Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC1253 - Matemáticas Discretas

Tarea 5

04 de noviembre de 2024

2º semestre 2024 - Profesores P. Bahamondes - D. Bustamante - M. Romero

---

Requisitos

- La tarea es individual. Los casos de copia serán sancionados con la reprobación del curso con nota 1,1.
- **Entrega**: Hasta las 23:59 del 13 de noviembre a través del buzón habilitado en el sitio del curso (Canvas).
  - Esta tarea debe ser hecha completamente en LaTeX. Tareas hechas a mano o en otro procesador de texto **no serán corregidas**.
  - Debe usar el template LaTeX publicado en la página del curso.
  - Cada solución de cada problema debe comenzar en una nueva hoja. ***Hint:*** Utilice `\newpage`
  - Los archivos que debe entregar son el archivo `PDF` correspondiente a su solución con nombre `numalumno.pdf`, junto con un `zip` con nombre `numalumno.zip`, conteniendo el archivo `numalumno.tex` que compila su tarea. Si su código hace referencia a otros archivos, debe incluirlos también.
- El no cumplimiento de alguna de las reglas se penalizará con un descuento de 0.5 en la nota final (acumulables).
- No se aceptarán tareas atrasadas (salvo que utilice su cupón `#problemaexcepcional`).
- Si tiene alguna duda, el foro de Github (issues) es el lugar oficial para realizarla.

Pregunta 1

(a) (1.0 pts) Sean $f, g : \mathbb{N} \to \mathbb{R}^+$ funciones arbitrarias. Demuestre que $f+g \in O(\text{máx}\{f, g\})$.

(b) (2.0 pts) Considere la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1 & n \geq 2 \end{cases}$$

Demuestre usando inducción que $T(n) \in O(n)$.

(c) (3.0 pts) Considere la recurrencia vista en clases para MergeSort:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & n \geq 2 \end{cases}$$

Demuestre usando inducción que $T(n) \in O(n \log n)$.

Hint: Para la parte (b) y (c), las siguientes propiedades pueden ser útiles:

$$n = \left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil \qquad \left\lfloor \frac{n}{2} \right\rfloor \leq \frac{n}{2} \qquad \left\lceil \frac{n}{2} \right\rceil \leq \frac{n+1}{2}$$

Para la parte (b), se recomienda probar algo más fuerte[1] que $T(n) \leq c \cdot n$. Por ejemplo, algo de la forma $T(n) \leq g(n)$ donde $g(n) < c \cdot n$ y g es una función que usted debe escoger. Lo mismo para la parte (c).

Solución

(a) Demostraremos que para todo $n \geq 0$, se tiene que $f(n) + g(n) \leq 2 \cdot \text{máx}\{f(n), g(n)\}$. Esto demuestra que $f + g \in O(\text{máx}\{f, g\})$ donde las constantes en la definición de O serían $n_0 = 0$ y $c = 2$.

Sea $n \geq 0$. Tenemos que $f(n) \leq \text{máx}\{f(n), g(n)\}$ y $g(n) \leq \text{máx}\{f(n), g(n)\}$. Luego,

$$f(n) + g(n) \leq \text{máx}\{f(n), g(n)\} + \text{máx}\{f(n), g(n)\} = 2 \cdot \text{máx}\{f(n), g(n)\}.$$

(b) Demostraremos por inducción fuerte que para todo $n \geq 1$, se tiene que $T(n) \leq 2n - 1$. Notar que esto implica que para todo $n \geq 1$, se cumple $T(n) \leq 2n$, y luego esto demuestra que $T \in O(n)$ donde las constantes en la definición de O serían $n_0 = 1$ y $c = 2$.

[1] Puede parecer contraintuitivo que probar por inducción algo más fuerte nos ayude. Esto es una estrategia bastante común. La ventaja es que al probar algo más fuerte por inducción, la hipótesis inductiva también se hace más fuerte, y luego la demostración de la tesis inductiva puede resultar más simple.

CB: Para $n = 1$ tenemos que $T(1) = 1 \leq 1 = 2 \cdot 1 - 1$.

HI: Sea $n \geq 2$. Para todo $k \in \{1, \ldots, n-1\}$, se cumple que $T(k) \leq 2k - 1$.

TI: Hay que demostrar que $T(n) \leq 2n - 1$. Tenemos que:

$$\begin{aligned}
T(n) &= T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + 1 \\
&\leq (2 \left\lfloor \frac{n}{2} \right\rfloor - 1) + (2 \left\lceil \frac{n}{2} \right\rceil - 1) + 1 \qquad \text{(HI)} \\
&= 2(\left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil) - 1 \qquad \text{(reordenando y factorizando por 2)} \\
&= 2n - 1 \qquad (n = \left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil)
\end{aligned}$$

(c) Demostraremos por inducción fuerte que para todo $n \geq 3$, se cumple $T(n) \leq 3n \log(n-1)$. Notar que esto implica que para todo $n \geq 3$, se cumple $T(n) \leq 3n \log(n)$ (ya que la función log es no decreciente), y luego esto demuestra que $T \in O(n \log n)$ donde las constantes en la definición de O serían $n_0 = 3$ y $c = 3$.

CB: Los casos bases son $n \in \{3, 4, 5\}$, ya que en esos casos la recurrencia depende de $T(1)$ y $T(2)$.

- Para $n = 3$, tenemos $T(3) = T(1) + T(2) + 3 = 1 + 4 + 3 = 8 \leq 3 \cdot 3 \cdot \log(2) = 9$.
- Para $n = 4$, tenemos $T(4) = T(2) + T(2) + 4 = 4 + 4 + 4 = 12 \leq 3 \cdot 4 \cdot \log(3) \approx 19,019$.
- Para $n = 5$, tenemos $T(5) = T(2) + T(3) + 5 = 4 + 8 + 5 = 17 \leq 3 \cdot 5 \cdot \log(4) = 30$.

HI: Sea $n \geq 6$. Para todo $k \in \{3, \ldots, n-1\}$, entonces $T(k) \leq 3k \log(k-1)$.

TI: Hay que demostrar que $T(n) \leq 3n\log(n-1)$. Tenemos que:

$$
\begin{aligned}
T(n) &= T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + n \\
&\leq 3\left\lfloor \frac{n}{2} \right\rfloor \log\left(\left\lfloor \frac{n}{2} \right\rfloor - 1\right) + 3\left\lceil \frac{n}{2} \right\rceil \log\left(\left\lceil \frac{n}{2} \right\rceil - 1\right) + n && \text{(HI)} \\
&\leq 3\left\lfloor \frac{n}{2} \right\rfloor \log\left(\left\lceil \frac{n}{2} \right\rceil - 1\right) + 3\left\lceil \frac{n}{2} \right\rceil \log\left(\left\lceil \frac{n}{2} \right\rceil - 1\right) + n && (\left\lfloor \frac{n}{2} \right\rfloor \leq \left\lceil \frac{n}{2} \right\rceil \text{ y log es no decreciente}) \\
&= 3n\log\left(\left\lceil \frac{n}{2} \right\rceil - 1\right) + n && (\text{factorizamos y } n = \left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil) \\
&\leq 3n\log\left(\frac{n+1}{2} - 1\right) + n && (\left\lceil \frac{n}{2} \right\rceil \leq \frac{n+1}{2} \text{ y log no decreciente}) \\
&= 3n\log\left(\frac{n-1}{2}\right) + n \\
&= 3n(\log(n-1) - \log(2)) + n \\
&= 3n\log(n-1) - 2n \\
&\leq 3n\log(n-1)
\end{aligned}
$$

Pauta (6 pts.)

(a) 1.0 pts, 0.5 por constantes que funcionen y 0.5 por la demostración. Descuentos y puntajes parciales a criterio del corrector.

(b) 2.0 pts, 0.5 por constantes que funcionen, 0.5 por CB+HI y 1.0 por la TI. Descuentos y puntajes parciales a criterio del corrector.

(c) 3.0 pts, 0.5 por constantes que funcionen, 1.0 por CB, 0.5 por HI y 1.0 por TI. Descuentos y puntajes parciales a criterio del corrector.

Pregunta 2

(a) (2.0 pts) Considere la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n-1) + n & n \geq 2 \end{cases}$$

Encuentre una notación Θ para $T(n)$.

(b) (2.0 pts) Considere el siguiente algoritmo para ordenar una lista:

input : Arreglo $A[0, \ldots, n-1]$, largo n
output: Arreglo ordenado

`QuickSort`(A, n)**:**

```
1   if n ≤ 1 then
2       return A
3   else
4       q ← A[0]
5       A_1, A_2 ← Particionar(A[1, ..., n − 1], q)
6       if A_1 ≠ ∅ then
7           B_1 ← QuickSort(A_1, largo(A_1))
8       if A_2 ≠ ∅ then
9           B_2 ← QuickSort(A_2, largo(A_2))
10      return [B_1, q, B_2]
```

En el algoritmo, la función *largo* retorna el largo de una lista, y la función *Particionar* se define como sigue:

input : Arreglo A, natural q
output: Arreglos A_1, A_2

`Particionar`(A, q)**:**

```
1   A_1 ← ∅
2   A_2 ← ∅
3   for i ∈ {0, ..., largo(A) − 1} do
4       if A[i] ≤ q then
5           A_1 ← [A_1, A[i]]
6       else
7           A_2 ← [A_2, A[i]]
8   return A_1, A_2
```

Encuentre una notación Θ para la complejidad de peor caso $T(n)$ de QuickSort.

(c) Encuentre una notación Θ para la complejidad de mejor caso $T(n)$ de QuickSort.

Hint: Al analizar QuickSort puede enfocarse en la comparación de la línea 1 de QuickSort, y en la comparación de la línea 4 de Particionar. Puede asumir que el peor caso para QuickSort es cuando la partición que entrega *Particionar* es lo más "desbalanceada" posible, y el mejor caso es cuando es lo más "balanceada" posible.

Solución

(a) Podemos desenrollar la recurrencia y obtener una expresión explícita para $T(n)$:

$$\begin{aligned}
T(n) &= T(n-1) + n \\
&= T(n-2) + (n-1) + n \\
&= T(n-3) + (n-2) + (n-1) + n \\
&\vdots \\
&= T(1) + 2 + 3 + \cdots + n \\
&= 1 + 2 + 3 + \cdots + n \\
&= \sum_{i=1}^{n} i \\
&= \frac{n(n+1)}{2} \\
&= \frac{n^2}{2} + \frac{n}{2}
\end{aligned}$$

Sabemos de clases que $\frac{n^2}{2} + \frac{n}{2} \in \Theta(n^2)$.

(b) Siguiendo el hint, el peor caso es cuando la subrutina *Particionar* entrega la partición más desbalanceada, es decir, $|A_1| = n-1$ y $|A_2| = 0$. Acá estamos en el caso en que q es mayor o igual que todos los elementos de $A[1, \ldots, n-1]$. (Como peor caso también se podría tomar $|A_1| = 0$ y $|A_2| = n-1$, el análisis es el mismo.)

Calculemos la complejidad de peor caso $T(n)$ de Quicksort. Si $n = 1$, sólo hacemos la comparación de la línea 1. En el peor caso, si $n \geq 2$, hacemos la comparación de la línea 1, las comparaciones de la subrutina *Particionar*, las cuales son $n-1$, y la peor cantidad de comparaciones para una instancia de tamaño $n-1$. Notar que sólo hacemos la llamada recursiva $B_1 \leftarrow \text{QuickSort}(A_1, \text{largo}(A_1))$. Si llevamos esto a una recurrencia obtenemos:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n-1) + n & n \geq 2 \end{cases}$$

Utilizando la parte (a), concluimos que $T(n) \in \Theta(n^2)$.

(c) Siguiendo el hint, el mejor caso es cuando la subrutina *Particionar* entrega la partición más balanceada, es decir, $|A_1| = \left\lfloor \frac{n-1}{2} \right\rfloor$ y $|A_2| = \left\lceil \frac{n-1}{2} \right\rceil$. (Como mejor caso también se podría tomar $|A_1| = \left\lceil \frac{n-1}{2} \right\rceil$ y $|A_2| = \left\lfloor \frac{n-1}{2} \right\rfloor$, el análisis es el mismo.)

Calculemos la complejidad de mejor caso $T(n)$ de Quicksort. Si $n \leq 1$, sólo hacemos la comparación de la línea 1. En el mejor caso, si $n \geq 2$, hacemos la comparación de la línea 1, las comparaciones de la subrutina *Particionar*, las cuales son $n-1$, la mejor cantidad de comparaciones para una instancia de tamaño $\lfloor \frac{n-1}{2} \rfloor$, y la mejor cantidad de comparaciones para una instancia de tamaño $\lceil \frac{n-1}{2} \rceil$. Si llevamos esto a una recurrencia obtenemos:

$$T(n) = \begin{cases} 1 & n \leq 1 \\ T(\lfloor \frac{n-1}{2} \rfloor) + T(\lceil \frac{n-1}{2} \rceil) + n & n \geq 2 \end{cases}$$

Notar que en la parte (b) ignoramos el caso $n = 0$ ya que podíamos comenzar de $n = 1$ y la recurrencia quedaba bien definida. En este caso no podemos hacer lo mismo ya que $T(2)$ depende de $T(0)$. Observe que esta recurrencia es muy similar a la de MergeSort, salvo que ahora aparece $\lfloor \frac{n-1}{2} \rfloor$ y $\lceil \frac{n-1}{2} \rceil$, en vez de $\lfloor \frac{n}{2} \rfloor$ y $\lceil \frac{n}{2} \rceil$. Intuitivamente, esperaríamos entonces que $T(n) \in \Theta(n \log(n))$.

Utilizando las mismas ideas de la pregunta 1 parte (c), podemos demostrar por inducción que $T(n) \in \Theta(n \log(n))$.

Veamos primero que $T(n) \in O(n \log(n))$. Demostraremos que para todo $n \geq 2$, se cumple $T(n) \leq 2n \log(n)$.

CB: Los casos bases son $n \in \{2, 3\}$, ya que en esos casos la recurrencia depende de $T(0)$ y $T(1)$.

- Para $n = 2$, tenemos $T(2) = T(0) + T(1) + 2 = 1 + 1 + 2 = 4 \leq 2 \cdot 2 \cdot \log(2) = 4$.
- Para $n = 3$, tenemos $T(3) = T(1) + T(2) + 3 = 1 + 4 + 3 = 8 \leq 2 \cdot 3 \cdot \log(3) \approx 9,509$.

HI: Sea $n \geq 4$. Para todo $k \in \{2, \ldots, n-1\}$, entonces $T(k) \leq 2k \log(k)$.

TI: Hay que demostrar que $T(n) \leq 2n\log(n)$. Tenemos que:

$$
\begin{aligned}
T(n) &= T(\left\lfloor \frac{n-1}{2} \right\rfloor) + T(\left\lceil \frac{n-1}{2} \right\rceil) + n \\
&\leq 2\left\lfloor \frac{n-1}{2} \right\rfloor \log\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right) + 2\left\lceil \frac{n-1}{2} \right\rceil \log\left(\left\lceil \frac{n-1}{2} \right\rceil\right) + n && \text{(HI)} \\
&\leq 2\left\lfloor \frac{n-1}{2} \right\rfloor \log\left(\left\lceil \frac{n-1}{2} \right\rceil\right) + 2\left\lceil \frac{n-1}{2} \right\rceil \log\left(\left\lceil \frac{n-1}{2} \right\rceil\right) + n && (\left\lfloor \frac{n-1}{2} \right\rfloor \leq \left\lceil \frac{n-1}{2} \right\rceil) \\
&\leq 2\left\lfloor \frac{n-1}{2} \right\rfloor \log\left(\frac{n}{2}\right) + 2\left\lceil \frac{n-1}{2} \right\rceil \log\left(\frac{n}{2}\right) + n && (\left\lceil \frac{n-1}{2} \right\rceil \leq \frac{(n-1)+1}{2} = \frac{n}{2}) \\
&= 2(n-1)\log\left(\frac{n}{2}\right) + n \\
&\leq 2n\log\left(\frac{n}{2}\right) + n \\
&= 2n(\log(n) - 1) + n \\
&= 2n\log(n) - n \\
&\leq 2n\log(n)
\end{aligned}
$$

Veamos ahora que $T(n) \in \Omega(n\log(n))$. Demostraremos que para todo $n \geq 0$, se cumple $T(n) \geq \frac{1}{4}(n+1)\log(n+2)$. Notar que esto implica que para todo $n \geq 0$, se cumple $T(n) \geq \frac{1}{4}n\log(n)$ (ya que la función log es no decreciente), y luego esto demuestra que $T \in \Omega(n\log(n))$ donde las constantes en la definición de Ω serían $n_0 = 0$ y $c = \frac{1}{4}$.

CB: Los casos bases son $n \in \{0, 1\}$.

- Para $n = 0$, tenemos $T(0) = 1 \geq \frac{1}{4}(0+1)\log(0+2) = \frac{1}{4}$.
- Para $n = 1$, tenemos $T(1) = 1 \geq \frac{1}{4}(1+1)\log(1+2) \approx 0{,}792$.

HI: Sea $n \geq 2$. Para todo $k \in \{0, \ldots, n-1\}$, entonces $T(k) \geq \frac{1}{4}(k+1)\log(k+2)$.

TI: Hay que demostrar que $T(n) \geq \frac{1}{4}(n+1)\log(n+2)$. Tenemos que:

$$\begin{aligned}
T(n) &= T(\left\lfloor \frac{n-1}{2} \right\rfloor) + T(\left\lceil \frac{n-1}{2} \right\rceil) + n \\
&\geq \frac{1}{4}\left(\left\lfloor \frac{n-1}{2} \right\rfloor + 1\right)\log\left(\left\lfloor \frac{n-1}{2} \right\rfloor + 2\right) + \frac{1}{4}\left(\left\lceil \frac{n-1}{2} \right\rceil + 1\right)\log\left(\left\lceil \frac{n-1}{2} \right\rceil + 2\right) + n \quad \text{(HI)} \\
&\geq \frac{1}{4}\left(\left\lfloor \frac{n-1}{2} \right\rfloor + \left\lceil \frac{n-1}{2} \right\rceil + 2\right)\log\left(\left\lfloor \frac{n-1}{2} \right\rfloor + 2\right) + n \quad \left(\left\lceil \frac{n-1}{2} \right\rceil \geq \left\lfloor \frac{n-1}{2} \right\rfloor \text{ y factorizando}\right) \\
&= \frac{1}{4}(n+1)\log\left(\left\lfloor \frac{n-1}{2} \right\rfloor + 2\right) + n \\
&\geq \frac{1}{4}(n+1)\log\left(\left\lfloor \frac{n-1}{2} \right\rfloor + 2\right) + n \qquad \left(\left\lfloor \frac{n-1}{2} \right\rfloor \geq \frac{(n-1)-1}{2} = \frac{n-2}{2}\right) \\
&= \frac{1}{4}(n+1)\log\left(\frac{n+2}{2}\right) + n \\
&= \frac{1}{4}(n+1)\log(n+2) + n - \frac{1}{4}(n+1)
\end{aligned}$$

Notar que $n - \frac{1}{4}(n+1) \geq 0$ siempre se cumple para $n \geq 2$, ya que

$$n - \frac{1}{4}(n+1) \geq 0 \iff \frac{3}{4}n \geq \frac{1}{4} \iff n \geq \frac{1}{3}.$$

Concluimos que:

$$T(n) \geq \frac{1}{4}(n+1)\log(n+2) + n - \frac{1}{4}(n+1) \geq \frac{1}{4}(n+1)\log(n+2)$$

Distribución de puntajes:

(a) 1.0 pts por decir que es $\Theta(n^2)$, 1.0 pts por dar un argumento correcto de esto.

(b) 1.0 pts por dar un análisis correcto junto a la recurrencia. 1.0 pts por dar y argumentar correctamente el orden $\Theta(n^2)$.

(c) 1.0 pts por dar un análisis correcto junto a la recurrencia. 1.0 pts por dar y argumentar correctamente el orden $\Theta(n\log(n))$.