Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Apellidos, Nombre D.N.I.

Titulación: Grau d'Enginyeria Informàtica **Curso:** Q1 2011–2012 (Final)
Asignatura: Algorísmia **Fecha:** 19 de Enero de 2012
Duración: 2h 30m

Instrucciones generales:

- Cada alumno debe contestar la pregunta **TA** de la competencia transversal de Trabajo Autónomo que le corresponde. En la hoja de respuesta, deberá indicar qué pregunta es la que responde.
- De las restantes preguntas, el alumno podrá escoger dos de las cinco restantes. En su respuesta debe indicar claramente cuáles son las preguntas elegidas. Si se contestan más de dos preguntas entonces la nota será la suma de las dos notas máximas y además, si cada una de las dos calificaciones supera 2.7 puntos sobre 4.5 (60% del peso de la pregunta) entonces las calificaciones de las otras preguntas contestadas se sumarán, en caso contrario serán desechadas.
- En cualquier caso, la calificación global de esta prueba escrita no podrá superar el 10, al sumar todos los puntos.

---

1. **(4.5 puntos)** Dado un vector $A[1..n]$, un k-iceberg es un elemento que aparece al menos $\lceil n/k \rceil$ veces en el vector. Si ordenamos el vector es muy fácil hallar todos los k-icebergs (no puede haber más de $k-1$). El objetivo, sin embargo, es que diseñéis un algoritmo que, usando el principio de *divide y vencerás* al menos en una de sus fases, determina si el vector contiene uno o más k-icebergs y cuáles son, con coste $O(kn)$ lo cual resulta ventajoso si k es una constante.

---

2. **(4.5 puntos)** Escribid un algoritmo eficiente que calcule los k árboles de expansión de peso mínimo para un grafo no dirigido $G = \langle V, E \rangle$ con pesos positivos en las aristas. Si el grafo no contiene k árboles de expansión, el algoritmo devuelve todos los árboles de expansión que tiene el grafo (p.e. un grafo unicíclico con un ciclo de longitud ℓ contiene ℓ árboles de expansión).

---

3. **(4.5 puntos)** Tenemos una barra de metal de longitud L y un conjunto de puntos por los cuales debemos cortarla: $x_1, x_2, \ldots, x_n$. Se cumple que $0 < x_i < L$, $1 \leq i \leq n$ y $x_i < x_{i+1}$, $1 \leq i < n$. Cortar una barra de longitud ℓ por cualquier punto tiene coste ℓ. Por tanto el orden en que se efectúen los cortes es relevante. Por ejemplo, si $x = (2, 4, 7)$ y $L = 10$ podemos empezar cortando por el punto $x_3 = 7$, luego por el punto $x_2 = 4$ el trozo de longitud 7 y finalmente por el punto $x_1 = 2$ el trozo de longitud 4. El coste total es $10 + 7 + 4 = 21$. Si en cambio comenzamos cortando por el punto $x_2 = 4$ y luego, en cualquier orden, por los puntos $x_1 = 2$ y $x_3 = 7$, el coste total será $10 + 4 + 6 = 20$. Escribir un programa de programación dinámica que encuentre el coste óptimo de cortar la barra, dados L y el vector x. Calcular su coste en función de n.

---

4. **(4.5 puntos)** Tenemos tres polinomios $p(x)$, $q(x)$ y $r(x)$ de grados n, n y $2n$, respectivamente, con coeficientes enteros. Los tres polinomios están representados mediante vectores, por ejemplo, $p[i]$ es el coeficiente de x^i en el polinomio $p(x)$. Calcular el producto de los dos polinomios $p(x)$ y $q(x)$ para luego comparar los coeficientes con los de $r(x)$ es demasiado costoso; escribid un algoritmo probabalístico eficiente que nos diga con alta probabilidad si $p(x) \cdot q(x) = r(x)$ o no. No es imprescindible calcular formalmente la probabilidad de que el algoritmo sea correcto, pero debéis razonar informalmente que su probabilidad de acierto es cercana a 1. Calculad el coste del algoritmo: debe ser mucho menor que el del algoritmo determinista que calcula el producto y compara coeficientes.

---

5. **(4.5 puntos)** Describe detalladamente cómo podemos resolver la siguiente variación del problema de flujo máximo sobre una red. Tenemos una red $R = \langle V, E, s, t, c, c' \rangle$ donde V es el conjunto de vértices, $E \subset V \times V$ es el conjunto de arcos, $s \in V$ es la fuente (ningún arco entra en s), $t \in V$ es el sumidero (ningún arco sale de t), $c(e) > 0$ es la capacidad (un natural positivo) del arco e y $c'(v)$ es la *capacidad* (un natural positivo) del vértice v ($v \neq s, v \neq t$). El objetivo es hallar un flujo f de valor máximo sobre esta red con capacidades máximas en los arcos y **los vértices**. Deben cumplirse las restricciones habituales:

   - Para todo arco e, $0 \leq f(e) \leq c(e)$.
   - Para todo vértice $v \in V \setminus \{s, t\}$,

$$f^{\text{in}}(v) = \sum_{e \text{ entra en } v} f(e) = f^{\text{out}}(v) = \sum_{e \text{ sale de } v} f(e)$$

del problema básico, pero adicionalmente ningún vértice puede recibir más flujo que el que indica su capacidad c': para todo vértice $v \in V \setminus \{s, t\}$

$$f^{\text{in}}(v) \leq c'(v).$$

Recordad que el valor del flujo máximo es $f^* = f^{\text{out}}(s) = f^{\text{in}}(t)$. Justificad la corrección del algoritmo. Si el coste de resolver el problema de maximización del flujo estándar es $f(m, n)$, siendo $n = |V|$ y $m = |E|$, ¿cúal es el coste del problema de maximización del flujo con capacidades en arcos y vértices?

Preguntas de Competencia Transversal

...