Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2283 - Diseño y Análisis de Algoritmos - 2$^{\text{do}}$ semestre 2022

PAUTA INTERROGACIÓN 2

Pregunta 1

Considere la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} n & n \leq 5 \\ 5T\left(\lfloor \frac{n}{3} \rfloor\right) + T\left(\lceil \frac{n}{4} \rceil\right) + n^2 \cdot \log n & n > 5 \end{cases}$$

Encuentre una función $f : \mathbb{N} \to \mathbb{N}$ tal que $T(n) \in \Theta(f)$. Debe demostrar que la propiedad se cumple.

Solución. Defina $T_1 : \mathbb{N} \to \mathbb{N}$ y $T_2 : \mathbb{N} \to \mathbb{N}$ como

$$T_1(n) = \begin{cases} 5 & n \leq 5 \\ 6T\left(\lfloor \frac{n}{4} \rfloor\right) + n^2 \cdot \log n & n > 5 \end{cases} \qquad T_2(n) = \begin{cases} 5 & n \leq 5 \\ 6T\left(\lfloor \frac{n}{3} \rfloor\right) + n^2 \cdot \log n & n > 5 \end{cases}$$

Es claro que para $n > 5$ se cumple que $T_1(n) \leq T(n) \leq T_2(n)$ (hay puntaje adicional si se demuestra esto formalmente). Luego, para usar el **tercer caso** del Teorema Maestro con $T_1(n)$ y $T_2(n)$, hay que mostrar que la función $n^2 \cdot \log n$ es $(6, 4)$-regular y $(6, 3)$-regular:

$$\begin{aligned} a \cdot \left(\left\lfloor \frac{n}{b} \right\rfloor\right)^2 \log\left(\left\lfloor \frac{n}{b} \right\rfloor\right) &\leq a \cdot \left(\frac{n}{b}\right)^2 \log\left(\frac{n}{b}\right) \\ &= \frac{a}{b^2} \cdot n^2(\log(n) - \log(b)) \\ &\leq \frac{a}{b^2} \cdot n^2 \log(n) \end{aligned}$$

Por lo que siempre que $a < b^2$, podemos tomar la constante $c = \frac{a}{b^2} < 1$, demostrando así la (a, b)-regularidad. En este caso, como $6 < 3^2 < 4^2$, entonces $n^2 \cdot \log n$ es $(6, 4)$-regular y $(6, 3)$-regular.
Ahora, sabemos que $n^2 \cdot \log n \in \Omega(n^2)$, y además, el valor de $\log_4(6)$ y $\log_3(6)$ está entre 1 y 2. Con esto, podemos encontrar fácilmente un $\varepsilon > 0$ tal que se cumpla la condición pedida en el Teorema Maestro, y por lo tanto, $T_1(n) \in \Theta(n^2 \cdot \log n)$ y $T_2(n) \in \Theta(n^2 \cdot \log n)$. Con esto, finalmente $T(n) \in \Theta(n^2 \cdot \log n)$.

Rúbrica. Dado lo anterior, la atribución de puntaje es la siguiente:

(**2 Puntos**)	Por definir correctamente $T_1(n)$ y $T_2(n)$ tal que $T_1(n) \leq T(n) \leq T_2(n)$, para $n \geq n_0$.
(**3 Puntos**)	Por mostrar que $n^2 \cdot \log n$ es (a, b)-regular para ambos casos.
(**1 Punto**)	Por concluir correctamente usando el Teorema Maestro que $T(n) \in \Theta(n^2 \log n)$.
(**0.5 Puntos**)	**BONUS:** Por demostrar **formalmente** que $T_1(n) \leq T(n) \leq T_2(n)$ a partir de algún n en adelante.
(**Consideración**)	El puntaje máximo a obtener en la pregunta sigue siendo 6 puntos, independiente del **BONUS**.

Pregunta 2

El siguiente es el algoritmo visto en clases para calcular la mediana de una lista en tiempo $\mathcal{O}(n)$:

CalcularMediana$(L[1 \ldots n])$
  if $n < 2001$ **then**
    Mergesort(L)
    return $L[\lceil \frac{n}{2} \rceil]$
  else
    sea R una lista de $\left\lceil n^{\frac{3}{4}} \right\rceil$ números enteros escogido con
        distribución uniforme y de manera independiente desde L
    Mergesort(R)
    $d := R\left[\left\lfloor \frac{1}{2} \cdot n^{\frac{3}{4}} - n^{\frac{1}{2}} \right\rfloor\right]$; $u := R\left[\left\lceil \frac{1}{2} \cdot n^{\frac{3}{4}} + n^{\frac{1}{2}} \right\rceil\right]$
    $S := \varnothing$; $m_d := 0$; $m_u := 0$
    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**$(S) > 4 \cdot \left\lfloor n^{\frac{3}{4}} \right\rfloor$ **then**
      return *sin_resultado*
    else
      Mergesort(S)
      return $S\left[\lceil \frac{n}{2} \rceil - m_d\right]$

1. Explique los pasos que realiza el algoritmo **CalcularMediana** cuando se retorna un resultado.
2. Indique cuándo el algoritmo **CalcularMediana** retorna *sin_resultado* y la razón de porqué no se entrega resultado.
3. Explique por qué el algoritmo **CalcularMediana** es correcto. No es necesaria una demostración matemática, más bien indique cuáles son las ideas centrales que muestran que el algoritmo es correcto.

Solución. A continuación se muestra una posible demostración para cada inciso:

1. La respuesta debe mencionar los siguientes conceptos:
   (a) Primero, para n pequeños, el algoritmo simplemente ordena la lista usando **Mergesort**, esto ya que en un análisis del algoritmo permite acotar la probabilidad de error de manera más fácil y porque no influye en la complejidad $\mathcal{O}(n)$ del algoritmo completo.
   (b) Luego se eligen $\lceil n^{3/4} \rceil$ números de manera uniforme e independiente desde L y se colocan en una nueva lista R que se ordena. Esto toma tiempo $\mathcal{O}(n^{3/4} \log n)$.
   (c) Se eligen dos índices d y u de la lista R, cada uno separado a una distancia de $\sqrt{n}$ de la mediana de la lista R.
   (d) Se clasifican cada uno de los elementos de L respecto a como se comparan con los elementos de R. En particular, se guardan todos los elementos x de L que cumplan con $d \leq x \leq u$ en una nueva lista S, y la cantidad de elementos menores y mayores que d y u se almacenan en m_d y m_u, respectivamente.
   (e) Si se cumple que $d \leq L'[\lceil n/2 \rceil] \leq u$ (con L' la lista L ordenada), entonces la mediana de L está en S. Luego, si es que el tamaño de S es $\mathcal{O}(n^{3/4})$, se procede a ordenar S y se retorna el elemento que corresponderá a la mediana de L.
2. La respuesta debe mencionar que el algoritmo no entrega resultado en tres casos:
   - Cuando m_d o m_u son mayores o iguales a $\lceil n/2 \rceil$. Esto porque en ambos casos necesariamente la mediana de L no estará dentro de S, puesto que el elemento más pequeño (o más grande) de S será necesariamente mayor (o menor) que la mediana de L. Como la mediana de L no estará en S, es imposible que el resto del algoritmo entregue la mediana de L correctamente.
   - Cuando el tamaño de S es mayor que $4 \cdot \lfloor n^{3/4} \rfloor$ (4 es una constante fija que podría haber sido otro número cualquiera). Esto es para asegurar que el tamaño de S es $\mathcal{O}(n^{3/4})$, dado que en el paso siguiente se ordena S y es necesario que eso tome tiempo $\mathcal{O}(n^{3/4} \log n)$ que es sublineal.
3. La respuesta debe mencionar los siguientes conceptos:

- Siempre en cada paso se toma tiempo $\mathcal{O}(n)$, evitando la complejidad $\mathcal{O}(n \log n)$.
- La probabilidad de que lista S pueda contener la mediana de la lista L puede acotarse por una constante menor a 1 (usando la desigualdad de Chebyshev).
- Cuando la mediana de L está en S entonces siempre se puede encontrar el resultado correcto. Si L' es la lista L ordenada, la posición $L'[\lceil n/2 \rceil]$ corresponde a la mediana de L. Entonces, S va a corresponder al rango $L'[m_d \dots m_d + \textbf{Length}(S)]$, luego claramente $L'[\lceil n/2 \rceil] = S[\lceil n/2 \rceil - m_d]$.

Rúbrica. Dado lo anterior, la atribución de puntaje es la siguiente:

En ítem 1.

(**0.4 Puntos**) Por cada concepto mencionado correctamente **(2 Puntos en Total)**.

En ítem 2.

(**1 Punto**) Por explicar correctamente los casos de $m_d \geq \lceil \frac{n}{2} \rceil$ y $m_u \geq \lceil \frac{n}{2} \rceil$.

(**1 Punto**) Por explicar correctamente el caso de $\textbf{Length}(S) > 4 \cdot \left\lfloor n^{\frac{3}{4}} \right\rfloor$.

En ítem 3.

(**0.6 Puntos**) Por cada uno de los primeros 2 conceptos mencionados correctamente **(1.2 Puntos en Total)**.

(**0.8 Puntos**) Por el último concepto mencionado correctamente.

Pregunta 3

Considere el siguiente algoritmo aleatorizado para verificar si un número es primo:

TestPrimalidad(n, k)
  if $n = 2$ **then return** PRIMO
  else if $n \equiv 0 \pmod 2$ **then return** COMPUESTO
  else if **EsPotencia**(n) **then return** COMPUESTO
  else
    sea $a_1, \dots, a_k$ una lista de números elegidos de
      manera uniforme e independiente desde $\{1, \dots, n-1\}$
    for $i := 1$ **to** k **do**
      if **MCD**$(a_i, n) > 1$ **then return** COMPUESTO
      $b_i :=$ **EXP**$(a_i, \frac{n-1}{2}, n)$
    $neg := 0$
    for $i := 1$ **to** k **do**
      if $b_i \equiv -1 \pmod n$ **then** $neg := neg + 1$
      else if $b_i \not\equiv 1 \pmod n$ **then return** COMPUESTO
    if $neg = 0$ **then return** COMPUESTO
    return PRIMO

1. Indique qué recibe como entrada, cómo se puede equivocar y cuál es la probabilidad de error de **TestPrimalidad**.
2. Explique los pasos del algoritmo **TestPrimalidad**.
3. Explique por qué el algoritmo **TestPrimalidad** es correcto. No es necesaria una demostración matemática, más bien indique cuáles son las ideas centrales que permiten acotar la probabilidad de error del algoritmo.

Solución. A continuación se muestra una posible demostración para cada inciso:

1. El algoritmo recibe dos números $n, k \in \mathbb{N}$, siendo n el número que se desea verificar si es primo, y k un parámetro que modula la certeza que tendrá el algoritmo de una respuesta correcta.

   El algoritmo se equivoca en dos casos:

   - Si n es primo y ocurre que la lista $a_1, \dots, a_k$ es tal que todo $b_i \equiv 1 \pmod n$, entonces el algoritmo se equivoca respondiendo COMPUESTO.

- Si n es compuesto y ocurre que:
  - No es par ni de la forma $n = a^b$.
  - La lista $a_1, \ldots, a_k$ es tal que
    - $a_i \in \mathbb{Z}_n^*$ para todo $1 \leq i \leq k$.
    - $b_i \equiv 1 \pmod{n}$ o $b_i \equiv -1 \pmod{n}$ para todo $1 \leq i \leq k$.
    - Existe $1 \leq i \leq k$ tal que $b_i \equiv -1 \pmod{n}$.

  Entonces el algoritmo se equivoca respondiendo PRIMO.

La probabilidad de error tanto de entregar como respuesta PRIMO dado que n no era primo, y de entregar COMPUESTO dado que n sí era primo, será $\left(\frac{1}{2}\right)^k$.

2. La respuesta debe mencionar los siguientes conceptos:
   (a) Primero, se verifica si n no es par, respondiendo acordemente.
   (b) Luego, se verifica si $n = a^b$ para algún $a, b \in \mathbb{N}$. Este paso se puede realizar en tiempo $\mathcal{O}(\log^3(n))$. Si n es compuesto y resulta no ser de la forma $n = a^b$, entonces es seguro que $n = n_1 \cdot n_2$, con $\gcd(n_1, n_2) = 1$.
   (c) Se eligen k números menores que n de manera uniforme e independiente. Por cada número a_i elegido se verifica si $\gcd(a_i, n) > 1$ usando el algoritmo de Euclides (tiempo $\mathcal{O}(k \log n)$). Si alguno resulta no ser primo relativo con n entonces necesariamente n es compuesto.
   (d) Dado que $\gcd(a_i, n) = 1$ para todo a_i, luego $a_i \in \mathbb{Z}_n^*$. Se computa $b_i = a_i^{\frac{n-1}{2}} \mod n$ usando exponenciación rápida (tiempo $\mathcal{O}(k \log n)$).
   (e) Para cada b_i se verifica si $b_i \equiv -1 \pmod{n}$, es decir, si $a_i \in S_n^-$. Si es cierto esto, se incrementa el contador neg.
   (f) Para cada b_i se verifica si $b_i \not\equiv -1 \pmod{n}$ y $b_i \not\equiv 1 \pmod{n}$. De ser cierto en ambos casos, entonces $a_i \notin S_n$ y n debe ser compuesto.
   (g) Finalmente, si todos los a_i cumplen con $a_i \in S_n$: si se encontró un $b_i \equiv -1 \pmod{n}$, se responde que n es primo, y si no, se responde que n es compuesto. Esta es la única parte en que el algoritmo puede cometer un error.

3. La respuesta debe mencionar los siguientes conceptos:
   - Cuando el algoritmo responde COMPUESTO por ser par, de la forma a^b, o no ser primo relativo de n, son claramente respuestas correctas por definición de números primos.
   - Dado que para n primo se cumple $|S_n^+| = |S_n^-| = \frac{n-1}{2}$, entonces si existe $b_i \not\equiv -1 \pmod{n}$ y $b_i \not\equiv 1 \pmod{n}$, claramente n es compuesto y el algoritmo no se equivoca en ese caso. Por la misma razón, para que el algoritmo responda COMPUESTO y se equivoque debe ocurrir que todos los $a_i \in S_n^+$, lo que posee probabilidad $\left(\frac{1}{2}\right)^k$.
   - Para que n se equivoque respondiendo PRIMO, debe ocurrir que $n = n_1 \cdot n_2$ con $\gcd(n_1, n_2) = 1$ (ya que $n \neq a^b$ para algún par a, b). Por lo que se sabe que $|S_n| \leq \frac{1}{2}|\mathbb{Z}_n^*|$, así que la probabilidad de elegir solo $a_i \in S_n$, dado que al menos se eligió un $a_i \in S_n^-$, va a ser acotada por $\left(\frac{1}{2}\right)^k$.

Rúbrica. Dado lo anterior, la atribución de puntaje es la siguiente:

En ítem 1.

(**0.2 Puntos**) Por describir correctamente lo que recibe el algoritmo.

(**1.2 Puntos**) Por describir correctamente cuándo el algoritmo se equivoca.

(**0.6 Puntos**) Por mencionar la probabilidad de error del algoritmo.

En ítem 2.

(**2/7 Puntos**) Por cada concepto respondido correctamente **(2 Puntos en Total)**.

En ítem 3.

(**0.4 Puntos**) Por el primer concepto mencionado correctamente.

(**0.8 Puntos**) Por el segundo concepto mencionado correctamente.

(**0.8 Puntos**) Por el tercer concepto mencionado correctamente.

Pregunta 4

Demuestre que **Quicksort** en el caso promedio es $\Theta(n\log(n))$, considerando que la entrada del algoritmo es una lista de enteros no repetidos y que la operación básica a contar es la comparación entre enteros.

Solución. Considere $n \geq 2$ y sea $\mathcal{E}_n$ el conjunto de listas L con n elementos distintos sacados de $\{1,\ldots,n\}$. Sea $L \in \mathcal{E}_n$. Primero, se define la variable aleatoria X_n como:

$$X_n \quad := \quad \text{Número de comparaciones realizadas por la llamada } \textbf{Quicksort}(L,1,n)$$

Queremos demostrar que $\mathrm{E}(X_n) \in \Theta(n \cdot \log n)$. Para esto podemos definir una variable aleatoria para cada $i,j \in \{1,\ldots,n\}$ con $i \leq j$ tal que:

$$Y_{ij} \quad := \quad \text{Número de veces que } i \text{ es comparado con } j \text{ en la llamada } \textbf{Quicksort}(L,1,n)$$

Entonces se tiene lo siguiente:

$$X_n = \sum_{i=1}^{n}\sum_{j=1}^{n} Y_{ij}$$

Dado que la esperanza de una variable aleatoria es una función lineal:

$$\mathrm{E}(X_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{E}(Y_{ij})$$

Notar que dada la definición de **Partición** no es posible comparar un elemento consigo mismo. Por lo que $Y_{ij} = 0$ para $i = j$, luego $\mathrm{E}(Y_{ij}) = 0$ en ese caso.

Considere el caso general $1 \leq i < j \leq n$. Note que la única forma en que **Quicksort** puede comparar i con j es que el primer elemento que escoja **Partición** desde el conjunto $\{i, i+1, \ldots, j\}$ sea i o j, y se realiza a lo más una comparación.

Dicho eso, entonces se tiene que Y_{ij} es igual a 0 o a 1, y además que:

$$\Pr(Y_{ij} = 1) = \frac{2}{j-i+1}$$

dado que $|\{i, i+1, \ldots, j\}| = j-i+1$ y existen solo dos casos favorables: que $pivote = i$ y $pivote = j$. Luego:

$$\mathrm{E}(Y_{ij}) = 0 \cdot \Pr(Y_{ij} = 0) + 1 \cdot \Pr(Y_{ij} = 1) = \frac{2}{j-i+1}$$

Luego se concluye que:

$$\begin{aligned}
\mathrm{E}(X_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{E}(Y_{ij}) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \mathrm{E}(Y_{ij}) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1} \frac{2}{k} \\
&= \sum_{k=2}^{n} (n+1-k) \cdot \frac{2}{k} \\
&= 2 \cdot (n+1) \cdot \left(\sum_{k=2}^{n} \frac{1}{k}\right) - 2 \cdot (n-1) \\
&= 2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k}\right) - 4 \cdot n
\end{aligned}$$

Se puede acotar la sumatoria armónica $\sum_{k=1}^{n} \frac{1}{k}$ usando la integral $\int \frac{1}{x} dx$, pues se cumple que

$$\sum_{k=2}^{n} \frac{1}{k} \leq \int_{1}^{n} \frac{1}{x} dx \leq \sum_{k=1}^{n} \frac{1}{k}$$

Dado que $\int_{1}^{n} \frac{1}{x} dx = \ln(n) - \ln(1) = \ln(n)$ se puede concluir que:

$$\ln(n) \leq \sum_{k=1}^{n} \frac{1}{k} \leq \ln(n) + 1$$

Por lo tanto:

$$2 \cdot (n+1) \cdot \ln(n) - 4 \cdot n \leq \mathrm{E}(X_n) \leq 2 \cdot (n+1) \cdot (\ln(n) + 1) - 4 \cdot n$$

Y se concluye entonces que $\mathrm{E}(X_n) \in \Theta(n \cdot \log(n))$.

Rúbrica. Dado lo anterior, la atribución de puntaje es la siguiente:

- (**1 Punto**) Por mencionar que se va a trabajar con listas de $\mathcal{E}_n$.
- (**1 Punto**) Por definir correctamente Y_{ij} y hacer notar que X_n se puede escribir como una suma de Y_{ij}.
- (**2 Puntos**) Por encontrar el valor de $\mathrm{E}(Y_{ij})$ para cualquier i, j.
- (**1 Punto**) Por desarrollar $\mathrm{E}(X_n)$ y llegar al término con la sumatoria armónica.
- (**1 Punto**) Por acotar la sumatoria armónica por logaritmos, y concluir que $\mathrm{E}(X_n) \in \Theta(n \cdot \log(n))$.