Primer Parcial de Programación 3 (4/10/2013)

Instituto de Computación, Facultad de Ingeniería

- Este parcial dura **4** horas y contiene 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 &, las sentencias *new*, *delete*, el uso de *cout* y *cin* y el tipo `bool` predefinido en `C++`.
- 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)

a) Dado un grafo **G = (V, E)** dirigido con ***n*** vértices y ***n-1*** aristas. ¿Cuántas componentes fuertemente conexas puede tener como máximo y cuántas como mínimo? Justifique detalladamente su respuesta.

b) Dadas las siguientes primitivas de un grafo dirigido:

```
Transpuesto: Grafo → Grafo
Inducido: Grafo × Conjunto de vértices → Grafo
DFS: Grafo × Vértice → Grafo
ObtenerVertices: Grafo → Conjunto de vértices
```

Sea un grafo **G = (V, E)** dirigido y un vértice **u∈V**. Escriba un algoritmo en base a las operaciones anteriores que encuentre la componente fuertemente conexa a la que pertenece **u**.

c) Considere un grafo **G = (V, E)** no dirigido. Sea **V*** ⊆ **V** tal que el subgrafo **G*** = **(V*, E*)** inducido por **V*** es completo.
Demuestre o de un contraejemplo de las siguientes afirmaciones.

1. Todos los vértices de **G*** aparecen necesariamente en el mismo árbol del bosque generado por la recorrida en profundidad (*DFS*) de **G**.

2. Todos los vértices de **G*** aparecen necesariamente consecutivos en un camino de un árbol del bosque generado por la recorrida en profundidad (*DFS*) de **G**

Ejercicio 2 (13 puntos)

a) Dado el siguiente algoritmo:

```
void algoritmo(int n, int* a) {
   int sum = 0;
   for (int i=1; i<n; i=i+2) {
        if (a[i-1] > a[i])
             sum++;
   }
   for (int i=1; i<n; i=i+2)
        swap(a, i-1, i);
   for (int i=1; i<n; i=i+2) {
        if (a[i-1] < a[i])
             sum++;
   }
}

void swap (int* a, int i, int j) {
   int aux = a[i];
   a[i] = a[j];
   a[j] = aux;
 }
```

que recibe como parámetro un arreglo **a** de tamaño **n** de enteros distintos, interesa calcular el costo en el mejor ($T_B(n)$) y peor ($T_W(n)$) caso, considerando simultáneamente las siguientes tres operaciones elementales:

1. sum++ tiene costo *c1*.
2. La comparación entre elementos de a con costo *c2*.
3. La asignación en el arreglo a con costo *c3*.

Se pide:

- Indique en forma genérica las entradas de tamaño n que maximizan el costo y luego calcule el costo del peor caso.
- Indique en forma genérica las entradas de tamaño n que minimizan el costo y luego calcule el costo del mejor caso.

b) Determine si son correctas las siguientes afirmaciones, realizando una demostración detallada de la veracidad o falsedad de las mismas. Para esta parte puede utilizar propiedades vistas en el curso, enunciándolas correctamente antes de aplicarlas.

1. $e^n 2^n \in O(e^{2n})$
2. $(n+n^2)! \in O((n^2)!)$
3. $n! \in O(n^n)$

Ejercicio 3 (13 puntos)

a) Defina el factor de carga de una tabla de dispersión abierta e indique si algún valor de éste evita el peor caso en futuras búsquedas.

b) Considere un conjunto de N personas de las cuales se conoce su cédula de identidad, nombre y fecha de nacimiento (día, mes y año). Considerando las siguientes operaciones y sus respectivos órdenes:

1. *char * obtenerNombre(Estructura E, const char* cedula)* en O(1) promedio. Retorna el nombre de la persona con cédula de identidad *cedula*. Precondición: la persona con cédula de identidad *cedula* existe en *E*.
2. *ListaPersonas obtenerListaDePersonasPorMesDeNacimiento(Estructura E, int mes)* en Θ(1) peor caso. Retorna la lista de personas cuyo mes de fecha de nacimiento es *mes*.
3. *Persona obtenerPersonaMayorEdad(Estructura)* en Θ(1) peor caso. Retorna la persona de mayor edad en *E*. En caso de existir varias con la misma fecha de nacimiento retorna cualquiera.

Dibuje un diagrama detallando una estructura de datos que permita realizar las operaciones anteriores y respete los órdenes indicados de cada una de ellas, justificando porque se cumplen los órdenes de ejecución.

c) Implemente en C* la operación:

```
void altaPersona(Estructura * E, const char * cedula, const char
* nombre, fecha fechaNacimiento)
```

que da de alta en la estructura *E* de la parte b), la persona con cédula de identidad *cedula*, nombre *nombre* y fecha de nacimiento *fechaNacimiento*. Defina todos los tipos datos necesarios.

Puede utilizar:

- los tipos de datos *Persona* y *fecha* con operaciones de creación y selectoras.
- operaciones sobre los tipos de datos vistos en este curso o anteriores definiendo, como se pide, dichos tipos de datos y explicando qué hace cada operación.