Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2283 - Diseño y Análisis de Algoritmos
Profesor: Nicolás Van Sint Jan
Ayudante: Dante Pinto

Ayudantía 9

Algoritmos aleatorizados

Problema 1: Multiplicación de Matrices

Sean A, B, $C \in \mathbb{Q}^{n \times n}$. Queremos determinar si $A \cdot B = C$.

1. Diseñe un algoritmo determinista que resuelva el problema y caracterice su tiempo de ejecución.

   Solución: El algoritmo más simple que podemos utilizar es simplemente multiplicar las matrices A y B y luego comparar el resultado, entrada por entrada, con la matriz C. Considerando la suma y multiplicación de racionales como operación a contar y utilizando la forma más simple de multiplicación de matrices, este algoritmo tomará tiempo cúbico respecto a n.

2. Diseñe un algoritmo aleatorizado que resuelva el problema con un mejor tiempo que el algoritmo anterior.

   Solución: En lugar de multiplicar directamente las dos matrices podemos generar, con distribución uniforme, un vector cualquiera v, calcular $A \cdot B \cdot v - C \cdot v$ y luego comparar este valor con el vector 0.

   La multiplicación de una matriz por un vector toma tiempo cuadrático, por lo que si calculamos $B \cdot v$ y luego multiplicamos el resultado de esta operación por A, el algoritmo tomará tiempo cuadrático.

3. Calcule la probabilidad de error del algoritmo 2.

   Solución: Formalizando el algoritmo anterior, tenemos:

```
1 Function Compare(A, B, C)
2 |  generate(v) ∈ {0,1}^n
3 |  c = C · v
4 |  b = B · v
5 |  a = a · b
6 |  r = a − c
7 |  if r = 0⃗ then
8 |  |  return true
9 |  return false
```

   En caso que $r \neq \vec{0}$, es imposible que $A \cdot B = C$, sin importar qué valor tenga el vector v, por tanto el algoritmo solo cometerá errores cuando retorne `true` y $A \cdot B \neq C$, por lo que solo debemos analizar este caso.

   En otras palabras, queremos calcular la probabilidad de que $r = \vec{0}$ cuando $A \cdot B \neq 0$. Si definimos $D = A \cdot B - C$, es claro que $r = D \cdot v$ y además que alguna de las entradas de D, d_{i_0,j_0} debe ser no nula, por lo que podemos acotar la probabilidad que buscamos por la probabilidad de que la entrada r_{i_0} sea 0.

   El valor de r_{i_0} está dado por:

$$r_{i_0} = \sum_{i=1}^{n} d_{i,j} \cdot v_i = d_{i_0,j_0} \cdot r_{i_0} + k \quad \wedge \quad k \in \mathbb{N}$$

$$\Rightarrow Pr[r_{i_0} = 0] = Pr[r_{i_0} = 0|k = 0] \cdot Pr[k = 0] + Pr[r_{i_0} = 0|k \neq 0] \cdot Pr[k \neq 0]$$

Calculando las probabilidades que buscamos podemos ver que:

$$Pr[r_{i_0} = 0|k = 0] = Pr[v_{i_0} = 0] = \frac{1}{2}$$
$$Pr[r_{i_0} = 0|k \neq 0] \leq Pr[v_{i_0} = 0] = \frac{1}{2}$$

Luego:

$$\begin{aligned} Pr[r = 0] &\leq Pr[r_{i_0} = 0] \\ &\leq \frac{1}{2} \cdot Pr[k = 0] + \frac{1}{2} \cdot Pr[k \neq 0] \\ &= \frac{1}{2} \left(Pr[k = 0] + Pr[k \neq 0]\right) \\ &= \frac{1}{2} \end{aligned}$$

Y por lo tanto la probabilidad de error del algoritmo será menor a un medio.

Problema 2: Descarga de monedas

Suponga que tiene acceso a una función `MonedaCargada` que retorna 0 con probabilidad p y 1 con probabilidad $1 - p$. Utilizando esta función:

1. Construya el algoritmo de Las Vegas `LanzarMoneda`, que retorna 0 con probabilidad $\frac{1}{2}$ y 1 con probabilidad $\frac{1}{2}$, es decir, funciona como una moneda no cargada.

   Solución:

```
1 Function LanzarMoneda()
2 |  m_1 = MonedaCargada()
3 |  m_2 = MonedaCargada()
4 |  if m_1 ≠ m_2 then
5 |  |  return m_1
6 |  return sin resultado
```

2. Demuestre que su algoritmo retorna los valores 0 y 1 con las probabilidades pedidas.

   Solución: Tenemos 4 resultados posibles para m_1 y m_2:

   (a) $m_1 = 0 \wedge m_2 = 0$: El algoritmo entrega sin resultado.

   (b) $m_1 = 1 \wedge m_2 = 1$: El algoritmo entrega sin resultado.

   (c) $m_1 = 0 \wedge m_2 = 1$: El algoritmo entrega 1.

   (d) $m_1 = 1 \wedge m_2 = 0$: El algoritmo entrega 0.

Luego:

$$\begin{aligned} Pr[\texttt{LM} = 0] &= Pr[m_1 = 0 \wedge m_2 = 1] \\ &= Pr[m_1 = 0] \cdot Pr[m_2 = 1] \\ &= p \cdot (1-p) \\ &= Pr[m_2 = 0] \cdot Pr[m_1 = 1] \\ &= Pr[m_2 = 0 \wedge m_1 = 1] \\ &= Pr[\texttt{LM} = 1] \end{aligned}$$

Por tanto, si el algoritmo entrega resultado, tendrá la misma probabilidad de entregar 0 y 1.

3. Calcule la probabilidad de fallo de su algoritmo.

   Solución: El algoritmo fallará en los casos (a) y (b) mostrados anteriormente, por tanto:

$$\begin{aligned} Pr[\texttt{LM} = \texttt{sin resultado}] &= Pr[(m_1 = 0 \wedge m_2 = 0) \vee (m_1 = 1 \wedge m_2 = 1)] \\ &= Pr[(m_1 = 0 \wedge m_2 = 0)] + Pr[(m_1 = 1 \wedge m_2 = 1)] \\ &= p^2 + (1-p)^2 \\ &= p^2 + 1 - 2p + p^2 \\ &= 1 - 2 \cdot p(1-p) \end{aligned}$$

4. Modifique el algoritmo para que siempre retorne un output 0 o 1. En promedio, ¿Cuántas veces se debe ejecutar `MonedaCargada`?

   Solución: Para que el algoritmo siempre entregue resultado bastará con volver a llamarlo cada vez que entregue sin resultado. La cantidad de veces que se ejecutará en promedio `MonedaCargada` corresponderá a la esperanza de una variable aleatoria con distribución geométrica y parámetro $p(1-p)$, por lo que la cantidad de llamadas a `MonedaCargada` será, en promedio, $\frac{1}{p(1-p)}$.

   Si no notamos la distribución de la variable, podemos calcular a mano esta esperanza como:

$$E[X] = E[X|\texttt{LM} = \texttt{sin resultado}] \cdot Pr[\texttt{LM} = \texttt{sin resultado}] + E[X|\texttt{LM} = 0 \vee 1] \cdot Pr[\texttt{LM} = 0 \vee 1]$$

   Ya calculamos anteriormente las probabilidades de entregar (o no) resultado, por lo que solamente nos faltaría calcular las esperanzas.

   Calculando primero $E[X|\texttt{LM} = 0 \vee 1$, es fácil ver que en cuando el algoritmo entrega output se realizan dos lanzamientos, por tanto esta esperanza tendrá valor 2. Por otra parte, si el algoritmo no entrega output, este realiza dos lanzamientos y luego se vuelve a ejecutar, es decir $E[X|\texttt{LM} = \texttt{sin resultado}] = (2 + E[X])$. Luego:

$$\begin{aligned} E[X] &= E[X|\texttt{LM} = \texttt{sin resultado}] \cdot Pr[\texttt{LM} = \texttt{sin resultado}] + E[X|\texttt{LM} = 0 \vee 1] \cdot Pr[\texttt{LM} = 0 \vee 1] \\ &= (2 + E[X]) \cdot Pr[\texttt{LM} = \texttt{sin resultado}] + 2 \cdot Pr[\texttt{LM} = 0 \vee 1] \\ &= (2 + E[X]) \cdot Pr[\texttt{LM} = \texttt{sin resultado}] + 2 \cdot (1 - Pr[\texttt{LM} = \texttt{sin resultado}]) \\ &= E[X] \cdot Pr[\texttt{LM} = \texttt{sin resultado}] + 2 \\ &= \frac{2}{1 - Pr[\texttt{LM} = \texttt{sin resultado}]} \\ &= \frac{2}{1 - (1 - 2 \cdot p(1-p))} \\ &= \frac{1}{p(1-p)} \end{aligned}$$

5. Construya el algoritmo `LanzarDado` que retorne un número entre 0 y 5 con probabilidad uniforme.

   Solución: Similar a lo que hicimos para la pregunta anterior, buscamos formar 6 eventos equiprobables, sin importar su probabilidad, y en caso de no caer en uno de estos 6 simplemente podemos retornar sin resultado.

   Una manera de hacer esto es hacer 3 lanzamientos independientes de una moneda (no cargada) y armar un string binario con ellos. En caso de que el número asociada al string binario esté entre 1 y 6, lo retornamos y si el valor encontrado es 0 o 7, retornamos `sin resultado`.

Problema 3: Comunicación aleatorizada

Una persona A desea enviarle un mensaje M, codificado como un string binario de m bits, pero por problemas de conexión solo puede enviar $n < m$ bits.

Para resolver lo anterior, A define $p = \frac{M}{2^m}$ y envía n bits, eligiendo un 1 con probabilidad p y un 0 con probabilidad $1 - p$, de manera independiente para cada bit.

1. Entregue un algoritmo que decodifica el mensaje de A a partir de los n bits y el largo m del mensaje.

   Solución: Los bits recibidos formarán un string binario $b = b_1...b_n$ y, utilizando los bits, podemos intentar estimar la probabilidad p que eligió A.

$$\overline{b} = \frac{1}{n}\sum_{i=1}^{n} b_i$$

   Con esta probabilidad estimada podemos multiplicarla por 2^m y luego redondear este valor (al entero más cercano) encontrando el mensaje decodificado. De esta forma calculamos:

$$M_0 = \overline{b} \cdot 2^m$$
$$M' = \texttt{round}(M_0)$$

2. Acote la probabilidad de que el mensaje decodificado sea incorrecto.

   Solución: Usando el método anterior, claramente fallaremos cuando $M' \leq M$, lo que puede ocurrir por dos razones, la primera es que el mensaje M_0 era similar a M, pero al redondear nos acercamos a un entero distinto de M y la segunda es que el mensaje calculado, M_0, simplemente no se parecía en nada al original. Estos dos casos pueden representarse por el mismo evento, la distancia entre M_0 y M es mayor o igual a 0.5.

$$\begin{aligned} Pr[error] &= Pr\left[|M_0 - M| \geq \frac{1}{2}\right] \\ &= Pr\left[|\overline{b} \cdot 2^m - p \cdot 2^m| \geq \frac{1}{2}\right] \\ &= Pr\left[|\overline{b} - p| \geq \frac{1}{2^{m+1}}\right] \end{aligned}$$

   La probabilidad anterior es bastante similar a la cota de Chebychev $\left(Pr[|X - E[X]| \geq a] \leq \frac{Var[X]}{a^2}\right)$, por lo que si encontramos la esperanza de nuestra variable aleatoria, $\overline{b}$, podremos aplicar la cota.

$$
\begin{aligned}
E[\bar{b}] &= E\left[\frac{1}{n}\sum_{i=0}^{n} b_i\right] \\
&= \frac{1}{n}\sum_{i=0}^{n} E\left[b_i\right] \\
&= \frac{1}{n}\sum_{i=0}^{n} p \\
&= p
\end{aligned}
$$

$$
\begin{aligned}
Pr[error] &= Pr\left[|\bar{b} - p| \geq \frac{1}{2^{m+1}}\right] \\
&= Pr\left[|\bar{b} - E[\bar{b}]| \geq \frac{1}{2^{m+1}}\right] \\
&\leq Var[\bar{b}] \cdot 2^{2m+2}
\end{aligned}
$$

Finalmente, calculando la varianza de $\bar{b}$ encontramos:

$$
\begin{aligned}
Var[\bar{b}] &= \frac{1}{n^2}\sum_{i=0}^{n} Var[b_i] \\
&= \frac{1}{n^2} \cdot n \cdot Var[b_i] \\
&= \frac{1}{n}(E[b_i^2] - E[b_i]^2) \\
&= \frac{1}{n}(E[b_i] - E[b_i]^2) \\
&= \frac{1}{n}(p - p^2) \\
\therefore Pr[error] &\leq \frac{p(1-p)}{n} \cdot 2^{2m+2}
\end{aligned}
$$

3. ¿Cuántos bits debe enviar A para que la probabilidad de error sea menor a ε? ¿Es eficiente la solución de A?

   Solución: Para calcular esto simplemente necesitamos reemplazar la probabilidad de error por ε y despejar:

$$
\begin{aligned}
\varepsilon &\leq \frac{p(1-p)}{n} \cdot 2^{2m+2} \\
\Rightarrow n &\geq \frac{p(1-p)}{\varepsilon} \cdot 2^{2m+2} \\
&\geq \frac{1}{4\varepsilon} \cdot 2^{2m+2} \\
&= \frac{2^{2m}}{\varepsilon}
\end{aligned}
$$

   La solución **no** es eficiente, pues, por ejemplo, para conseguir una probabilidad de error menor o igual a $\frac{1}{2}$ necesitaremos enviar 2^{2m+1} bits, que claramente es mayor que los m bits correspondientes a mandar el mensaje completo.