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

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

EXAMEN

Puntaje: Cada pregunta posee el mismo puntaje. Cada ítem dentro de una misma pregunta posee el mismo puntaje. Ítems de preguntas entregadas en blanco se evaluarán con un puntaje correspondiente a obtener 0.5 puntos de 6 en ese ítem.

Pregunta 1

Considere la siguiente ecuación de recurrencia:

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

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

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 cuando 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.

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 $a_1, \ldots, a_k$ una lista de números elegidos de
        manera uniforme e independiente desde $\{1, \ldots, n-1\}$
    for $i := 1$ **to** k **do**
      if MCD$(a_i, n) > 1$ **then return** COMPUESTO
      $b_i :=$ **EXP**$(a_i, \frac{n-1}{2}, n)$
    $neq := 0$
    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.

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.

Formulario

Notación asintótica

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$ una función, definimos:

$$\mathcal{O}(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid \exists c \in \mathbb{R}^+. \exists n_0 \in \mathbb{N}. \forall n \geq n_0.\, g(n) \leq c \cdot f(n)\}$$

$$\Omega(f) = \{g : \mathbb{N} \to \mathbb{R}_0^+ \mid \exists c \in \mathbb{R}^+. \exists n_0 \in \mathbb{N}. \forall n \geq n_0.\, c \cdot f(n) \leq g(n)\}$$

$$\Theta(f) = \mathcal{O}(f) \cap \Omega(f)$$

Teorema Maestro

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$ una función y $a, b \in \mathbb{R}$ constantes tales que $a \geq 1$ y $b > 1$. La función f es **$(\mathbf{a}, \mathbf{b})$-regular** si existen constantes $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tales que $c < 1$ y

$$\forall n \geq n_0.\ a \cdot f\left(\left\lfloor \frac{n}{b} \right\rfloor\right) \leq c \cdot f(n)$$

Sea $f : \mathbb{N} \to \mathbb{R}_0^+$ una función, $a, b, c \in \mathbb{R}_0^+$ constantes tales que $a \geq 1$ y $b > 1$, y $T(n)$ una función definida por la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} c & n = 0 \\ a \cdot T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + f(n) & n \geq 1 \end{cases}$$

Se tiene que:

1. Si $f(n) \in \mathcal{O}\left(n^{\log_b(a) - \varepsilon}\right)$ para $\varepsilon > 0$, entonces $T(n) \in \Theta\left(n^{\log_b(a)}\right)$
2. Si $f(n) \in \Theta\left(n^{\log_b(a)}\right)$, entonces se cumple que $T(n) \in \Theta\left(n^{\log_b(a)} \cdot \log_2(n)\right)$
3. Si $f(n) \in \Omega\left(n^{\log_b(a) + \varepsilon}\right)$ para $\varepsilon > 0$ y f es (a, b)-regular, entonces $T(n) \in \Theta\left(f(n)\right)$

Medidas sobre variables aleatorias

Para una variable aleatoria X de recorrido $\Omega \subseteq \mathbb{R}$ la **esperanza** de X (denotada como $\mathrm{E}(X)$) se define como:

$$\mathrm{E}(X) = \sum_{r \in \Omega} r \cdot \Pr(X = r)$$

Por otro lado, la **varianza** de X (denotada como $\mathrm{Var}(X)$) se define como:

$$\mathrm{Var}(X) = \mathrm{E}((X - \mathrm{E}(X))^2) = \mathrm{E}(X^2) - \mathrm{E}(X)^2$$

Distribuciones útiles

Una variable aleatoria X sigue una distribución de Bernoulli de parámetro $0 < p < 1$ (denotada como $X \sim \mathrm{Bernoulli}(p)$) si:

$$\Pr(X = x) = \begin{cases} 1 - p & \text{si } x = 0 \\ p & \text{si } x = 1 \end{cases}$$

En particular se tiene que si $X \sim \mathrm{Bernoulli}(p)$ entonces:

$$\mathrm{E}(X) = p \qquad \mathrm{Var}(X) = p(1-p)$$

Desigualdades importantes

Sea X una variable aleatoria no negativa. Luego para todo $a \in \mathbb{R}^+$ se cumple la **desigualdad de Markov**:

$$\Pr(X \geq a) \ \leq \ \frac{\mathrm{E}(X)}{a}$$

Un caso particular de la desigualdad anterior es la **desigualdad de Chebyshev**:

$$\Pr(|X \geq \mathrm{E}(X)| \geq a) \ \leq \ \frac{\mathrm{Var}(X)}{a^2}$$

Máximo común divisor

Sean $a, b \in \mathbb{Z} - \{0\}$. Se define el **máximo común divisor** $\gcd(a, b)$ de a y b como el mayor número d tal que d divide a a y a b al mismo tiempo (es decir $d \mid a$ y $d \mid b$).

Test de primalidad

Para $n \geq 2$ se definen los siguientes conjuntos:

$$\begin{aligned} \mathbb{Z}_n^* &= \{a \in \{1, \ldots, n-1\} \mid \gcd(a, n) = 1\} \\ S_n^+ &= \{a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n}\} \\ S_n^- &= \{a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv -1 \pmod{n}\} \\ S_n &= S_n^+ \cup S_n^- \end{aligned}$$

Proposición 1. Si $n \geq 3$ es primo, entonces $S_n = \mathbb{Z}_n^*$.

Proposición 2. Si $n \geq 3$ es primo:

$$|S_n^+| \ = \ |S_n^-| \ = \ \frac{n-1}{2}.$$

Proposición 3. Sea $n = n_1 \cdot n_2$, donde $n_1, n_2 \geq 3$ y $\gcd(n_1, n_2) = 1$. Si existe $a \in \mathbb{Z}_n^*$ tal que $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$, entonces:

$$|S_n| \ \leq \ \frac{1}{2} \cdot |\mathbb{Z}_n^*|$$

Algoritmo EsPotencia

Dado un número natural $n \geq 2$, la siguiente función verifica si existen $m, k \in \mathbb{N}$ tales que $k \geq 2$ y $n = m^k$

```
EsPotencia(n)
    if n ≤ 3 then return no
    else
        for k := 2 to ⌊log_2(n)⌋ do
            if TieneRaízEntera(n, k, 1, n) then
                return sí
        return no
```

La siguiente función verifica si existe $m \in \{i, \ldots, j\}$ tal que $n = m^k$

```
TieneRaízEntera(n, k, i, j)
    if i = j then
        if EXP(i, k) = n then return sí
        else return no
    else if i < j then
        p := ⌊(i+j)/2⌋
        val := EXP(p, k)
        if val = n then
            return sí
        else if val < n then
            return TieneRaízEntera(n, k, p + 1, j)
        return TieneRaízEntera(n, k, i, p − 1)
    return no
```

Caso promedio

Dado un algoritmo $\mathcal{A}$, para cada $n \in \mathbb{N}$ usamos una variable aleatoria X_n tal que para cada $w \in \Sigma^n$ se tiene que

$$X_n(w) := \mathit{tiempo}_{\mathcal{A}}(w)$$

Decimos que $\mathcal{A}$ en el caso promedio es $O(f(n))$ si $E(X_n) \in O(f(n))$

Quicksort

Dada una lista L el algoritmo **Quicksort** se define:

```
Quicksort(L, m, n)
    if m < n then
        ℓ := Partición(L, m, n)
        Quicksort(L, m, ℓ − 1)
        Quicksort(L, ℓ + 1, n)
```

```
Partición(L, m, n)
    pivote := L[m]
    i := m
    for j := m + 1 to n do
        if L[j] ≤ pivote then
            i := i + 1
            intercambiar L[i] con L[j]
    intercambiar L[m] con L[i]
    return i
```