PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

Diseño y Análisis de Algoritmos - IIC2283
Interrogación 2
Tiempo: 2.5 horas

1. [1.5 puntos] En clases estudiamos el siguiente algoritmo aleatorizado para calcular la mediana de una lista de enteros $L[1 \dots n]$, suponiendo que n es impar y L no tiene elementos repetidos:

CalcularMediana($L[1 \dots n]$)
if $n < 2001$ **then**
Mergesort(L)
return $L[\lceil \frac{n}{2} \rceil]$
else
sea R una lista de $\lceil n^{\frac{3}{4}} \rceil$ números enteros escogido con
distribución uniforme y de manera independiente desde L
Mergesort(R)
$d := R[\lfloor \frac{1}{2} \cdot n^{\frac{3}{4}} - n^{\frac{1}{2}} \rfloor]$
$u := R[\lceil \frac{1}{2} \cdot n^{\frac{3}{4}} + n^{\frac{1}{2}} \rceil]$
$S := \emptyset$
$m_d := 0$
$m_u := 0$
for $i := 1$ **to** n **do**
if $d \leq L[i]$ **and** $L[i] \leq u$ **then** **Append**($S, [L[i]]$)
else if $L[i] < d$ **then** $m_d := m_d + 1$
else $m_u := m_u + 1$
if $m_d \geq \lceil \frac{n}{2} \rceil$ **or** $m_u \geq \lceil \frac{n}{2} \rceil$ **or**
Length(S) $> 4 \cdot \lfloor n^{\frac{3}{4}} \rfloor$ **then return** *sin_resultado*
else
Mergesort(S)
return $S[\lceil \frac{n}{2} \rceil - m_d]$

Suponiendo que $n \geq 2001$ y m es la mediana de la lista L, defina Y_1 como la siguiente variable aleatoria:

$$Y_1 \;=\; |\{i \in \{1, \dots, \lceil n^{\frac{3}{4}} \rceil\} \mid R[i] \leq m\}|.$$

En esta pregunta usted debe demostrar que:

$$\Pr(Y_1 < \lfloor \frac{1}{2} \cdot n^{\frac{3}{4}} - n^{\frac{1}{2}} \rfloor) \;\leq\; n^{-\frac{1}{4}}.$$

2. Dado un conjunto no vacío de variables V, una ecuación sobre V es una expresión de la forma:

$$\sum_{x \in W} x \quad = \quad \ell, \tag{1}$$

donde W es un subconjunto no vacío de V y $\ell \in \mathbb{N}$. Por ejemplo, $x_1+x_3+x_7 = 2$ y $x_5+x_8 = 23$ son ecuaciones sobre el conjunto de variables $\{x_1, \ldots, x_{10}\}$. Una solución para una ecuación de la forma (1) sobre un conjunto de variables V es una función $\rho : V \to \{0, 1\}$ tal que:

$$\sum_{x \in W} \rho(x) \quad = \quad \ell.$$

Por ejemplo, considere la ecuación $x_1+x_3+x_7 = 2$ sobre el conjunto de variables $\{x_1, \ldots, x_{10}\}$. Una función $\rho_1 : \{x_1, \ldots, x_{10}\} \to \{0, 1\}$ definida como $\rho_1(x_1) = \rho_1(x_3) = 1$ y $\rho_1(x_i) = 0$ para cada $i \in \{2, 4, 5, 6, 7, 8, 9, 10\}$ es una solución para esta ecuación, mientras que $\rho_2 : \{x_1, \ldots, x_{10}\} \to \{0, 1\}$ definida como $\rho_2(x_j) = 1$ para cada $j \in \{1, \ldots, 10\}$ no lo es. Como otro ejemplo, note que la ecuación $x_5 + x_8 = 23$ sobre el conjunto de variables $\{x_1, \ldots, x_{10}\}$ no tiene solución.

Un sistema de ecuaciones S sobre un conjunto de variables V es una secuencia e_1, …, e_m que satisface las siguientes condiciones: (i) $m \geq 1$; (ii) para cada $i \in \{1, \ldots, m\}$, se tiene que e_i es una ecuación sobre V; y (iii) para cada variable $x \in V$, existe $j \in \{1, \ldots, m\}$ tal que la variable x es mencionada en la ecuación e_j. Además, una función $\rho : V \to \{0, 1\}$ es una solución para el sistema de ecuaciones S si existe $i \in \{1, \ldots, m\}$ tal que ρ es una solución para la ecuación e_i. En esta pregunta vamos a desarrollar un algoritmo aleatorizado para aproximar el número de soluciones de un sistema de ecuaciones $S = e_1, \ldots, e_m$ sobre un conjunto de variables V, lo cual es denotado como $\#\mathrm{SOL}(S)$.

(a) [0.2 puntos] Construya un algoritmo de tiempo polinomial que verifique si la ecuación e_i tiene solución ($1 \leq i \leq m$).

Dada la existencia del algoritmo en (a), podemos eliminar desde S las ecuaciones que no tienen solución. Si todas las ecuaciones de S son eliminadas, entonces S no tiene solución y el algoritmo que estamos desarrollando en esta pregunta retorna 0.

Dado lo anterior, desde ahora en adelante vamos a suponer que $S = e_1, \ldots, e_m$ donde cada ecuación e_i tiene solución ($1 \leq i \leq m$).

(b) [0.4 puntos] Defina $E_i = |\{\rho : V \to \{0, 1\} \mid \rho \text{ es solución de la ecuación } e_i\}|$, para cada $i \in \{1, \ldots, m\}$. Construya un algoritmo de tiempo polinomial que retorne E_i, y justifique por qué su algoritmo funciona en este tiempo.

(c) Defina $M = \sum_{i=1}^{m} E_i$ y $N(\rho) = |\{i \in \{1, \ldots, m\} \mid \rho \text{ es solución de la ecuación } e_i\}|$ para cada función $\rho : V \to \{0, 1\}$. Utilizando estos valores, defina una variable aleatoria X tal que para cada función $\rho : V \to \{0, 1\}$ se tiene que:

$$X(\rho) \quad = \quad \begin{cases} \frac{M}{N(\rho)} & \rho \text{ es solución de } S \\ 0 & \text{en caso contrario} \end{cases}$$

Para obtener un valor de X se realiza los siguientes pasos: escoger $i \in \{1, \ldots, m\}$ con probabilidad $\frac{E_i}{M}$, luego escoger con distribución uniforme una función $\rho : V \to \{0, 1\}$ tal que ρ es solución de la ecuación e_i, y finalmente retornar $X(\rho)$. Demuestre las siguientes propiedades.

(c.1) [0.4 puntos] Para cada función $\rho : V \to \{0,1\}$ que es solución de S se tiene que $\Pr(\rho) = \frac{N(\rho)}{M}$

(c.2) [0.3 puntos] $\mathrm{E}(X) = \#\mathrm{SOL}(S)$

(d) [0.5 puntos] Suponga que $X_1, \ldots. X_k$ son k variables aleatorias independientes, y suponga que cada una de ellas es idéntica a X. Demuestre que:

$$\mathrm{E}\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right) \quad = \quad \mathrm{E}(X)$$

Además, dado $\varepsilon \in (0,1)$, demuestre que:

$$\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathrm{E}(X)\right| \geq \varepsilon \cdot \mathrm{E}(X)\right) \quad \leq \quad \frac{\mathrm{Var}(X)}{k \cdot \varepsilon^2 \cdot \mathrm{E}(X)^2}$$

(e) [0.2 puntos] Demuestre que para cada función $\rho : V \to \{0,1\}$ tal que ρ es una solución de S, se tiene que $\frac{M}{m} \leq X(\rho) \leq M$. A partir de esto demuestre que $\frac{M}{m} \leq \mathrm{E}(X) \leq M$.

(f) [0.5 puntos] Demuestre que:

$$\mathrm{Var}(X) \quad \leq \quad \left(\frac{M}{m}\right)^2 \cdot (m-1)^2$$

A partir de esto demuestre que:

$$\frac{\mathrm{Var}(X)}{\mathrm{E}(X)^2} \quad \leq \quad (m-1)^2$$

(g) [0.5 puntos] Utilizando los resultados anteriores demuestre que para cada $\varepsilon \in (0,1)$:

$$\Pr\left(\frac{|\frac{1}{k}\sum_{i=1}^{k} X_i - \#\mathrm{SOL}(S)|}{\#\mathrm{SOL}(S)} < \varepsilon\right) \quad \geq \quad 1 - \frac{(m-1)^2}{k \cdot \varepsilon^2}$$

En particular, indique qué valor del número de muestras k nos asegura que vamos a tener un error menor al 1 % con probabilidad $\frac{99}{100}$, vale decir, qué valor de k asegura que:

$$\Pr\left(\frac{|\frac{1}{k}\sum_{i=1}^{k} X_i - \#\mathrm{SOL}(S)|}{\#\mathrm{SOL}(S)} < \frac{1}{100}\right) \quad \geq \quad \frac{99}{100}$$

3. [1.5 puntos] En esta pregunta consideramos listas de números naturales mayores a cero, con al menos un elemento y sin repeticiones. Dadas dos listas L_1 y L_2 con n elementos, definimos $\mathbf{EXP}(L_1, L_2)$ de la siguiente forma:

$$\prod_{i=1}^{n} L_1[i]^{L_2[i]}.$$

Por ejemplo, si $L_1 = [2, 1, 5]$ y $L_2 = [10, 2, 4]$, entonces:

$$\mathbf{EXP}(L_1, L_2) \ = \ 2^{10} \cdot 1^2 \cdot 5^4 \ = \ 640000.$$

Además, dadas dos listas L_1 y L_2 con n elementos, y dos biyecciones $\pi_1, \pi_2 : \{1, \ldots, n\} \to \{1, \ldots, n\}$, definimos $\mathbf{PERM\text{-}EXP}(L_1, \pi_1, L_2, \pi_2)$ como:

$$\prod_{i=1}^{n} L_1[\pi_1(i)]^{L_2[\pi_2(i)]}.$$

Por ejemplo, si $L_1 = [2, 1, 5]$, $L_2 = [10, 2, 4]$, π_1 es definida como $\pi_1(1) = 2$, $\pi_1(2) = 3$ y $\pi_1(3) = 1$, y π_2 es definida como $\pi_2(1) = 3$, $\pi_2(2) = 2$ y $\pi_2(3) = 1$, entonces se tiene que:

$$\begin{aligned}
\mathbf{PERM\text{-}EXP}(L_1, \pi_1, L_2, \pi_2) \ &= \ L_1[\pi_1(1)]^{L_2[\pi_2(1)]} \cdot L_1[\pi_1(2)]^{L_2[\pi_2(2)]} \cdot L_1[\pi_1(3)]^{L_2[\pi_2(3)]} \\
&= \ L_1[2]^{L_2[3]} \cdot L_1[3]^{L_2[2]} \cdot L_1[1]^{L_2[1]} \\
&= \ 1^4 \cdot 5^2 \cdot 2^{10} \ = \ 25600.
\end{aligned}$$

En esta pregunta usted debe construir un algoritmo codicioso que, dadas dos listas L_1 y L_2 con n elementos, retorna dos biyecciones $\pi_1, \pi_2 : \{1, \ldots, n\} \to \{1, \ldots, n\}$ que maximizan el valor de $\mathbf{PERM\text{-}EXP}(L_1, \pi_1, L_2, \pi_2)$. Su algoritmo debe funcionar en tiempo polinomial, y además debe demostrar que funciona correctamente.