Examen de Programación 3

14 de julio de 2017

- Este parcial dura **3** horas y consta de 2 carillas. El total de puntos es **100**.
- **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.

Parte obligatoria

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.

Pregunta 1 (15 puntos)

Justifique si las siguientes afirmaciones son ciertas o falsas:

(a) $3^n \in O(2^n)$

(b) Sean $f, h : \mathbb{N} \to \mathbb{R}^+$. Entonces $f(n) \in O(h(n)) \Rightarrow 2^{f(n)} \in O(2^{h(n)})$

Pregunta 2 (15 puntos)

(a) Describa el método *Greedy* de resolución de problemas.

(b) Enuncie un problema de optimización y describa un algoritmo *greedy* para resolverlo que no siempre obtenga la solución óptima.

(c) Muestre tres ejemplos: uno en el que el algoritmo encuentra la solución óptima, otro en el que encuentra una solución que no es óptima y otro en el que, aunque existe solución, el algoritmo no encuentra ninguna.

Pregunta 3 (10 puntos)

(a) ¿Qué representan los nodos internos y las hojas de un árbol de decisión?

(b) ¿Qué representa la cantidad de hojas y la altura de un árbol de decisión?

(c) En el problema de sorting mediante comparaciones dos a dos, ¿cuál es la altura mínima posible de un árbol de decisión en función del tamaño n de la entrada?

(d) Para una secuencia de tamaño $n = 5$, el costo de mergesort en el peor caso es 8. ¿Cuál es la cota inferior de sorting mediante comparaciones dos a dos para ese mismo tamaño? Explique si esto es compatible con la proposición de que mergesort es óptimo.

Problemas

Problema A (30 puntos)

Se considera una grilla como la mostrada en la Figura 1, en la que cada línea horizontal o vertical de la grilla representa una calle por la cual se puede transitar. Aparecen destacados dos edificios E_1 y E_2 en los extremos de la grilla.

Se asignan coordenadas a los puntos de la grilla que son intersección de dos calles, de modo que el edificio E_1 se considera ubicado en el origen con coordenadas $(0, 0)$ y el edificio E_2 está situado en las coordenadas

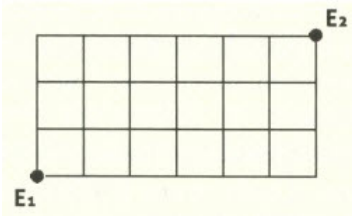

Figura 1: Grilla de calles y edificios destacados

(m, n). Esto corresponde a una grilla con $m+1$ calles verticales y $n+1$ calles horizontales. Para el ejemplo de la Figura 1 se tiene $m = 6$ y $n = 3$.

Se definen los caminos que llevan desde un punto genérico de la grilla de coordenadas (i, j) hasta el edificio E_2, como los recorridos sobre la grilla desde (i, j) hasta (m, n) con la restricción de que al pasar por una intersección de calles sólo se puede continuar hacia la derecha ó hacia arriba, sin salirse de la grilla (lo que hace que dichos caminos sean de largo mínimo). A modo de ejemplo, en la Figura 2 se muestran tres de los caminos posibles entre E_1 y E_2.

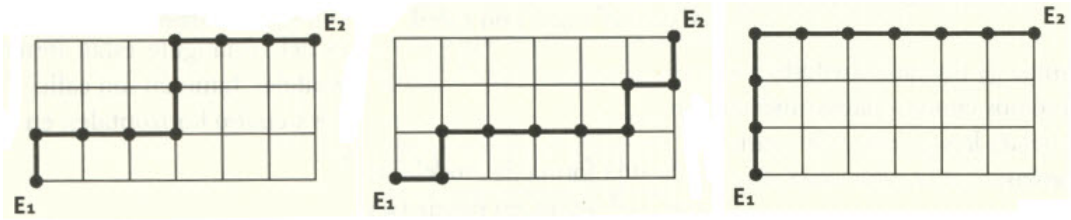

Figura 2: Ejemplos de caminos entre E_1 y E_2

Se define $NC(i, j)$ como el número total de caminos que hay entre el punto de la grilla de coordenadas (i, j) y el edificio E_2. Se toma la convención de que $NC(m, n) = 1$.

(I) Especificar completamente la relación de recurrencia que verifica $NC(i, j)$, para $0 \leq i \leq m, 0 \leq j \leq n$.

(II) Implementar en C^* utilizando la técnica de *Programación Dinámica* la función

```
int NumeroDeCaminos(int m,int n)
```

que calcula y devuelve la cantidad de caminos que llevan desde el edificio E_1 al edificio E_2, para una grilla cuyos valores de m y n se pasan como parámetro.

(III) Para el caso particular $m = 6$ y $n = 3$ correspondiente a la Figura 1, calcular en base al algoritmo implementado la cantidad de caminos entre E_1 y E_2, justificando el resultado obtenido.

Problema B (30 puntos)

Se tiene un conjunto de tareas que se quieren realizar, pero existen relaciones de precedencia entre ellas. Por ejemplo, si se quieren llevar a cabo las tareas A, B y C, en donde C precisa que A esté finalizada para realizarla (*A precede a C*), las maneras posibles de hacerlo son ABC, BAC y ACB.

(I) Modelar el problema en término de grafos.

(II) ¿Es posible resolver el problema para cualquier conjunto de tareas y precedencias? Explicar brevemente por qué, mostrando un ejemplo.

(III) Implementar en seudocódigo una función que dado un grafo de estas características devuelva una estructura lineal que permita recorrer en orden las tareas cumpliendo las restricciones de precedencia. Puede usar y suponer como dadas las estructuras de datos básicas (incluyendo grafos) y las operaciones elementales correspondientes para poder manejarlas. Al implementar, asuma que el problema se puede resolver para el grafo dado.

(IV) ¿Qué cambios habría que hacer a la parte III para que en el caso que no exista solución se retorne el valor nulo?