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 2

Notación asintótica y ecuaciones de recurrencia

1. Demuestre las siguientes afirmaciones:

   - $n! \in \Omega(2^n)$.
     Es claro que:

$$\begin{aligned} n! &= 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot \ ... \ \cdot n \\ &= 1 \cdot 2 \cdot 3 \cdot (2 \cdot 2) \cdot 5 \cdot \ ... \ \cdot n \\ &= 2 \cdot 2 \cdot 3 \cdot 2 \cdot 5 \cdot \ ... \ \cdot n \\ &> 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot \ ... \ \cdot 2 \\ &= 2^n \end{aligned}$$

     Luego, considerando $c = 1$ y $n = 4$, tendremos que $n! \in \Omega(2^n)$.

   - $n^n \notin \mathcal{O}(n!)$.
     Podemos demostrar esta afirmación utilizando límites. Es fácil ver que:

$$0 < \frac{n!}{n^n}$$

     Además:

$$\begin{aligned} \frac{n!}{n^n} &= \frac{n \cdot (n-1) \cdot ... \cdot 1}{n \cdot \quad n \quad \cdot ... \cdot n} \\ &= 1 \cdot \frac{n-1}{n} \cdot ... \cdot \frac{1}{n} \\ &\leq 1 \cdot 1 \cdot ... \cdot \frac{1}{n} \\ &= \frac{1}{n} \end{aligned}$$

     Por otra parte, es claro que:

$$\lim_{n\to\infty} 0 = 0 \ \wedge \ \lim_{n\to\infty} \frac{1}{n} = 0$$

$$\Rightarrow \lim_{n\to\infty} \frac{n!}{n^n} = 0$$

     Por lo tanto, $n! \in \mathcal{O}(n^n)$ y $n^n \notin \mathcal{O}(n!)$

- $\log(n^n) \in \mathcal{O}(\log(n!))$. Podemos desarrollar $\log(n!)$ como:

$$\begin{aligned}
\log(n!) &= \log(n) + \log(n-1) + ... + \log(n/2) + \log(n/2-1) + ... + \log(2) + \log(1) \\
&\geq \log(n/2) + \log(n/2) + ... + \log(n/2) + \log(n/2-1) + ... + \log(2) + \log(1) \\
&\geq \frac{n}{2} \cdot \log(n/2) + \log(n/2-1) + ... + \log(4) + \log(3) + \log(2) + \log(1) \\
&= \frac{n}{2} \cdot \log(n/2) + \log(n/2-1) + ... + \log(2^2) + \log(3) + \log(2) + 0 \\
&\geq \frac{n}{2} \cdot \log(n/2) + \log(n/2-1) + ... + \log(2) + \log(3) + \log(2) + \log(2) \\
&= \frac{n}{2} \cdot \log(n/2) + \frac{n}{2}\log(2) \\
&= \frac{n}{2} \cdot (\log(n) - \log(2) + \log(2)) \\
&= \frac{n}{2} \cdot \log(n)
\end{aligned}$$

  Por tanto, si consideramos $c = 1/2$ y $n_0 = 8$, tenemos que $\log(n^n) \in \mathcal{O}(\log(n!))$

2. El tiempo que demora el algoritmo de `MergeSort` se puede expresar con la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2 \cdot T(\frac{n}{2}) + f(n) & n > 1 \end{cases}$$

   Donde $f(n)$ es el tiempo utilizado por la subrutina `merge`. Asumiendo que $n = 2^k$, resuelva la recurrencia para los siguientes casos:

   (a) El `merge` se implementó como es usual, $f(n) = C \cdot n$, con C una constante tal que $C > 0$ Podemos expandir la recurrencia, encontrando:

$$\begin{aligned}
T(2^k) &= 2T(2^{k-1}) + C2^k \\
&= 2(2T(2^{k-2}) + C2^{k-1})) + C2^k \\
&= 2^2T(2^{k-2}) + C2^k + C2^k \\
&= 2^2(2T(2^{k-3}) + C2^{k-2}) + C2^k + C2^k \\
&= 2^3T(2^{k-3}) + C2^k + C2^k + C2^k \\
&= ... \\
&= 2^iT(2^{k-i}) + i \cdot C2^k \\
&= ... \\
&= 2^kT(1) + k \cdot C2^k \\
&= n + n \cdot \log_2(n) \cdot C
\end{aligned}$$

$$\therefore T(n) \in \mathcal{O}(n \cdot log(n))$$

(b) El `merge` se implementó de forma ineficiente, resultando en $f(n) = C \cdot n^2$, con C una constante tal que $C > 0$. Similarmente, la recurrencia puede ser expandida como:

$$
\begin{aligned}
T(2^k) &= 2T(2^{k-1}) + C(2^k)^2 \\
&= 2(2T(2^{k-2}) + C(2^{k-1})^2) + C(2^k)^2 \\
&= 2^2T(2^{k-2}) + C\left(2(2^{k-1})^2 + (2^k)^2\right) \\
&= 2^2\left(2T(2^{k-3}) + C(2^{k-2})^2\right) + C\left(2(2^{k-1})^2 + (2^k)^2\right) \\
&= 2^3T(2^{k-3}) + C\left(2^2(2^{k-2})^2 + 2(2^{k-1})^2 + (2^k)^2\right) \\
&= \ldots \\
&= 2^iT(2^{k-i}) + C\sum_{j=0}^{i-1} 2^j(2^{k-j})^2 \\
&= \ldots \\
&= 2^kT(1) + C\sum_{j=0}^{k-1} 2^j(2^{k-j})^2 \\
&= 2^k + C2^{2k}\sum_{j=0}^{k-1} \frac{1}{2^j} \\
&= n + n^2 \cdot C \cdot \frac{1}{2^j}
\end{aligned}
$$

$$\therefore T(n) \in \mathcal{O}(n^2)$$

(c) El `merge` se implementó de forma innovadora (prácticamente mágica) y funciona en tiempo $f(n) = C$, con C una constante tal que $C > 0$ De la misma forma que en las anteriores, encontramos:

$$
\begin{aligned}
T(2^k) &= 2T(2^{k-1}) + C \\
&= 2(2T(2^{k-2}) + C) + C \\
&= 2^2T(2^{k-2}) + 2C + C \\
&= 2^2(2T(2^{k-3}) + C) + 2C + C \\
&= 2^3T(2^{k-3}) + C(2^2 + 2^1 + 2^0) \\
&= \ldots \\
&= 2^iT(2^{k-i}) + C\sum_{j=0}^{k-1} 2^j \\
&= \ldots \\
&= 2^kT(1) + C\sum_{j=0}^{k-1} 2^j \\
&= 2^k + C(2^k - 1) \\
&= n + C(n-1)
\end{aligned}
$$

$$\therefore T(n) \in \mathcal{O}(n)$$