Elementos de complejidad algorítmica

Relaciones de recurrencia

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

22 de agosto de 2024

Definición

Definición matemática

Definición (Relación de recurrencia)

Una *relación de recurrencia* para una **sucesión** $\{a_n\}_{n=0}^{\infty} = \{a_0, a_1, a_2, a_3, \ldots\}$ es una fórmula que expresa cada término a_n, a partir de cierto $n_0 \in \mathbb{N}$, en función de uno o más de los términos que le preceden.

- Los valores de los términos necesarios para empezar a calcular se llaman condiciones iniciales.
- Una **solución** de la relación de recurrencia es una sucesión cuyos términos (a partir de a_2) verifiquen la relación.
- **Resolver** una recurrencia (que liga los términos de una sucesión $\{a_n\}$) significa hallar una fórmula explícita *algebráica* en la que al sustituir n obtengamos al término a_n. Fernández Gallardo y Fernández Pérez 2018

Ejemplos básicos

Ejemplos:

- *Progresión aritmética:* $a_n = a_{n-1} + d$.
  Ejemplar:
  $$a_0 = 3 \qquad d = 7$$
  $$\{a_n\} = \{3, 10, 17, 24, 31, 38, 45, 52, 59, 66, 73\}$$
  Solución: $a_n = a_0 + dn$.
- *Progresión geométrica:* $a_n = ra_{n-1}$
  Ejemplar:
  $$a_0 = 3 \qquad r = 5$$
  $$\{a_n\} = \{3, 15, 75, 375, 1875, 9375, 46875, 234375\}$$
  Solucion: $a_n = a_0 r^n$

- *Sucesión de Fibonacci:*

$$
\begin{aligned}
a_0 &= a_1 = 1 \\
a_n &= a_{n-1} + a_{n-2} && \forall n \geqslant 2 \\
\{a_n\} &= \{1, 1, 2, 3, 5, 8, 13, 21, 34\}
\end{aligned}
$$

Solución:

$$a_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

- *Lineales homogéneas* de orden m

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_m a_{n-m}$$

Relaciones de recurrencia y complejidad

- Las relaciones de recurrencia se utilizan en forma natural para estimar la complejidad en tiempo de algoritmos recursivos.
- Para el cálculo, cada llamada recursiva se sustituye por el número de pasos que toma evaluarla.

```
public static long Factorial(long n)
{
  if (n < 0) throw new ArgumentException();   // O(1)
  if (n == 0 || n == 1) return 1;             // O(1)
  else return n * Factorial(n - 1);           // c + Tf(n-1)
}
```

$$
\begin{aligned}
T_{factorial}(n) &= \Theta(1) + Tf(n-1) \\
&= \mathcal{O}(n)
\end{aligned}
$$

¡Es la progresión aritmética!

Recursión doble

Ejemplo: Triángulo de Pascal

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1

$$T_p(n) = \Theta(1) + 2T_p(n-1)$$

```
public static long Pascal(long n, long m)
{
  if (n < 0 || m < 0 || m < n)
      throw new ArgumentException();
  if (n == 0 || n == m) return 1;
  else return Pascal(n - 1, m - 1) + Pascal(n - 1, m);
}
```

Orden de complejidad para el Triángulo de Pascal

$$
\begin{aligned}
T_p(n) &= \Theta(1) + 2T_p(n-1) \\
&= \Theta(1) + 2(\Theta(1) + 2T_p(n-2)) \\
&= (1+2)\Theta(1) + 2^2 T_p(n-2) \\
&= (1+2+2^2)\Theta(1) + 2^3 T_p(n-3) \\
\dots \\
&= \left(\frac{2^{n-1}-1}{2-1}\right)\Theta(1) + 2^n\Theta(1) \\
&= \mathcal{O}(2^n)
\end{aligned}
$$

Bibliografía

Bibliografía I

- Fernández Gallardo, Pablo y José Luis Fernández Pérez (2018). «El discreto encanto de la matemática». En: cap. Recurrencias. URL: http://verso.mat.uam.es/~pablo.fernandez/cap8-dic18.pdf.
- Preiss, Bruno (1999). *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. John Wiley & Sons.

Licencia