Primer Parcial de Programación 3 (1/10/2009)

Instituto de Computación, Facultad de Ingeniería

- Este parcial dura 4 horas y contiene 5 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 &, y las sentencias new, delete y el uso de cout y cin.
- NO se puede utilizar ningún tipo de material de consulta. Salvo que se indique lo contrario puede 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 y la cédula de identidad.

ii. Utilizar las hojas de un sólo lado y escribir con lápiz.

iii. Iniciar cada ejercicio en hoja nueva.

iv. Poner en la carátula la cantidad de hojas entregadas, y un índice indicando en qué hojas respondieron cada problema.

Ejercicio 1 (14 puntos)

1. **(9 puntos)** Considere la siguiente función en C*, la que recibe un arreglo A con *n + 1* valores enteros en el rango [0 : k]; siendo k una constante.

```
1  int F (int *A, int n) {
2        int c=0, r=0, i, j;
3        for (i=0; i<n+1; i++) {
4              r = 1;
5              if (A[i] > A[0]) {
6                    for (j=1; j<n+1; j++)
7                          if (i!=j)
8                                if (A[i]==A[j])
9                                      r = 0;
10             }
11             c = c+r;
12       }
13       return c;
14 }
```

a) ¿Qué calcula la función F? No describir el algoritmo sino especificar conceptualmente qué resultado retorna.

Para el resto del ejercicio se considera que la operación básica de F es la **asignación** de algún valor a la variable ***r***.

b) **Mejor caso y Peor caso**: calcule la cantidad de operaciones básicas que se efectúan para el mejor y el peor caso. Indique también cuales casos constituyen el mejor y peor caso. Explique lo más detalladamente posible en relación al algoritmo.

c) **Caso promedio**: suponga que para todos los elementos A[i] de la secuencia A, si $i \neq 0$ e $i \neq j$ se cumple que:

- la probabilidad de que un elemento sea menor o igual que A[0] es la misma que dicho elemento sea mayor, es decir que

  P(A[i] ≤ A[0]) = P(A[i] > A[0]) = 1/2

- Son independientes y tienen distribución uniforme esto quiere decir que:

  P(A[i] = A[j]) = 1 / (k+1)

Calcule la cantidad de operaciones básicas que se efectúan en el caso promedio. Explique detalladamente y justifique en términos del algoritmo.

d) **Ordenes**: indique el orden (expresión simple más exacta) de los tres casos calculados en las partes b) y c). Para esta parte no se requiere justificar. Sólo se tendrá en cuenta esta parte si calculó lo pedido en las partes b) y c).

2. (5 puntos) Sean $T_1(n) \in O(T_2(n))$ y $T_2(n) \in O(f(n))$. ¿Cuáles de las siguientes afirmaciones son ciertas? Justifique detalladamente sus respuestas demostrando o dando un contraejemplo partiendo de las **definiciones dadas en el teórico**. **Si utiliza alguna otra propiedad debe demostrarla.**

a) $T_1(n) + T_2(n) \in O(T_1(n))$

b) $|T_1(n) - T_2(n)| \in O(f(n))$

c) $T_1(n) * T_2(n) \in O(f(n))$

Ejercicio 2 (14 puntos)

Una biblioteca maneja información acerca de los libros que la misma tiene, de los socios que pertenecen a la misma, así como información relacionada a los préstamos de los libros.

De cada libro se conoce su título, autor, tema y código. Un libro se identifica por su código y la biblioteca tiene **L** libros. A modo de simplificación, la biblioteca cuenta con un único ejemplar de cada libro, es decir, no van a existir dos libros en la biblioteca con igual título y autor.

De cada lector se conoce su código de socio de la biblioteca, la fecha de inscripción y el conjunto de libros que tiene en préstamo en un momento dado. Un lector se identifica por su código de socio y la biblioteca tiene **S** socios. Como máximo, en un momento dado un lector puede tener en préstamo **P** libros.

A su vez, para cada libro se registra la cantidad de veces que un mismo lector pidió en préstamo el libro.

Llamaremos *Biblioteca* a la estructura de datos utilizada para representar la realidad anterior. Se desean realizar las siguientes operaciones:

1. *void imprimirLibrosPrestados(Biblioteca B)*
   Dada *B*, se imprime para cada libro que se encuentra a préstamo su título y autor. La información de cada libro debe ser impresa ordenada cronológicamente según se prestó el libro. En caso de que no exista ningún libro a préstamo al momento de invocar la operación se imprime el siguiente mensaje: “No hay libros prestados en este momento.” Esta operación debe realizarse en O(L) promedio.

2. *void imprimirLibrosPrestadosDeUnLector(String codigoSocio, Biblioteca B)*
   Dada *B*, imprime la información de los libros que el socio de código *codigoSocio* tiene en préstamo. De cada libro se imprime su código y tema, ordenados en forma ascendente por el código del libro. Esta operación debe realizarse en O(P) promedio.

   Pre-condición: el socio de código *códigoSocio* pertenece a *B*.

3. *void devolverLibro(String codigoLibro, String codigoSocio, Biblioteca &B)*
   Dada *B*, el código del libro *codigoLibro* y el código del socio *codigoSocio*, se devuelve el libro a la biblioteca. Esta operación debe realizarse en O(logP) promedio.

   Pre-condiciones: el libro de código *codigoLibro* y el socio de código *codigoSocio* pertenecen a *B* y el libro de código *codigoLibro* lo tiene prestado el socio de código *codigoSocio* al momento de invocar la operación.

4. *int consultarHistorialDeUnLibro(String codigoLibro, String codigoSocio, Biblioteca B)*
   Dada *B*, el código del libro *codigoLibro* y el código del socio *codigoSocio*, devuelve la cantidad de veces que el socio pidió en préstamo el libro a la biblioteca. En caso de que

el socio nunca haya pedido en préstamo el libro se devuelve 0. Esta operación debe realizarse en O(1) promedio.

Pre-condiciones: el libro de código *codigoLibro*, y el socio de código *codigoSocio* pertenecen a *B*.

1. (**8 puntos**) Diseñe una estructura (haga un dibujo con los comentarios pertinentes) que permita soportar las operaciones mencionadas con los órdenes de ejecución indicados. Explique detalladamente su estructura y justifique los órdenes de ejecución para cada operación.
2. (**2 puntos**) Indique el orden en el caso promedio para la siguiente operación según la estructura que definió en la parte anterior del ejercicio:

   void prestarLibroAUnSocio(String codigoLibro, String codigoSocio, Biblioteca &B)
   Dada *B*, el código del libro *codigoLibro* y el código del socio *codigoSocio*, se presta el libro de código *codigoLibro* al socio de código *codigoSocio*.

   Pre-condiciones: el libro de código *codigoLibro* y el socio de código *codigoSocio* pertenecen a *B*. Al momento de invocar esta operación asuma que el socio no tiene P libros a préstamo.

   Debe indicar paso a paso como se resuelve la operación (pseudocódigo, NO implementar).
3. (**4 puntos**) Suponga ahora que se quiere registrar para cada autor la cantidad de veces que se han prestados libros de su autoría. Suponga que se tienen **A** autores.

   void imprimirInformacionAutor(String nombreAutor, Biblioteca B)
   Dada *B* y el nombre de un autor, imprime la cantidad de veces que se han prestado libros de su autoría. En caso de que nunca se hayan prestado libros de su autoría se devuelve 0.

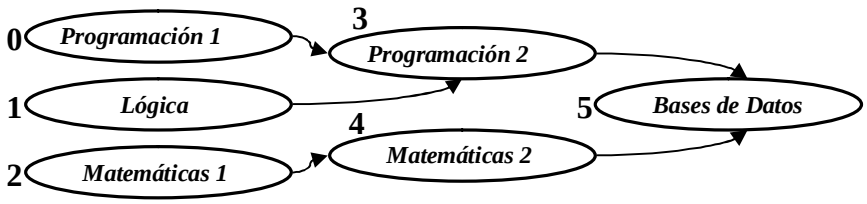
   Pre-condiciones: existe un autor de nombre *nombreAutor*.

   Discuta los cambios a la solución propuesta en las partes anteriores para resolver eficientemente esta nueva operación, **sin afectar los órdenes de las operaciones anteriores** (1 y 2), y especifique el orden de esta nueva operación en el **peor caso**. **En caso de que tener que actualizar alguna de las operaciones anteriores justifique porqué se sigue cumpliendo el orden pedido para la misma.**

Ejercicio 3 (12 puntos)

Sea $G = (N, A)$ un grafo dirigido acíclico (grafo dirigido que no tiene ciclos), un **ordenamiento topológico** de G es un ordenamiento lineal de todos los nodos de G, tal que si G contiene la arista (u, v), entonces u aparece antes que v en el orden dado.

Un ejemplo clásico es la representación de previas de las asignaturas de una carrera.



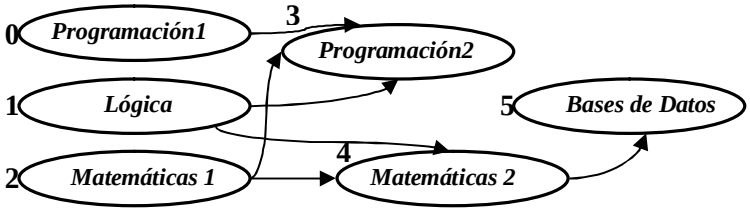
Se puede ver que en cuanto a las asignaturas de una carrera, un ordenamiento topológico daría un plan de estudios. En el ejemplo anterior un ordenamiento topológico es: *Matemáticas 1, Lógica, Programación 1, Matemáticas 2, Programación 2, Bases de Datos.*

Observar que en una recorrida DFS, es fácil detectar primero los nodos que sólo tienen aristas incidentes (del que no "salen" arcos).

1. **(10 puntos)** Escriba un algoritmo basado en **DFS** para encontrar un ordenamiento topológico en un grafo dirigido acíclico. Asuma que:
   - la cantidad de nodos del grafo es fija y definida por la constante **N**: **#define N …**
   - los identificadores de los nodos son enteros en el rango 0..N-1
   - se dispone del tipo Grafo (**como matriz de adyacencia**): **typedef int Grafo[N][N]**
   - si g es del tipo Grafo entonces g[u][v] es 1 sii (u, v) $\in$ A y 0 en otro caso.

   No se pueden usar TADs auxiliares y en caso de utilizar alguna operación auxiliar se debe implementar.

   Implementar la función **`int* ordenTopologico (Grafo g)`** que retorna un arreglo con los identificadores de los nodos del grafo ordenados topológicamente.
2. (**1 punto**) Muestre el arreglo con el ordenamiento topológico que resulta de ejecutar la función de la parte 1 sobre el siguiente grafo:



3. (**1 punto**) ¿Existe un único ordenamiento topológico de un grafo dirigido acíclico? Justifique su respuesta.