Examen de Programación 3 y III (12/12/2013)

Instituto de Computación, Facultad de Ingeniería, UdelaR

1. Este examen dura 4 horas y contiene **2** 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:

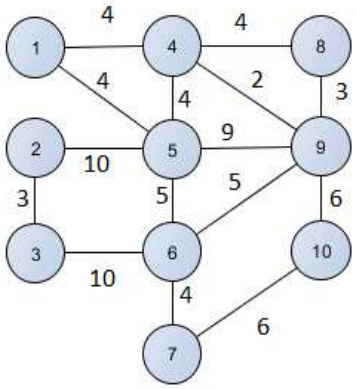
- Numerar todas las hojas e incluir en cada una el nombre y la cédula de identidad.
- Utilizar las hojas de un solo lado y escribir con lápiz, iniciando cada ejercicio en hoja nueva.
- Poner en la primera hoja la cantidad de hojas entregadas, y entregar el índice indicando en qué hoja se respondió cada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

Ejercicio 1 (10 puntos)

a) Enuncie y demuestre la propiedad MST (Minimum Spanning Tree).
b) Dado el siguiente grafo:



pruebe que existe un árbol de cubrimiento de costo mínimo que contiene la arista (*2,3*), utilizando la parte a).
Nota: esta parte no se corregirá si no se hizo la parte a) de forma satisfactoria.

Ejercicio 2 (10 puntos)

Demuestre o de un contraejemplo para el siguiente enunciado: Para todo grafo *G=(V,E)* dirigido, si existe un camino dirigido entre dos vértices *u* y *v* de *G* y si *u* es visitado antes que *v* en una recorrida *DFS*, entonces *v* es descendiente de *u* en alguno de los árboles del bosque de la recorrida *DFS*.

Ejercicio 3 (10 puntos)

Implemente en el lenguaje C* el algoritmo ***Selection Sort***. Calcule el costo en el peor, mejor y caso promedio. En términos de complejidad, ¿el algoritmo es optimo si se hace un análisis basado en comparaciones?

Ejercicio 4 (10 puntos)

Dadas las 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 que para todo *n*, $f(n) \neq 0$ y $g(n) \neq 0$. Para esta parte puede utilizar propiedades de límite además de las definiciones.

Problemas

Problema 1 (30 puntos)

Sea una matriz de enteros de dimensión $n \times n$. Se desea encontrar el máximo y el mínimo de la misma.

Se pide:

a) Construir un algoritmo que utilice la técnica de **Divide & Conquer**.
b) Identifique en su algoritmo donde aplica los distintos ítems de la técnica (división, combinación, paso base y paso recursivo).
c) Determine cual es la operación básica de su algoritmo.
d) Encuentre la cantidad de operaciones básicas que realiza su algoritmo en el peor caso.

Suponga **n** potencia de algún número conveniente.

Problema 2 (30 puntos)

Un estudiante se propone obtener al menos ***M*** créditos en el próximo período. Hay ***n*** actividades distintas con las que puede obtenerlos. Cada actividad tiene una fecha de inicio, una fecha de fin y otorga una cantidad de créditos. El estudiante quiere obtener los créditos haciendo la menor cantidad posible de actividades. Además no quiere que haya superposición en el tiempo entre ninguna de las actividades que elija (si la fecha de fin de una coincide con la de inicio de otra se considera que se superponen). Las actividades se identifican con enteros del 1 al ***n***.

a) Se plantea un algoritmo que consiste en elegir las actividades en orden decreciente de la cantidad de créditos hasta que la cantidad de créditos alcance o supere a ***M***. Al elegir una actividad no se debe superponer con las elegidas anteriormente.

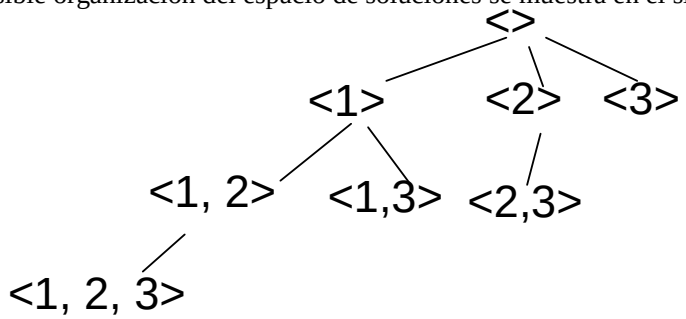
   i. ¿El algoritmo planteado es ávido (Greedy)? Responda **Sí** o **No**.
   ii. Demuestre que, si hay solución, este algoritmo siempre encuentra una solución óptima, o muestre un ejemplo, con la menor cantidad de actividades posibles, de que no siempre la encuentra.

b) Se plantea resolver el problema por **Backtracking**. Los datos están disponibles en el vector *Actividades*, cuyo índice está entre 1 y ***n***. *Actividades[i].Inicio*, *Actividades[i].Fin* y *Actividades[i].Creditos* indican la fecha de inicio, la fecha de fin y los créditos de la actividad *i* respectivamente. Las actividades están ordenadas en el vector de manera creciente según la fecha de inicio (o sea, *Actividades[i].Inicio* $\leq$ *Actividades[i+1].Inicio*).

   i. Formalizar el problema en términos de **Backtracking** Describa formalmente y con lenguaje natural, la forma de la tupla solución (l*a forma de la tupla solución debe ser de longitud variable e indicar las actividades elegidas de manera creciente por la fecha de inicio*), las restricciones explícitas e implícitas, la función objetivo y los predicados de poda.
   ii. Considere las siguientes actividades, con una meta de ***M*** = 10 créditos:

Actividad	Inicio	Fin	Créditos
1	1	4	6
2	2	5	10
3	6	9	6

una posible organización del espacio de soluciones se muestra en el siguiente árbol de soluciones:



¿En qué nodos del árbol de soluciones dado se cumplen las restricciones explícitas? ¿En qué nodos del árbol de soluciones dado se cumplen las restricciones implícitas? ¿Qué nodos del árbol de soluciones dado representan soluciones óptimas?