Primer Parcial de Programación 3

2 de octubre de 2015

- 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 `&`, comentarios en línea, declaración de tipos y enumerados como en `C++`, las sentencias `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.

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.

IV) Poner en la carátula la cantidad de hojas entregadas y un índice indicando el número de página en la que comenzó a resolver cada problema.

Ejercicio 1 (10 puntos)

1. Considere el siguiente algoritmo que recibe una matriz de enteros $n \times n$:

```
1 void algoritmo (int n, int** A)
2 {
3    for (int c = 0; c <= n-1; c++)
4       for (int f = n - 1; f > c; f--)
5          if (A[f][c] == A[c][f])
6             calcular(n);
7 }
```

   Asumiendo que la ejecución de la función `calcular(n)` tiene un costo $2\log(n)$ y que el resto de las instrucciones tienen costo cero se pide:

   a) Describa en forma general las entradas de tamaño $n \times n$ en las que se da el peor caso, justifique.

   b) Describa en forma general las entradas de tamaño $n \times n$ en las que se da el mejor caso, justifique.

   c) Calcule los tiempos de ejecución en el peor y el mejor caso $T_W(n)$ y $T_B(n)$, indique el orden exacto al que pertenecen.

2. Determine a partir de las definiciones si las siguientes afirmaciones son verdaderas o falsas, justifique detalladamente. En caso de utilizar alguna propiedad deberá enunciarla y demostrarla.

   a) $n^2 \in O(n\log(n))$

   b) $2^n \in O(3^n)$ y $2^n \notin \Omega(3^n)$

Ejercicio 2 (20 puntos)

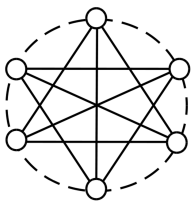
a) Para determinar una ordenación topológica de un grafo dirigido acíclico se dispone del siguiente código:

```
1  // Sea el grafo 'g' representado mediante matriz de adyacencia con
2  // identificadores de vértices enteros y cantidad de vértices 'N'.
3  int *ordentopo(Grafo g) {
4     bool visitado[N]; // Marcas de visitado ('true' es visitado).
5     int orden[N];     // Ordenamiento de vértices
6     int pos = 0;      // Indice del ordenamiento
7     for (int v = 0; v < N; v++) {
8        if (! visitado[v])
9           topo(g, v, visitado, orden, pos);
10    }
11    return orden;
12 }
13
14 void topo(Grafo g, int v, bool *visitado, int *orden, int &pos) {
15    visitado[v] = true;
16    orden[pos] = v;
17    pos++;
18    for (int w = 0; w < N; w++) {
19       if (g[v][w] == 1) {
20          if (! visitado[w]) {
21             topo(g, w, visitado, orden, pos);
22          }
23       }
24    }
25 }
```

1) Indicar si el código es correcto; y en caso de no serlo, describir los errores detectados.
2) En el caso de detectar errores, establecer las correcciones al código para que sea correcto (sugerencia: utilizar los números de líneas como referencia).

b) Una recorrida BFS en un grafo no dirigido permite clasificar las aristas según los vértices, a los que establecen adyacencia, hayan o no sido visitados. La recorrida permite categorizar las aristas a vértices que se visitan como integrantes de un árbol de cubrimiento y permite establecer niveles de distancia desde los vértices al vértice de partida. Además, permite categorizar las aristas a vértices ya visitados (que no pertenecen al árbol) como aristas entre vértices de niveles adyacentes o aristas entre vértices de igual nivel, según el caso.

Se define el grafo estrella a partir de la remoción de un anillo (aristas de un ciclo simple de todos los vértices) de un grafo completo. A modo de ejemplo el grafo estrella de $n = 6$ vértices es de la siguiente forma:



donde las aristas punteadas son las del anillo que se ha removido.

Para cualquier recorrida BFS en el grafo estrella (con $n \geq 5$), determinar en función de n las cantidades de:

1) aristas del grafo,
2) aristas del árbol de cubrimiento y su cantidad de niveles,
3) aristas entre vértices de niveles adyacentes.

Ejercicio 3 (10 puntos)

Cada semestre el Sistema de Bedelías habilita la inscripción a los cursos por parte de los estudiantes de la facultad. Se sabe que hay E estudiantes, cada uno de los cuales se identifica con su cédula y del que además se conoce su nombre. Para inscribirse a cursos un estudiante debe ingresar su cédula y el código del curso. Hay C cursos, identificados con un entero entre 1 y C.

Luego de cerrado el período de inscripción, bedelías realiza el control de estudiantes habilitados para cada curso, debiendo entregar el listado de los mismos a cada responsable de curso. Un estudiante está habilitado para un curso únicamente si tiene aprobadas las previas correspondientes. La cantidad máxima de previas de un curso es P. La cantidad máxima de cursos aprobados por un estudiante es A.

Diseñar estructuras de datos apropiadas para implementar eficientemente las siguientes operaciones, describiendo cómo se obtienen las cotas de tiempo pedidas:

1. Dada la cédula de un estudiante y el código de un curso, registra la aprobación del curso por el estudiante. La operación debe realizarse en $O(A)$ caso promedio.
2. Dada la cédula de un estudiante y el código de un curso, inscribir al estudiante en el curso. La operación debe realizarse en $O(E + \log(N))$ peor caso, siendo N la cantidad de estudiantes en el curso.
3. Dado el código de un curso, retornar la lista de estudiantes inscriptos habilitados a cursar, ordenados alfabéticamente por su nombre. La operación debe realizarse en $O(N \times max\{A, P\})$ peor caso.
4. Dado dos códigos de cursos, agregar al primer curso como previa del segundo. La operación debe realizarse en $O(P)$ peor caso.