Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 10 - "Algoritmos Probabilísticos y Aleatorizados"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliar: Dustin Cobas

P1. Polinomios

Dados 3 polinomios en una variable p, q de grado n y r de grado $2n$. Muestre un algoritmo probabilístico que tome tiempo $\mathcal{O}(n)$ en determinar si $pq = r$.

P2. Las Vegas vs Monte Carlo

a) Muestre que si se tiene un algoritmo de tipo Las Vegas de tiempo esperado T y una constante $c > 0$, se puede construir un algoritmo Monte Carlo de tiempo cT y probabilidad de error a lo más $1/c$ para el problema.

b) ¿Siempre se puede transformar un algoritmo Monte Carlo en un algoritmo de tipo Las Vegas? ¿Que restricción adicional sobre el problema podemos colocar para que sea cierto?

P3. 3 Mayoritario

Considere un árbol de altura h perfectamente balanceado, donde cada nodo interno tiene 3 hijos, por lo que hay $n = 3^h$ hojas. Cada hoja tiene un valor booleano, 0 ó 1. El valor booleano de cada nodo interno se calcula como el mayoritario entre sus 3 hijos.

a) Muestre que cualquier algoritmo determinístico necesita en el peor caso examinar las $n = 3^h$ hojas para encontrar el valor de la raíz.

b) Considere un algoritmo aleatorizado que elige dos hijos al azar, los evalúa recursivamente, y si dan el mismo valor evita calcular el tercero, de otro modo calcula el tercero para responder. Demuestre que el número esperado de hijos analizados por cada nodo es $\leq 8/3$.

c) Analice el costo esperado del algoritmo aleatorizado y muestre que es $o(n)$.

Soluciones

P1. Polinomios

Elegimos uniformemente $x \in \{0, \ldots, 4n+1\}$ y luego respondemos `true` si $p(x) \cdot q(x) = r(x)$ y `false` en otro caso.

- En el caso de que la igualdad sea correcta, el algoritmo responde `true` de manera correcta.
- Sin embargo, cuando la igualdad es falsa ($pq \neq r$) el algoritmo podría responder erróneamente `true` en caso que $p(x) \cdot q(x) = r(x)$ o equivalentemente cuando el polinomio $g = pq - r$ se hace cero, i.e. cuando x es cero de g.
  Veamos que g es de grado a lo más $2n+1$ por lo que tiene a lo más $2n+1$ raíces, y luego la probabilidad que el algoritmo se equivoque es:

$$\mathbb{P}(x \; es \; raiz \; de \; g) \leq \frac{2n+1}{4n+2} = \frac{1}{2}$$

P2. Las Vegas vs Monte Carlo

Preliminares :

- Desigualdad de Markov: Si X es una variable aleatoria no negativa con valor esperado positivo entonces
$$\forall c > 0, \; \mathbb{P}(X \geq c\mathbb{E}(X)) \leq \frac{1}{c}$$
- Algoritmo Monte Carlo: Algoritmo probabilístico que termina en un tiempo determinado, pero tiene una probabilidad de error distinta de 0.
- Algoritmo Las Vegas: Algoritmo probabilístico con probabilidad de error 0, pero que no asegura el término de su ejecución, solo asegura un valor esperado finito del tiempo que demora.

a) Si tenemos un algoritmo A "Las Vegas" que resuelve el problema en tiempo esperado $T > 0$, creamos el siguiente algoritmo "Monte Carlo":
   - Correr A por cT tiempo, si ha terminado, respondemos lo mismo que A, si no respondemos "`null`".

   Definamos la variable aleatoria X como el tiempo que demora el algoritmo A. Por Desigualdad de Markov sabemos que $\mathbb{P}(X \geq cT) \leq \frac{1}{c}$, pero $\mathbb{P}(X \geq cT)$ es la probabilidad de que A se demore más de cT y por lo tanto una cota superior a la probabilidad de que nuestro algoritmo "Monte Carlo" se equivoque.

b) La condición para hacer la transformación es contar con un procedimiento eficiente que verifique el `output` dado por el algoritmo "Monte Carlo" sea correcto, pues si lo tenemos podemos hacer un algoritmo "Las Vegas" que corra el "Monte Carlo" hasta que este último entregue una respuesta correcta.

P3. 3 Mayoritario

a) Sea A algoritmo que resuelve el problema. Mantendremos un adversario que a cada pregunta en una hoja responde:

- 1 si es el primer hijo preguntado.
- 0 si es el segundo hijo preguntado.
- Si es el tercer hijo preguntado, reconoce el ancestro más bajo que tiene alguna hoja no preguntada en su subárbol. En caso que los otros dos hijos tengan hojas no preguntadas, el adversario responde 1. En caso que solo uno de sus hijo tenga hojas no preguntadas respondo 0. Finalmente si no existe tal ancestro respondo 1.

Notemos que si no existe el ancestro del punto 3 significa que ya han sido consultadas todas las hojas. Por otro lado, en todo momento, si se han consultado menos de 3^h hojas que el adversario puede completar con 0s o 1s a su gusto el resto de las hojas, obteniendo resultados de evaluación diferentes y por lo tanto haciendo que el algoritmo se equivoque frente a cualquier respuesta que de.

b) No importa cual sea el input, en un nodo siempre se cumple que al menos dos de sus hijos deben evaluar al mismo valor. Por lo tanto, con probabilidad $\geq 1/3$ el algoritmo no elige el hijo diferente, solo teniendo que analizar 2 nodos, y con probabilidad $\leq 2/3$ debe analizar 3. Con esto el número esperado de hijos a analizar será $\leq 1/3 \cdot 2 + 2/3 \cdot 3 = 8/3$.

c) Si definimos el costo aleatorizado para evaluar un árbol de altura h tendremos que:

$$T(h) \leq \frac{1}{3} \cdot 2 \cdot T(h-1) + \frac{2}{3} \cdot 3 \cdot T(h-1) = \frac{8}{3} T(h-1)$$

$$T(h) \leq \left(\frac{8}{3}\right)^h$$

Como un árbol de altura h tendrá $n = 3^h$ hojas, tenemos que $h = \log_3 n$, por lo que

$$T(h) \leq \left(\frac{8}{3}\right)^h = \left(\frac{8}{3}\right)^{\log_3 n} = n^{\log_3 \frac{8}{3}}$$

Lo que significa que $T(h) = \mathcal{O}\left(n^{\log_3 \frac{8}{3}}\right) = o(n)$.