Examen de Programación 3 y III

12 de diciembre de 2015

- Este examen dura **4 horas** y contiene 3 carillas. El total de puntos es 100 y se requieren 60 para su aprobación.
- **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.
- **NO** se responderá a preguntas en los últimos 30 minutos de prueba.

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.

Parte Obligatoria (Ejercicios 1 a 4)

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.

Ejercicio 1 (10 puntos)

Suponga que tiene una estructura de árbol binario definida de la siguiente forma:

```
1 struct TreeB {
2    int info;
3    struct TreeB* der, izq;
4 }
```

A su vez considere el siguiente algoritmo:

```
1  TreeB* algoritmo (int k, TreeB* A)
2  {
3     TreeB* retorno;
4     if (A == NULL) {
5        retorno = new TreeB;
6        retorno->info = k;
7        retorno->der = NULL;
8        retorno->izq = NULL;
9     else if (k > A->info) {
10       retorno = algoritmo(k, A->der);
11    } else {
12       retorno = algoritmo(k, A->izq);
13    }
14    return retorno;
15 }
```

Teniendo en cuenta que la operación básica es la comparación entre enteros y que n es la cantidad de nodos, responda:

a. Suponiendo que el parámetro de entrada `A` es un ABB y `k` es mayor estricto que todos los elementos del árbol, determine el costo $T(n)$ del algoritmo en el peor caso en función de n. Especifique cuándo se da este caso.

b. Suponiendo que el parámetro de entrada `A` **NO** es un ABB y `k` es mayor estricto que todos los elementos del árbol, determine el costo $T(n)$ del algoritmo en el peor caso en función de n. Especifique cuándo se da este caso.

c. Considerando también la asignación de enteros como una operación básica, determine si el orden de complejidad de $T(n)$ cambia en alguno de los dos casos. Justifique brevemente.

Ejercicio 2 (10 puntos)

a. Defina la variante del problema de la mochila que en el curso se resuelve utilizando Programación Dinámica.

b. Demuestre el principio de optimalidad para dicho problema.

c. Plantee la recurrencia.

d. Muestre mediante un contraejemplo, lo más sencillo posible, que la estrategia Greedy utilizada para resolver de forma óptima la otra variante del mismo problema no siempre resuelve de manera óptima la variante de la parte a.

Ejercicio 3 (10 puntos)

a. Sea G un grafo sin costos en sus aristas. ¿Qué tipo de recorrida de grafos permite obtener el camino más corto (cantidad de aristas) desde un nodo dado de G a todos los demás? Justifique brevemente y escriba el algoritmo correspondiente a dicha recorrida.

b. Si en cambio G tuviera costos no negativos asociados a sus aristas, ¿el algoritmo anterior resuelve el problema de obtener los caminos de costo mínimo (sumando los costos de las aristas) desde un nodo dado de G a todos los demás? Justifique. En caso negativo indique cuál algoritmo lo resuelve.

Ejercicio 4 (10 puntos)

Dado el algoritmo Selection Sort, cuya implementación es:

```
1  void selectionSort(int n, int * & A) {
2      for (int i = 0; i < n-1; i++) {
3          int iMin = i;
4          for (int j = i+1; j < n; j++) {
5              if (A[j] < A[iMin]) {
6                  iMin = j;
7              }
8          }
9          int tmp = A[i];
10         A[i] = A[iMin];
11         A[iMin] = tmp;
12     }
13 }
```

a. Explique cómo se construye el árbol de decisión de este algoritmo.

b. Si se desea utilizar el algoritmo para ordenar tres números enteros, ¿cuál es el árbol de decisión asociado al mismo?

c. ¿Existe alguna hoja del árbol construido en b. que se corresponda con una ejecución del algoritmo que nunca podría darse? Si existe, ¿cuál es y por qué?

Problemas

Problema A (30 puntos)

Sea $G = (V, E)$ un grafo simple no dirigido con $|V| = n$. Se dice que una de sus aristas es **puente** si al removerla el grafo resultante tiene mayor cantidad de componentes conexas que el original.

a. Sean $v_1, v_2 \in V$ puntos de articulación. Si existe $(v_1, v_2) \in E$ ¿es una arista puente? Demuestre o dé un contraejemplo.

b. Sea $(v_1, v_2) \in E$ una arista puente. Muestre un ejemplo en el que v_1 o v_2 NO son puntos de articulación.

c. ¿Qué cantidad máxima de puentes puede haber en G?

d. ¿Qué relación hay entre puentes y ciclos en G?

e. Escribir un seudocódigo, basado en DFS, que dados G y una arista (v_1, v_2) permita determinar si dicha arista es puente.

Problema B (30 puntos)

Un turista está planificando un viaje en su vehículo. Dispone de un mapa de la zona representado en una tabla D. Cada sitio se identifica con un entero entre 1 y n. En $D[i, j]$ (fila i y columna j de D) se encuentra la duración del viaje en el tramo que va desde el sitio i hasta el sitio j. Si no existe un tramo desde i a j, entonces $D[i, j] = \infty$. Los tramos son flechados.

Debido a diferentes condiciones de los tramos (calidad del pavimento, clima, cuestas, interrupción por el tráfico, etc.) el consumo de combustible en cada tramo no es proporcional a la duración del viaje en el mismo. El consumo se encuentra en la tabla C. Esto es, $C[i, j]$ es el consumo en el viaje por el tramo que va desde i hasta j. La capacidad del tanque de combustible del vehículo es T y se asume que está lleno al comenzar el viaje. En algunos sitios hay surtidores de combustibles en los cuales el viajero puede llenar el tanque. Esto se representa en la tabla S. El valor de $S[i]$ es 'SÍ' o 'NO' dependiendo de si en el sitio i hay o no un surtidor.

El turista quiere determinar el camino que lo lleve desde el sitio A al sitio B en el menor tiempo posible. Para esto, además de las duraciones de los viajes por los tramos, debe tener en cuenta que cada vez que decida recargar el tanque en un surtidor la duración se incrementa R unidades de tiempo.

En el siguiente **ejemplo** se ven las tablas S, D y C, y los valores T, R, A y B.

Sitio	S	Duración (D)					Consumo (C)				
		1	2	3	4	5	1	2	3	4	5
1	NO	0	∞	5	∞	∞	0	∞	30	∞	∞
2	SÍ	∞	0	∞	∞	4	∞	0	∞	∞	45
3	NO	∞	3	0	∞	8	∞	35	0	∞	65
4	SÍ	6	∞	∞	0	∞	15	∞	∞	0	∞
5	SÍ	∞	∞	∞	7	0	∞	∞	∞	55	0

$T = 70$
$R = 2$
$A = 3$
$B = 4$

La solución óptima es partir desde 3, pasar por 5 (recargando el tanque) y llegar a 4, con una duración 17, generada por: 8 en el tramo $(3, 5)$, 7 en el tramo $(5, 4)$ y 2 ($= R$) en la recarga necesaria en el sitio 5. La otra solución es partir desde 3, pasar por 2 (recargando el tanque), pasar por 5 (recargando el tanque) y llegar a 4. En esta la duración en los tramos es 14, menor que en la solución anterior, pero debido a que hacen falta 2 recargas (en los sitios 2 y 5) la duración total es 18. En otro ejemplo, si el destino es el sitio 1 la solución óptima es partir desde 3, pasar por 5 (recargando el tanque), pasar por 4 (SIN recargar el tanque) y llegar a 1, con una duración 23 $= 8 + 2 + 7 + 6$. No hace falta recargar en el sitio 4 ya que al llegar al sitio 1 se puede ver que el último sitio donde se recargó fue en el segundo sitio del camino, y desde ahí los consumos de los tramos son $55 + 15 \leq 70$.

Se pide:

Resuelva el problema usando la técnica **Backtracking**:

a. Exprese en lenguaje natural y en lenguaje formal la forma de la tupla, restricciones explícitas, restricciones implícitas y función objetivo en el caso que correspondan.

b. Dibuje el árbol del espacio de soluciones correspodiente al ejemplo anterior (con $A = 3$, $B = 4$). En cada nodo se debe mostrar el cumplimiento o no de las distintas restricciones, el valor de la función objetivo, y si corresponde o no a una solución factible o solución óptima.