Primer Parcial de Programación 3

29 de setiembre de 2016

- Este parcial dura **3** horas y consta de 3 carillas. El total de puntos es **40**.
- En los enunciados llamamos `C`* a la extensión de `C` al que se agrega el operador de pasaje por referencia `&`, comentarios en línea, declaración de tipos y enumerados como en `C++`, los operadore `new` y `delete` y el tipo de datos `bool`.
- **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.
- Justifique sus respuestas.

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.

Ejercicio 1 (13 puntos)

(a) Indique la veracidad de las siguientes afirmaciones justificando todos los pasos. Sea una función $f : N \to R^*$ una función arbitraria. Se cumple entonces que:

I. $f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$

II. $f(n) \in \Omega(n) \Rightarrow 2^{f(n)} \in \Omega(2^n)$

(b) Considere un hospital que cuenta con S salas de internación, las cuales se identifican por un número $[0..S-1]$, y n pacientes internados. Considere las siguientes operaciones y sus respectivos órdenes:

- `int obtenerCantidadCamasLibres(Estructura E, int id)` Devuelve la cantidad de camas libres de una sala identificada por su número de sala, en $O(1)$ peor caso.
- `void internarPaciente(Estructura E, char * documento)` Se ingresa un nuevo paciente en el hospital, identificado por documento, en $O(log\ n + S)$ caso promedio
- `void altaMedicaPaciente(Estructura E, char * documento)` Se realiza el egreso de un paciente que se encontraba internado, dado su documento, en $O(log\ n)$ caso promedio.
- `int* obtenerListaDeSalasDisponibles(Estructura E)` Retorna una lista de salas con camas libres (no asignadas a pacientes) ordenadas en forma ascendente según su número identificador de sala, en $O(S)$ peor caso.
- `int obtenerIdentificadorSalaPaciente(Estructura E, char * documento)` Dado el documento de identidad de un paciente dentro del hospital, devuelve el número de sala en el que se encuentra internado, en $O(log\ n)$ caso promedio.

I. Dibuje un diagrama detallando una estructura de datos que permita realizar las operaciones anteriores y respete los órdenes indicados de cada una de ellas. Justifique el cumplimiento de dichos órdenes.

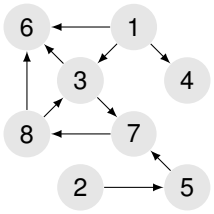
II. Implemente el pseudocódigo de la operación `internarPaciente` definiendo los cambios necesarios en la estructura. No es necesario especificar la implementación de operaciones de TADs cuyas estructuras fueron utilizadas para resolver la parte anterior.

Ejercicio 2 (14 puntos)

Sea $G = (V, A)$ un grafo dirigido simple. En una recorrida `DFS` se denota con A_T el conjunto de aristas que pertencen a los árboles de cubrimiento, llamadas aristas *de árbol* (tree). Las aristas que pertencen a $A - A_T$ se clasifican en: *de regreso* (back), *directas* (forward) y *cruzadas* (cross).

(a) Defina las aristas de regreso, directas y cruzadas.

(b) Muestre los árboles de cubrimiento que resultan de la recorrida `DFS` en el grafo de la figura de la derecha (cuando haya que elegir entre más de un vértice, tanto al recorrer las listas de adyacencia como al iniciar un nuevo árbol, se debe elegir el vértice identificado con el número menor). Incluya las aristas de regreso, directas y cruzadas, etiquetando cada una con su tipo. Escriba a la izquierda de cada vértice el tiempo en que fue descubierto (*prenum*) y a la derecha el tiempo en que se terminó de procesar (*postnum*). El tiempo se incrementa en una unidad cada vez que un vértice es descubierto o terminado de procesar. El primer vértice es descubierto en el tiempo 1.



(c) En esta parte se elimina la restricción establecida en la parte b de elegir el menor cuando haya más de un vértice para elegir (es decir, se permite elegir cualquiera de ellos). ¿Podría una recorrida `DFS` de ese mismo grafo dar como resultado un único árbol de cubrimiento? Es caso afirmativo muestre el ejemplo. En otro caso explique por qué no puede haber menos de dos árboles

(d) Abajo se muestra el seudocódigo incompleto de una recorrida `DFS`. Se tratan como variables globales el grafo $G = (V, A)$, la cantidad *n* de vértices en *V*, el entero *tiempo*, la variable booleana *hay_ciclo* y los arreglos *prenum* y *postnum*, ambos de *n* enteros. Cada arista tiene un atributo llamado *tipo* cuyo valor puede ser: *arbol*, *regreso*, *directa*, *cruzada* o estar indefinido.

Al finalizar la recorrida debe quedar asignado el valor correcto de `(v,w).tipo` para cada $(v, w) \in A$, y el valor de *hay_ciclo* debe ser `true` si y sólo si en G hay algún ciclo.

```
Recorrida
  Para i desde 1 hasta n
    prenum[i] ← ∞
    postnum[i] ← ∞
  Para cada a ∈ A
    a.tipo ← indefinido
  hay_ciclo ← false
  tiempo ← 0
  Para i desde 1 hasta n
    <invocacion>
```

```
DFS(v)
  <preprocesamiento>
  Para cada (v,w) ∈ A
    Si prenum[w] = ∞
      (v,w).tipo ← arbol
      DFS(w)
    en otro caso
      <caso no-arbol>
  <posprocesamiento>
```

Escriba el seudocódigo de `invocacion`, `preprocesamiento`, `posprocesamiento` y `caso no-arbol` que completan los algoritmos **Recorrida** y **DFS(v)**. No se puede utilizar otras variables o funciones.

Ejercicio 3 (13 puntos)

Merge sort es una estrategia de tipo Divide& Conquer para ordenar arrays que consiste en dividir un array en partes de igual tamaño, ordenar cada parte, y luego utilizar las partes ordenadas para ordenar el array original.

Suponga que en lugar de dividir a la mitad cada paso del Merge Sort como se vió en el teórico, el array a ser ordenado se divide en tres, se ordena cada tercio, y finalmente se las combina usando una función de merge de tres partes.

Como ayuda se recuerda el pseudocodigo de la función Merge vista en el teórico. Se ignoran los casos base.

```
C = output [largo = n]
A = 1er array ordenado [largo n/2]
B = 2do array ordenado [largo n/2]

Merge(Arreglo A, Arreglo B, Arreglo C)
    i = 1
    j = 1
    for k = 1 to n
        if A(i) < B(j)
            C(k) = A(i)
            i++
        else [B(j) < A(i)]
            C(k) = B(j)
            j++
        end
    end
```

(a) Escriba el pseudocódigo de la función *3_way_merge* que sustituye a la Merge presentada (el array de entrada no tiene repetidos).

(b) ¿Cuál es el orden asintótico del tiempo de ejecución de este algoritmo? Justifique brevemente utilizando los conceptos volcados en clase.

- $O(n)$
- $O(nlog(n))$
- $O(n(log(n))^2)$
- $O(n^2log(n))$