PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2283 Diseño y análisis de algoritmos
2° semestre 2019

Ayudantía 6

Dudas a Antonio López (alopez7@uc.cl)

Problema 1

Dadas las matrices $A, B, C \in \mathbb{Q}^{n \times n}$ se desea determinar si $A \cdot B = C$

1. Proponga un algoritmo determinístico para resolver este problema.

   Solución: Un algoritmo determinística para este problema es simplemente multiplicar A y B y luego comparar el resultado con C. Esto tiene complejidad $O(n^{2,807})$ usando el algoritmo de *Strassen*.

2. Proponga y analice un algoritmo aleatorizado para resolver el problema que tome tiempo $O(n^2)$.

   Solución: El algoritmo es como sigue:

   a) Elegir un vector $r \in \{0,1\}^n$ aleatorio.
   b) Calcular $P = A \cdot (B \cdot r) - C \cdot r$
   c) Si $P = 0^n$ retornar `true`, sino `false`

   Este algoritmo toma tiempo $O(n^2)$ ya que multiplicar una matriz por un vector solo toma $O(n^2)$ operaciones y lo mismo para la resta de matrices. Ahora calcularemos la probabilidad de error del algoritmo.

   Notar que el algoritmo no se puede equivocar si $A \cdot B = C$. Por lo tanto calcularemos la probabilidad de que el algoritmo retorne `true` cuando $A \cdot B \neq C$.

   Sea $D = A \cdot B - C$ y sea $d_{i,j}$ el valor de D de la fila i y columna j. Ya que $A \cdot B \neq C$ existe algún $d_{i,j} \neq 0$. Sea $P = D \cdot r$: si $P = 0^n$, el algoritmo se equivoca. Por lo tanto calcularemos la probabilidad de que $P[j] = 0$ dado que $d_{i,j} \neq 0$.

   $$P[i] = d_{1,j} \cdot r_1 + \ldots d_{i,j} \cdot r_i + \ldots d_{n,j} \cdot r_n = d_{i,j} \cdot r_i + y$$

   $$Pr[P[i] = 0] = Pr[P[i] = 0 | y = 0] \cdot Pr[y = 0] + Pr[P[i] = 0 | y \neq 0] \cdot Pr[y \neq 0]$$

   Además,

$$Pr[P[i] = 0|y = 0] = Pr[r_j = 0] = \frac{1}{2}$$

$$Pr[P[i] = 0|y \neq 0] \leq Pr[r_j = 1] = \frac{1}{2}$$

Por lo tanto

$$Pr[P[i] = 0] \leq \frac{1}{2} \cdot Pr[y = 0] + \frac{1}{2} \cdot Pr[y \neq 0] = \frac{1}{2}$$

Por lo tanto se concluye que la probabilidad de que el algoritmo falle es menor a $\frac{1}{2}$

Problema 2

A necesita enviar un mensaje de m bits a B, quien conoce el largo del mensaje. Para esto, A representa el mensaje como un número M, define $p = \frac{M}{2^m}$ y luego envía a B un 1 con probabilidad p y 0 con probabilidad $(1 - p)$ todas las veces que sean necesarias. Luego B calcula el promedio de los primeros t mensajes y determina el entero M' que minimiza la distancia entre $M'/2^m$ y el promedio obtenido. B asume que el mensaje enviado es M'. ¿Cuanto debe ser t para que la probabilidad de error sea menor a ϵ?

Solución: Sean $X_1 \dots X_t$ los mensajes enviados y $\overline{X_t} = \frac{X_1 + \cdots + X_t}{t}$. Tenemos que:

$$E[\overline{X_t}] = \frac{E[X_1] + \cdots + E[X_t]}{t} = \frac{t \cdot p}{t} = p$$

Para estimar el mensaje M', lo que hace B es simplemente calcular $\overline{X_t} \cdot 2^m$ y aproximar al entero más cercano. Eso minimiza la distancia entre $\frac{M'}{2^m}$ y $\overline{X_t}$. B se equivoca cuando $|\overline{X_t} \cdot 2^m - M| \geq \frac{1}{2}$ ya que en ese caso al aproximar, se llega a otro valor:

$$\begin{aligned}
Pr[error] &= Pr\left[\left|\overline{X_t} \cdot 2^m - M\right| \geq \frac{1}{2}\right] \\
&= Pr\left[\left|\overline{X_t} \cdot 2^m - p \cdot 2^m\right| \geq \frac{1}{2}\right] \\
&= Pr\left[\left|\overline{X_t} - p\right| \cdot 2^m \geq \frac{1}{2}\right] \\
&= Pr\left[\left|\overline{X_t} - p\right| \geq \frac{1}{2 \cdot 2^m}\right] \\
&= Pr\left[\left|\overline{X_t} - E[\overline{X_t}]\right| \geq \frac{1}{2 \cdot 2^m}\right]
\end{aligned}$$

Este probabilidad puede ser acotada usando la desigualdad de Chebyshev:

$$Pr[error] = Pr\left[\left|\overline{X_t} - E[\overline{X_t}]\right| \geq \frac{1}{2 \cdot 2^m}\right] \leq Var(\overline{X_t}) \cdot (2 \cdot 2^m)^2$$

$$\begin{aligned}
Var(\overline{X_t}) &= \left(\frac{1}{t}\right)^2 \cdot (Var(X_1) + \cdots + Var(X_t)) \\
&= \left(\frac{1}{t}\right)^2 \cdot t \cdot (p(1-p))
\end{aligned}$$

$$= \frac{p(1-p)}{t}$$

Así, necesitamos que

$$\frac{p(1-p)}{t} \cdot (2 \cdot 2^m)^2 \leq \epsilon$$

Notar que $p(1-p) \leq \frac{1}{4}$, por lo tanto:

$$\frac{(2 \cdot 2^m)^2}{4t} \leq \epsilon$$

$$t \geq \frac{(2^m)^2}{\epsilon}$$

Problema 3

Se tiene un conjunto S de n elementos. Además se tienen cojunton $S_1, \ldots, S_k$ donde $S_i \subseteq S$ y $|S_i| > r$ $\forall i \in 1, \ldots, k$. Dado conjuntos de esta forma, donde $k < 2^{r-2}$, se necesita asignar uno de dos colores a cada elemento, de forma de que cada S_i tenga al menos un elemento de cada color.

- Describa un algoritmo de monte carlo que permita encontrar una coloración como la descrita en tiempo lineal y acote su probabilidad de error.

  Solución:

  - El algoritmo es simplemente asignar colores de manera aleatoria a los elementos de S. Para acotar la probabilidad de error, definamos las siguientes variables auxiliares:

$$C_i = \begin{cases} 1 & \text{todos los elementos de } S_i \text{ tienen color 0} \\ 0 & \text{en otro caso} \end{cases}$$

$$U_i = \begin{cases} 1 & \text{todos los elementos de } S_i \text{ tienen color 1} \\ 0 & \text{en otro caso} \end{cases}$$

  - Se define la operación $A \cup B$ para dos variables aleatorias binarias donde:

$$A \cup B = \begin{cases} 1 & A = 1 \vee B = 1 \\ 0 & A = 0 \wedge B = 0 \end{cases}$$

$$Pr[A \cup B = 1] = Pr[A = 1 \wedge B = 1] + Pr[A = 1 \wedge B = 0] + Pr[A = 0 \wedge B = 1]$$

$$Pr[A \cup B = 1] = Pr[A = 1] + Pr[A = 0 \wedge B = 1] \leq Pr[A = 1] + Pr[B = 1]$$

  - Con esto podemos acotar la probabilidad de error:

$$Pr[error] = Pr\left[\bigcup_{i=1}^{k} (C_i \cup U_i)\right]$$

$$\leq \sum_{i=1}^{k} \left(\left(\frac{1}{2}\right)^r + \left(\frac{1}{2}\right)^r \right)$$

$$= k \cdot \left(\frac{1}{2}\right)^{r-1}$$

$$\leq 2^{r-2} \cdot \left(\frac{1}{2}\right)^{r-1} = \frac{1}{2}$$

- Describa un algoritmo de las vegas que permita encontrar una coloración como la descrita en tiempo lineal, y acote la probabilidad de que entregue `sin resultado`.

  Solución: Para hacer un algoritmo de las vegas basta con revisar si la coloración es correcta luego de asignar los colores. Si la coloración es correcta simplemente se retorna y sino, se retorna `no hay solución`.