problema con un tamaño de n = 8, ¿cuál podría ser el tamaño del problema si dispusiéramos de un tiempo de 4 segundos?

Pregunta 2 (1 p.)

Teniendo en cuenta los siguientes métodos, indica sus complejidades y explica cómo las has calculado:

a) (0,5 p.)

```
public void algohara(int n) {
	if (n > 0) {
		algohara(n-3);
		for (int i = 0; i < n; i++)
			for (int j = 0; j < n; j++)
				System.out.println("i=" + i + "j=" + j); //O(1)
	}
}
```

b) (0,5 p.)

```
public void algoharaparecido(int n) {
	if (n > 0) {
		algohara(n-2);
		for (int i = 0; i < n; i++)
			for (int j = 0; j < n; j++)
				System.out.println("i=" + i + "j=" + j); //O(1)
	}
}
```

Pregunta 3 (2 p.)

Por favor, responde a las siguientes preguntas sobre el algoritmo de ordenación Quicksort:

a) (1 p.) Dada la siguiente secuencia de números: **6, 5, 4, 3, 9, 7, 1, 8, 2** ordénalos utilizando Quicksort con la estrategia de la mediana a tres para seleccionar el pivote en cada iteración. Indica claramente la traza del algoritmo.

b) (0,5 p.) Indica los casos mejor, peor y medio de las complejidades para el algoritmo de Quicksort. Explica claramente cuando se dan los casos mejor y peor.

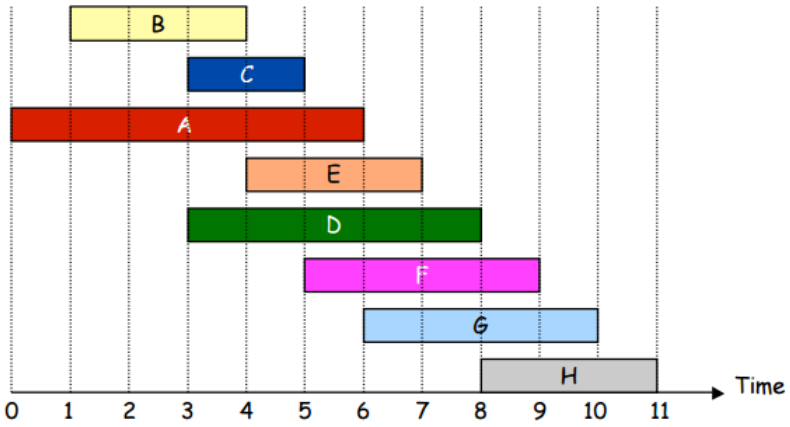
c) (0,5 p.) ¿Cuáles son las posibles complejidades (caso mejor, peor y medio) que se pueden obtener con los algoritmos de burbuja, selección e inserción?

Pregunta 4 (2 p.)

Supongamos lo siguiente:

- Tienes que realizar 8 tareas hoy
- Sólo puedes hacer una actividad al mismo tiempo
- Quieres hacer tantas actividades como te sea posible hoy

Así, necesitas crear una planificación que maximice el número de actividades. A continuación, puede verse un ejemplo de actividades y sus tiempos de realización:



Cada una de las actividades toma un tiempo t_i y dos actividades son compatibles si no se superponen. La idea es establecer un orden de acuerdo a un cierto criterio y elegir las actividades que no se superpongan con actividades que ya hayan sido elegidas.

a) (0,75 p.) Describe un heurístico que no ofrezca una solución óptima y pruébalo con un contraejemplo:

b) (0,75 p.) Describe un heurístico que ofrezca una solución óptima (siempre, no solo para el ejemplo) y utilízalo con el ejemplo dado. ¿Cuántas actividades podrías hacer ese día?

c) (0,5 p.) ¿Cuál sería la mejor complejidad temporal si quisiéramos implementar en Java el algoritmo que nos ofrece una solución óptima? Explícalo brevemente.

Apellidos, nombre _____ NIF: _____

Pregunta 5 (2 p.)

Dado un vector de enteros positivos *v* y un valor *sumaDada*, realizar un programa que devuelva un subconjunto del vector que sumen ese valor dado. Si no existe un subconjunto que devuelva la suma exacta, devolver el que más se aproxime, por arriba o por abajo. Para resolverlo se ha utilizado la técnica de backtracking. Ten cuenta que en el código se ha optimizado para que no haga más llamadas recursivas si se encuentra la solución exacta.
Completa el código rellenando los huecos (en esta hoja). Se dispone de un método *int abs(int num)* que devuelve el valor absoluto del entero pasado como parámetro.

```
public class SubconjuntosSumaDada
{
	static int n;		// número de elementos en el conjunto de partida
	static int[]v;			// vector de números positivos diferentes
	static int sumaDada;		// valor de la suma

	static boolean[]marca;		// marca True los elementos usados
	static int sumaParcial;		// suma parcial acumulada hasta un estado

	private static int mejorSuma;
	private static boolean[] mejorMarca;
	private static boolean solExacta;

	static void subconjuntoCercaSumaDada(_____)
	{
		if (_____) {
			if (_____)
			{
				_____
				_____
				if (abs(sumaParcial - sumaDada) == 0)
					solExacta = true;
			}
		} else
			for (_____) {
				if (!solExacta)
				{
					if (j == 1) {
						_____
						_____
					}
					subconjuntoCercaSumaDada(_____);
					if (j == 1) {
						_____
						_____
					}
				}
			}
	}
...
```

Pregunta 6 (2 p.)

El *cuadriatlón* es una variante de triatlón en la que se combinan Natación(N)-Piragua(P)-Ciclismo(C)-Carrera a pie(A). Ahora que están en auge los deportes de resistencia *César Acero* y tres amigos han decidido participar en una prueba que se realiza por relevos, es decir, cada participante realizará un deporte y le dará el relevo a su compañero. César que es muy competitivo, dispone sus tiempos suyos y de sus compañeros para cada una de los deportes. Quiere diseñar el algoritmo que mediante la técnica de ramificación y poda permita decidir a César qué deportista realizará cada prueba para obtener el mejor tiempo posible.
Deportista 1 (César) (18-39-24-15). Deportista 2 (23-28-25-18). Deportista 3 (17-29-26-17). Deportista 4 (22-30-25-20). Los tiempos corresponden respectivamente a cada uno de los deportes.

a) (0,25 p.) Explicar cuál sería un heurístico de ramificación bueno y cómo se calcula. Y aplicarlo al estado en el que asignamos al deportista 1 a ciclismo y al deportista 2 a natación (y quedan dos deportistas por asignar).
b) (0,25 p.) Explicar cómo se calcula la cota inicial de poda y razonar cuándo se produce el cambio de esta cota.
c) (1 p.) Representar el árbol de estados después de haber expandido dos estados del árbol.
d) (0,5 p.) Representar de forma ordenada los estados que quedan en la cola de prioridad en la situación descrita en el punto c).