PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC1253 — Matemáticas Discretas — 1' 2019

Ayudantía 10

Análisis de algoritmos e Inducción

Problema 1

Considere el siguiente código para un valor $a \geq 2$.

Function `quickAlgo` *(n)*
  $x := 1$
  if $a^n \geq n!$ **then**
    for $i = 1$ **to** n **do**
      for $j = 1$ **to** a^i **do**
        $x := x + 1$
  else
    while $n \geq 1$ **do**
      for $i = 1$ **to** n **do**
        $x := x + 1$
      $n := \left\lfloor \frac{n}{2} \right\rfloor$
  return x

Encuentre una función f y demuestre que el tiempo de `quickAlgo` en términos de n es $\Theta(f(n))$.

Solución propuesta.

Debido a que se nos pregunta por el comportamiento asintótico de la función tiempo del algoritmo $T(n)$, podemos hacer el análisis a partir de cierto n_0. De esta forma, nos aprovecharemos del hecho de que el bloque **if** del algoritmo solo se ejecuta cuando $a^n \geq n!$.

Debido a que $a^n \in \mathcal{O}(n!)$ para todo $a \geq 2$, sabemos que la ecuación $a^n = n!$ tiene alguna solución no trivial en un $n_0 > 0$. Luego, sabemos que para $n \in \{0, 1, \ldots, n_0\}$, el algoritmo ejecuta el bloque **if** y para $n \geq n_0 + 1$ ejecuta el bloque **else**. Como el comportamiento asintótico se puede visualizar para los $n \geq n_0 + 1$, entonces solo analizaremos el bloque **else**.

Sea ℓ el número de iteraciones del bloque **while** que se ejecutan, y sea f_i la cantidad de iteraciones del bloque **for** que se ejecutan, para la iteración i-ésima del **while**. El tiempo del algoritmo completo puede tomarse entonces como

$$T(n) = f_0 + f_1 + \ldots + f_\ell = \sum_{i=0}^{\ell} f_i$$

donde ℓ depende de n. En efecto, sabemos que el **while** tiene su última ejecución cuando *la variable* n tiene valor 1. Esto ocurre con la siguiente secuencia de divisiones:

$$n, \frac{n}{2}, \frac{n}{4}, \ldots, 1$$

que puede reescribirse en términos de la iteración i según

$$\frac{n}{2^0}, \frac{n}{2^1}, \frac{n}{2^2}, \ldots, \frac{n}{2^{\log_2(n)}}$$

de manera que deducimos que el número de iteraciones del **while** es $\ell = \log_2(n)$.

Con lo anterior,

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_2(n)} n \left(\frac{1}{2}\right)^i \\ &\leq \sum_{i=0}^{\infty} n \left(\frac{1}{2}\right)^i \\ &= n \cdot \frac{1}{1 - 1/2} \\ &= 2n \end{aligned}$$

Con este resultado, proponemos usar $f(n) = n$ como la función pedida. Comprobamos que efectivamente, $T(n) \in \Theta(n)$:

- $T(n) \in \mathcal{O}(n)$ pues como probamos, $T(n) \geq 2n$ para todo $n \geq n_0 + 1$, donde $c' = 2$ y $n_0' = n_0 + 1$ son las constantes que atestiguan la definición de $\mathcal{O}$.
- $n \in \mathcal{O}(T(n))$ pues basta observar que n es uno de los sumandos que definen $T(n)$:

  $$n \leq n + \frac{n}{2} + \ldots + 1 = 1 \cdot T(n)$$

  por lo cual, tomar $c' = 1$ y $n_0' = n_0 + 1$ demuestra lo pedido.

Problema 2

Considere el siguiente algoritmo para calcular el n-ésimo número de la secuencia de Fibonacci.

Function `Fibonacci` (n)
    $x := 1$
    if $n = 0$ **then**
        return 0
    else if $n = 1$ **then**
        return 1
    else
        return `Fibonacci` $(n-1)$ + `Fibonacci` $(n-2)$

Demuestre que el algoritmo es correcto, es decir, que para cualquier natural n, Fibonacci(n) es el n-ésimo número en la sucesión de Fibonacci.

Solución propuesta.

Sea $\{F_n\}_{n\geq 0}$ la sucesión de Fibonacci, es decir, la sucesión definida por

$$F_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F_{n-1} + F_{n-2} & n \geq 2 \end{cases}$$

Definimos la proposición

$$P(n) := \text{Para } n \in \mathbb{N} \text{ se cumple } F_n = \texttt{Fibonacci}(n)$$

y demostraremos lo pedido usando inducción fuerte. Notamos que la proposición se cumple para 0 y 1 pues

- $P(0)$: Al ejecutar $\texttt{Fibonacci}(0)$, el algoritmo entra al primer condicional y retorna 0, que coincide con F_0.
- $P(1)$: Al ejecutar $\texttt{Fibonacci}(1)$, el algoritmo entra al segundo condicional y retorna 1, que coincide con F_1.

Suponemos como hipótesis inductiva que $P(k)$ es verdadero para todo $n > 2$ y $k < n$. Consideremos entonces el paso inductivo $P(n)$. Por construcción del algoritmo, al ejecutar $\texttt{Fibonacci}(\texttt{n})$ se retorna

$$\texttt{Fibonacci}(n) = \texttt{Fibonacci}(n-1) + \texttt{Fibonacci}(n-2)$$

Por hipótesis inductiva, se tiene que

$$\texttt{Fibonacci}(n) = \texttt{Fibonacci}(n-1) + \texttt{Fibonacci}(n-2) = F_{n-1} + F_{n-2}$$

Luego, con la definición de la sucesión, se tiene que

$$\texttt{Fibonacci}(n) = F_n$$

de manera que se verifica $P(n)$. Con esto, $P(n)$ es verdadero para todo natural y por lo tanto el algoritmo es correcto.

Problema 3

Decimos que $I \subseteq \mathbb{R}$ es un intervalo abierto en $\mathbb{R}$, si existe $a, b \in \mathbb{R}$ con $a < b$ tal que $I = \{x \mid a < x < b\}$. Sean $I_1, \ldots, I_n$ intervalos abiertos en $\mathbb{R}$ tal que $I_i \cap I_j \neq \emptyset$ para todo $1 \leq i \leq n$ y $1 \leq j \leq n$. Demuestre usando inducción que $I_1 \cap I_2 \cap \ldots \cap I_n \neq \emptyset$.

Solución propuesta.

Definimos la proposición

$$P(n) := \text{Si } I_1, \ldots, I_n \text{ son abiertos tales que } I_i \cap I_j \neq \emptyset \text{ para todo } i, j \leq n, \text{ entonces } I_1 \cap \ldots \cap I_n \neq \emptyset$$

Demostraremos la proposición usando inducción simple.

- **Caso base:** Tomando $P(2)$, tenemos como supuesto que I_1, I_2 es una secuencia de abiertos tales que $I_1 \cap I_2 \neq \emptyset$. Esto es precisamente lo que se quiere concluir, por lo cual $P(2)$ es verdadera.
- **Hipótesis inductiva:** Suponemos que se cumple $P(n)$.
- **Paso inductivo:** Nos interesa probar $P(n+1)$ usando $P(n)$. Sean $I_1, \ldots, I_{n+1}$ conjuntos abiertos tales que $I_i \cap I_j \neq \emptyset$ para todo $i, j \leq n+1$. Definimos el conjunto $I^* = I_n \cap I_{n+1}$ que por el supuesto anterior es tal que $I^* \neq \emptyset$. Luego, para poder usar la hipótesis inductiva sobre $I_1, \ldots, I_{n-1}, I^*$ tenemos que demostrar primero que $I_i \cap I^* \neq \emptyset$, para todo $i \leq n-1$.

  ado que dos abiertos tienen intersección no vacía si y solo si tienen elementos en común, si denotamos cada abierto por $I_i = \{x \mid a_i < x < b_i\}$, entonces para cualquier par de abiertos,

  $$I_i \cap I_j \neq \emptyset \Leftrightarrow \text{máx}\{a_i, a_j\} < \text{mín}\{b_i, b_j\}$$

  Definimos $Q(n) :=$ Dados $I_1, \ldots, I_n$ abiertos de la forma $I_i = \{x \mid a_i < x < b_i\}$, su intersección es no vacía si y solo si $\text{máx}\{a_i, \ldots, a_n\} < \text{mín}\{b_1, \ldots, b_n\}$. Demostraremos esta propiedad por inducción:

- **Caso base:** $P(2)$ se cumple trivialmente como se había mostrado.
- **Hipótesis inductiva:** Suponemos se cumple $P(n)$.
- **Paso inductivo:** Interesa tomar $I_1, \ldots, I_{n+1}$. Definimos $I^* = I_n \cap I_{n+1} = \{x \mid a_* < x < b_*\}$, donde $a_* = \text{máx}\{a_n, a_{n+1}\}$ y $b_* = \text{mín}\{b_n, b_{n+1}\}$. Luego, por H.I. se tiene que

$$I_1 \cap \ldots \cap I_{n-1} \cap I^* \neq \emptyset \Leftrightarrow \text{máx}\{a_i, \ldots, a_{n-1}, a_*\} < \text{mín}\{b_1, \ldots, b_{n-1}, b_*\}$$

  Notamos que

$$\begin{aligned} \text{máx}\{a_i, \ldots, a_{n-1}, a_*\} &= \text{máx}\{a_i, \ldots, a_{n-1}, \text{máx}\{a_n, a_{n+1}\}\} \\ &= \text{máx}\{a_i, \ldots, a_{n-1}, a_n, a_{n+1}\} \end{aligned}$$

  de manera que

$$I_1 \cap \ldots \cap I_{n-1} \cap I^* \neq \emptyset \Leftrightarrow \text{máx}\{a_i, \ldots, a_{n+1}\} < \text{mín}\{b_1, \ldots, b_{n+1}\}$$

  e incorporando la definición de I^*, obtenemos la conclusión de $P(n+1)$:

$$I_1 \cap \ldots \cap I_{n+1} \neq \emptyset \Leftrightarrow \text{máx}\{a_i, \ldots, a_{n+1}\} < \text{mín}\{b_1, \ldots, b_{n+1}\}$$

Usando esta propiedad demostramos que $I_i \cap I^* \neq \emptyset$. En efecto, como

$$I_i \cap I^* = I_i \cap \{x \mid \text{máx}\{a_n, a_{n+1}\} < x < \text{mín}\{b_n, b_{n+1}\}\}$$

tenemos que

$$I_i \cap I^* = \{x \mid \text{máx}\{a_i, a_n, a_{n+1}\} < x < \text{mín}\{b_i, b_n, b_{n+1}\}\}$$

Como las intersecciones $I_i \cap I_n$ y $I_i \cap I_{n+1}$ son no vacías, por la propiedad recién demostrada tenemos que $\text{máx}\{a_i, a_n, a_{n+1}\} < \text{mín}\{b_i, b_n, b_{n+1}\}$ y en consecuencia, $I_i \cap I^* \neq$.

Luego, por hipótesis inductiva

$$I_1, \ldots, I_{n-1} \cap I^* \neq \emptyset$$

e insertando la defininción de I^* obtenemos lo pedido:

$$I_1, \ldots, I_{n+1} \neq \emptyset$$

Con esto se demuestra que para todo natural $n \geq 2$, $P(n)$ es verdadera.

Problema 4

Demuestre usando inducción que si $a_1, a_2, \ldots, a_n$ es una secuencia de números distintos, entonces se necesitan exactamente $(n-1)$ multiplicaciones de a pares para calcular $a_1 \times a_2 \times \ldots \times a_n$, sin importar el orden o la forma como se agrupan las multiplicaciones.

Definimos la proposición $P(n)$ según

$$P(n) := \text{Para todo } a_1, \ldots, a_n \text{ distintos, } a_1 \times \ldots \times a_n \text{ se calcula con } n-1 \text{ multiplicaciones}$$

Demostraremos el resultado usando inducción fuerte. Notamos que para $P(2)$, la secuencia de números es de tamaño 2 y por lo tanto $a_1 \times a_2$ se calcula con una multiplicación, i.e. $n - 1 = 2 - 1 = 1$.

Suponemos que se cumple $P(k)$ para todo $k < n$, con $n > 2$. Luego, podemos descomponer la secuencia de números en dos grupos de multiplicaciones. Sean k, ℓ naturales tales que $n = \ell + k$. Luego, podemos descomponer

$$a_1 \times a_n = (b_1 \times \ldots \times b_k) \times (c_1 \times \ldots \times c_\ell)$$

donde cada b_i, c_j son elementos de la secuencia $a_1, \ldots, a_n$ distintos entre si y presentes en cualquier orden. Por hipótesis, como $k < n$ y $\ell < n$, entonces se cumplen $P(k)$ y $P(\ell)$, de forma que las secuencias de k y ℓ multiplicaciones se efectúan en $k-1$ y $\ell-1$ multiplicaciones respectivamente. Esto da lugar a

$$a_1 \times \ldots \times a_n = b \times c$$

en $k-1+\ell-1 = k+\ell-2 = n-2$ multiplicaciones, donde $b = b_1 \times \ldots \times b_k$ y $c = c_1 \times \ldots \times c_\ell$. Finalmente, efectuando una multiplicación se reduce $b \times c$ en cualquier orden, de manera que toda la secuencia se multiplica en $n-2+1 = n-1$ multiplicaciones, tal como se necesitaba probar.