Examen de Programación 3 y III (07/02/2012)

Instituto de Computación, Facultad de Ingeniería, UdelaR

1. Este examen dura 4 horas y contiene **3** carillas. El total de puntos es 100 y se requieren 60 para su aprobación.
2. 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*, el uso de *cout* y *cin y el tipo bool*.
3. NO se puede utilizar ningún tipo de material de consulta. Puede usarse lo dictado en el curso sin demostrarlo, indicando claramente lo que se está usando.
4. No se contestarán dudas durante la última media hora.

Se requiere:

- Numerar todas las hojas e incluir en cada una el nombre y la cédula de identidad.
- Utilizar las hojas de un solo lado y escribir con lápiz, iniciando cada ejercicio en hoja nueva.
- Poner en la primera hoja la cantidad de hojas entregadas, y un **índice** indicando en qué hoja se respondió cada problema.

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. Usted podrá encontrar planteos prácticos. Los mismos deben ser resueltos justificando detalladamente la correspondencia con la base teórica que utilice en su respuesta.

Ejercicio 1 (10 puntos)

Dadas las siguientes funciones arbitrarias $f : N \to R^*, g : N \to R^*$ y $h : N \to R^*$, partiendo de las definiciones de límite θ ,Ω y O:

a) Pruebe que si el $\lim_{n \to +\infty} \frac{f(n)}{g(n)} = 0$ entonces $f(n) \in O(g(n))$ y $g(n) \notin O(f(n))$

b) Pruebe que si el $\lim_{n \to +\infty} \frac{f(n)}{g(n)} = +\infty$ entonces $f \notin O(g)$ y $g \in O(f)$. Puede suponer para todo *n*, $f(n) \neq 0$ y $g(n) \neq 0$. Para esta parte puede utilizar propiedades de límite además de las definiciones.

Ejercicio 2 (10 puntos)

Una empresa dispone de cierto presupuesto *P* para contratar publicidad en un canal de televisión. El canal cuenta con *N* rangos horarios en los que se puede transmitir el comercial (los rangos no pueden ser fraccionados). Cada rango $0 <= i <= N$ tiene un precio p_i y un valor de r_i correspondiente al *rating* del programa que se emite en ese horario.

Se desea determinar el valor de rating máximo que se puede lograr contratando espacios publicitarios sin sobrepasar el presupuesto disponible.

a) Presente el pseudo código de un algoritmo Greedy para resolver el problema. Indique si su algoritmo siempre obtiene una solución óptima y justifique su respuesta.
b) Basándose en programación dinámica, plantee una recurrencia que determine el valor de rating máximo.

Ejercicio 3 (10 puntos)

Implemente en C* la función *minmax* utilizando la estrategia Divide & Conquer. Dado un arreglo de enteros, la función *minmax* retorna el mínimo y el máximo entero de dicho arreglo.

```
//la función retorna en min y max el menor y el mayor entero del arreglo a entre los índices ini y fin
//PRE: 0 <= ini <= fin
void minmax(int* a, int ini, int fin, int &min, int &max);
```

Ejercicio 4 (10 puntos)

a) Sea T un árbol de decisión, demuestre formalmente o muestre un contraejemplo de las siguientes afirmaciones:

1. La cantidad de nodos externos de T es menor **o** igual a 2^h, donde h es la altura de T.
2. La suma de las profundidades de los nodos externos es menor o igual a *m log m*, donde m es la cantidad de nodos externos de T.

Nota: la profundidad de un nodo es igual a la cantidad de aristas en el camino que lo une con la raíz. La altura de un árbol es igual a la mayor de las profundidades de sus nodos externos.

b) Considere la clase C de los algoritmos que ordenan secuencias mediante comparaciones de pares de elementos de la secuencia. Sea A un algoritmo de C.

Postule una cota inferior lo más grande posible para la cantidad de comparaciones que deben realizarse al ejecutar A en el peor caso. Justifique.

Nota: puede asumir que log (n!) pertenece a Ω (n log n).

Problemas

Problema 1 (30 puntos)

Una ciudad ha decidido flechar todas sus calles. Previo a implementar la decisión se debe verificar que el plan de flechado permita la circulación de vehículos desde cualquier par de puntos de la ciudad.

Para lo que se solicita:

1. Describir como se podría modelar el plan de flechado y el proceso que verifica la circulación entre todo par de puntos de la ciudad.
2. Escribir un algoritmo en C* que a partir de un plan de flechado permita verificar el requisito de circulación según lo descrito en el apartado 1. Asuma que la ciudad tiene calles que conectan n puntos que se identifican con números de 0 a n–1. En el caso de utilizar TAD auxiliares, solo se requiere declarar sus estructuras y operaciones.

Nota: Se penalizará la ineficiencia del algoritmo propuesto, tomando como base la alternativa más eficiente presentada en el curso.

Problema 2 (30 puntos)

Un restaurante que ofrece una variada gama de platos recibe a grupos de N turistas exclusivamente. Su menú cuenta con *p* platos diferentes identificados con enteros en *[0..p)*. Dado que todos los platos deben servirse al mismo tiempo, la cocina solo puede producir hasta *k* platos de un determinado tipo *p*, donde $k<N$.

Para evitar problemas entre los turistas, el guía del grupo le pide a cada turista que liste 3 platos según su preferencia, asegurándoles al menos que serán servidos con alguno de ellos. El guía construye la matriz $preferencias_{Nx3}$, donde *preferencias[i][0]* indica el plato de mayor preferencia para el turista *i*, y ejecuta la función *asignarPlatos* que asigna todos los platos de manera que la satisfacción del grupo sea óptima considerando la preferencia de cada turista.

La valoración de la preferencia de cada plato asignado esta dada por *3/(posición_preferencia+1), con 0<= posición_preferencia < 3, e.g.* si al turista *i* le toca el plato de preferencia 0, entonces se incrementará en 3 la satisfacción grupal.

Se pide:

a) Formalizar el problema en términos de Backtracking.

b) Implementar la función *int* asignarPlatos()* que retorna la asignación óptima utilizando la estrategia BackTracking. Indique las porciones de código que se corresponden con las diferentes partes de la formalización. No se permite el uso de TADs, cualquier función adicional debe ser implementada.

Nota: Asuma que *int k, p, N; int preferencias[N][3]* son variables definidas y de alcance global. Puede retornar el valor *null* en caso que su método sea incapaz de determinar una asignación.