Examen de Programación 3

4 de febrero de 2017

- Este parcial dura **3** horas y consta de 2 carillas. El total de puntos es **100**.
- **NO** se puede utilizar ningún tipo de material de consulta. Salvo que se indique lo contrario podrá usarse todo lo visto en el teórico, práctico y laboratorio sin demostrarlo, indicando claramente lo que se está usando.

Se requiere:

I) Numerar todas las hojas e incluir en cada una el **nombre, cédula de identidad, número de página y cantidad de hojas entregadas**.

II) Utilizar las hojas de **un solo lado** y escribir con lápiz.

III) Iniciar cada ejercicio en hoja nueva.

Parte obligatoria

Esta parte es eliminatoria. Para la aprobación del examen debe obtenerse un mínimo del **50 % de esta parte (20 puntos)**. En caso de no llegar a dicho mínimo, **NO** se corregirán los problemas.

Pregunta 1 (15 puntos)

Sean $f, g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ funciones de los naturales en los reales no negativos. Demuestre o muestre un contraejemplo de cada una de las siguientes proposiciones.

(a) $f + g \in O(\text{máx}(f, g))$.

(b) $f + g \in \Omega(\text{máx}(f, g))$.

(c) $\text{máx}(f, g) \in \Theta(f + g)$.

Pregunta 2 (12 puntos)

Sea $G = (V, E)$ un grafo conexo no dirigido con una función de costo definida sobre E. Sean U un subconjunto de los vértices de V y A un subconjunto de las aristas de E tal que toda arista de A tiene uno de sus extremos en U y el otro en $V - U$.

Se propone el siguiente enunciado como la propiedad `MST` vista en el curso:

> Sea $a \in A$ una arista tal que ninguna arista de A tiene costo menor estricto que a. Entonces si T es un árbol de cubrimiento de costo mínimo de G, a es una arista de T.

Muestre con un contraejemplo que el enunciado no es correcto y escríbalo correctamente.

Pregunta 3 (13 puntos)

(a) Sea `FibDC` el algoritmo que calcula los números de Fibonacci mediante la técnica *Dividir y Conquistar*. Llamamos $S(n)$ a la cantidad de sumas que se realizan al invocar `FibDC(n)`. Plantee la recurrencia que calcula $S(n)$.

(b) Escriba el algoritmo `FibPD` que calcula los números de Fibonacci mediante *Programación Dinámica*. El algoritmo debe usar $\Theta(1)$ espacio.

(c) Construya una tabla con la cantidad de sumas que realizan `FibDC` y `FibPD` para los valores de n desde 0 hasta 10. Explique en una oración a qué se debe la diferencia del crecimiento de la versión `FibDC` con respecto al de la versión `FibPD`.

Problemas

Problema A (30 puntos)

Una organización va a comenzar un nuevo proyecto para el cual convoca a un conjunto de sus funcionarios, a los que identifica con números del 1 al n. Los funcionarios serán divididos en equipos, cada uno de los cuales debe tener un coordinador que lo identifica, a quien se le pagará un aumento. Ese aumento, que varía según el funcionario, se mantiene en el arreglo `Costo` de tamaño n. El coordinador debe ser uno de los integrantes del equipo. Se pretende invertir el mínimo posible en los aumentos para los coordinadores. Cada funcionario debe ser asignado a un equipo, y sólo a uno. Algunos de los funcionarios están vinculados a otros en proyectos anteriores. Se exige que en ningún grupo queden funcionarios que ya estaban vinculados a otros funcionarios del mismo grupo. El conjuntos de vínculos de cada funcionario se mantiene en el arreglo `Vinculos`. Se asume que si j pertenece a `Vinculos`[i] entonces i pertenece a `Vinculos`[j]. Ningún funcionario pertenece a su propio conjunto de vínculos.

Exprese, con lenguaje natural y formal, en términos de `Backtracking` el problema de organizar los grupos.

(I) Defina la forma de la tupla.

(II) Formalice las siguientes expresiones, indicando a que categoría corresponden: restricción explícita, restricción implícita, función objetivo, predicado de poda.

a. Si un funcionario es coordinador de un equipo, debe ser el coordinador del equipo que integra.

b. No se asigna como coordinador del equipo de un funcionario i alguien que es más costoso que el propio i.

c. El coordinador del equipo de cada funcionario no pertenece a la lista de vínculos del funcionario.

(III) Complete la formalización.

Problema B (30 puntos)

Se pretende ordenar de manera decreciente un arreglo de enteros no negativos con índices entre 0 y $n-1$. El arreglo está formado por 2^h segmentos cada uno de los cuales tiene longitud k y está ordenado de manera decreciente.

En lo que sigue se considera que la operación básica es la comparación entre elementos del arreglo.

(I) Implemente el algoritmo `ordenar`. El tiempo de ejecución en el peor caso debe ser $\Theta(n \cdot h)$. Fundamente por qué se cumple la cota de tiempo requerida.

```
void ordenar(int A[], int n, int k, int h);
```

Nota: Puede usar, asumiendo implementada, la funcion `merge` de la parte II, que ordena de manera decreciente un segmento formado por dos segmentos consecutivos que están ordenados de manera decreciente.

(II) Implemente la funcion `merge`. Muestre que el peor caso del tiempo de ejecución está en $\Theta(\text{largo})$.

```
/*
   Ordena de manera decreciente el segmento de A desde pos hasta pos + largo - 1.
   Precondiciones:
      - Los segmentos desde pos hasta pos + largo/2 - 1 y
      desde pos + largo/2 hasta pos + largo - 1
      estan ordenados de manera decreciente.
      - 0 <= pos <= n - largo - 2.
      - largo es multiplo de 2.
 */
void merge(int A[], int pos, int largo);
```

El siguiente es un ejemplo con $n = 12$, $k = 3$ y $h = 2$ del formato de A:

8 5 0 7 3 3 9 3 2 12 8 1

El resultado de una llamada a `merge(A, 6, 6)` debe ser:

8 5 0 7 3 3 12 9 8 3 2 1