Examen de Programación 3 y III (11/02/2016)

Instituto de Computación, Facultad de Ingeniería, UdelaR

1. Este examen dura 4 horas y contiene **4** 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.
4. No se contestarán dudas durante la última media hora.

Se requiere:

- Numerar todas las hojas e incluir en cada una el nombre, la cédula de identidad, el número de hoja y el TOTAL de hojas.
- Utilizar las hojas de un solo lado y escribir con lápiz, iniciando cada ejercicio en hoja nueva.
- En la **carátula**: completar el índice indicando en qué hoja se comenzó la respuesta a cada problema/ejercicio y el total de hojas entregadas.

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)

Para resolver un problema presentado en su trabajo, un estudiante de Programación 3 presenta cinco algoritmos, *A1, A2, A3, A4* y *A5*. Dado un parámetro *n* de entrada, el algoritmo *j* tiene una complejidad de ejecución $f_j(n)$ según los siguientes valores: $f_1(n) = 2^6\ n\ log(n)$, $f_2(n) = 2^4\ n$, $f_3(n) = 2^n$, $f_4(n) = 2^3\ n^2$ y $f_5(n) = 2^{15}\ log(n)$, donde el logaritmo es en base 2.

1. Ordenar los algoritmos en orden creciente de complejidad asintótica.
2. Ordenar los algoritmos en orden creciente de complejidad cuando $n=2^4$ (n=16).
3. Las ordenaciones en las partes anteriores es diferente. En particular, para n=16 es más eficiente el algoritmo de complejidad exponencial que aquel que tiene complejidad logarítmica. Aparentemente puede parecer una contradicción. Justificar, usando estos ejemplos porqué, según el tamaño de la entrada, puede ser más eficiente un algoritmo exponencial que uno de menor complejidad asintótica para resolver un problema dado.

Ejercicio 2 (10 puntos)

Se desea ordenar el siguiente vector de valores [5, 2, 7, 4, 9, 1, 8, 3, 6] usando ***Quicksort***, donde el pivote es el primer elemento del vector. Mostrar paso a paso, cómo funciona el algoritmo visto en clase para hacer la partición, indicando claramente cómo el pivote queda en su lugar correcto en la ordenación final, y cuáles son los sub-vectores a ordenar recursivamente.

Ejercicio 3 (10 puntos)

1. Describa la forma general de la técnica Greedy en pseudocódigo.
2. Diseñe un Grafo (G) que tenga árboles de cubrimiento mínimo diferentes para el resultado de la ejecución de Prim y otro para Kruskal. Muestre paso la construcción de ambos árboles.

Ejercicio 4 (10 puntos)

Una empresa de software considera un grupo de n programadores para armar un equipo procurando maximizar la experiencia en desarrollo total del equipo formado. Para ello se cuenta con un presupuesto P y un mínimo de experiencia E que debe cumplir el equipo formado.

Asuma que se tienen definidas las siguientes funciones:

- e(k) indica la experiencia del programador k
- p(k) indica el costo del programador k.

Se pide:
Dar una fórmula recursiva f para solucionar el problema. Explicar brevemente que representan los índices que defina, cada paso base y recursivo de la solución.

Problemas

Problema 1 (30 puntos)

En el diseño de una red de comunicaciones, se desea que haya conectividad redundante entre dos centrales críticas de dicha red. La red se modela como un grafo simple no dirigido, y la condición buscada es que entre dos vértices V_1 y V_2 dados que representan a las centrales críticas de la red existan al menos dos caminos simples que no compartan nodos intermedios, a excepción de los nodos extremos V_1 y V_2 (denominados caminos nodo-disjuntos).

1. Dado un cierto diseño de red representado por un grafo *G*, se propone el siguiente procedimiento para verificar si dicho diseño satisface la condición de conectividad buscada:

   i) Hacer una recorrida DFS sobre el grafo *G* a partir del nodo de la central V_1, determinando un primer camino C_1 entre V_1 y V_2 (o la inexistencia de caminos).

   ii) Eliminar todos los nodos intermedios de C_1, obteniendo un grafo G^*.

   iii) Hacer una recorrida DFS sobre el grafo G^* a partir del nodo de la central V_1, determinando un segundo camino C_2 entre V_1 y V_2 (o la inexistencia de este segundo camino).

   Indique porque el procedimiento no es correcto y muestre un contraejemplo.

2. Escriba un algoritmo (seudocódigo) basado en **DFS** que reciba un grafo *G* y dos vértices distinguidos V_1 y V_2 y devuelva un booleano indicando si el diseño de red representado por *G* cumple la condición de conectividad redundante buscada entre V_1 y V_2, y en caso de cumplirse devuelva además dos caminos nodo-disjuntos C_1 y C_2 entre V_1 y V_2 como listas de vértices.

   Considere la siguiente firma para el seudocódigo:

   ```
   bool Hay2Caminos(Grafo G,vertice V1, vertice V2, ListaVertices &C1, ListaVertices &C2)
   ```

   Para escribir el seudocódigo puede asumirse la existencia del TAD ListaVertices, ColaVertices y Grafo *G* con n nodos del tipo vértice, identificados por el rango 1..n. Si el algoritmo devuelve false, es irrelevante el contenido final de C_1 y C_2, las cuales se asumen inicializadas como listas vacías.

Problema 2 (30 puntos)

El área de logística de una empresa importadora se dedica a trasladar cada contenedor que arriba al puerto de Montevideo a su correspondiente destino dentro del país.

La empresa cuenta únicamente con un flete para realizar los traslados. El mismo puede trasladar hasta K contenedores a la vez y puede visitar distintos destinos en el recorrido. Una vez entregados todos los contenedores que trasladaba vuelve al puerto.

Cada entrega tiene una fecha límite que la importadora debe de cumplir. No se acepta la entrega luego de dicha fecha.

El conductor del flete no puede trabajar más de 8 horas seguidas por día (jornada). Si a las 8 horas no finalizó el recorrido que tenía previsto, incluyendo la vuelta al puerto, continúa al otro día. Considerar que el flete sólo puede quedarse en alguno de los destinos a visitar, si se excede la jorndada para llegar al próximo destino, debera continuar al día siguiente. El flete no puede quedarse a mitad de camino entre dos destinos, tampoco puede detenerse en ningún lugar donde no vaya a dejar un contenedor.

Se busca una definir el recorrido del flete minimizando los costos de traslado, cumpliendo con las fechas de entrega y restricciones definidas.

Se pide:

Plantee el problema en términos de **Backtracking,** definiendo formalmente y en lenguaje natural la forma de la tupla, restricciones explícitas, implícitas, función objetivo y predicados de poda que apliquen

Para la solución se cuenta con la siguiente información:

- *Contenedores[]* = arreglo que representa los contenedores a repartir.
- *CostoTraslado[A][B]* = es el costo de ir del punto A al punto B en pesos.
- *TiempoTraslado[A][B]* = es el tiempo que lleva ir del punto A al punto B en minutos. Cada traslado no supera las 8 horas.
- *Contenedor.destino* = indica el destino de entrega del contenedor.
- *Contenedor.fechaLimite* = indica la fecha límite de entrega del contenedor. Las fechas límites no tienen un horario, se pueden realizar las entregas en cualquier horario inclusive del mismo día de la fecha límite.
- *PuertoMontevideo* = es el punto de partida de los contenedores.
- *FechaArribo* = fecha en la que llegan los contenedores al PuertoMontevideo. Se pueden sumar y restar cantidad de días a cada fecha y utilizar los comparadores lógicos. Este mismo día el flete ya puede comenzar y completar sus 8 horas de trabajo.
- *C* = indica la cantidad de contenedores a trasladar.