Capítulo 1. Análisis de algoritmos

Las técnicas de análisis de algoritmos permiten responder a algunas de las preguntas sobre eficiencia que un diseñador de algoritmos podría formularse, principalmente *¿cuánto tiempo va a tardar la ejecución de mi algoritmo?* y *¿cuánta memoria va a consumir mi algoritmo?*.

El análisis de algoritmos es de suma utilidad pues permite determinar si un algoritmo es adecuado para la resolución de un problema, por ejemplo un algoritmo que tome 10 años en resolver el problema podría ser de dudoso valor práctico.

También permite comparar diversos algoritmos que resuelvan un mismo problema, ayuda a determinar ¿cuál es la mejor solución? y ¿porqué?.

Finalmente el análisis permite obtener una comprensión profunda del algoritmo que puede llevar a implementaciones cada vez más eficientes.

Para acercarnos a las preguntas sobre tiempo de ejecución y consumo de memoria, es necesaria una perspectiva teórica y por tanto hablamos de análisis de *complejidad temporal* y de *complejidad espacial*.

Esta perspectiva teórica permite establecer una independencia entre el algoritmo y las características de su implementación y la máquina que lo ejecutará.

Esta independencia es necesaria para poder establecer conclusiones generales sobre la eficiencia de un algoritmo.

Existe una gran cantidad de instancias de máquinas diferentes entre si que son capaces de ejecutar algoritmos, cada una con sus propias características de construcción. Realizar un análisis del algoritmo en una máquina especifica limitaría las conclusiones y requeriría repetir el mismo análisis al considerar otra máquina diferente. Incluso entre máquinas con características similares, la carga de trabajo de cada máquina podría contribuir a producir diferencias entre las observaciones del algoritmo en cuestión. Adicionalmente, las máquinas están en constante evolución, nuevas implementaciones surgen con frecuencia.

En un capítulo anterior hablamos de cómo la eficiencia algorítmica tiene una dimensión fundamentalmente práctica. Y sin embargo, ahora nos encontramos con la necesidad de establecer un nivel de abstracción que nos permita alejarnos de los detalles diferenciadores entre las múltiples máquinas físicas posibles.

La solución a este contradictorio dilema es utilizar una máquina virtual. Esta máquina debe tomar en cuenta los detalles importantes sobre procesamiento y almacenamiento dejando de lado el resto de detalles técnicos que podrían interponerse a nuestro objetivo de generalización.

Modelo RAM

La máquina de acceso aleatorio, RAM por sus siglas en inglés, es una máquina teórica basada en una arquitectura de registros, de procesamiento lineal y funcionalmente equivalente a una máquina de Turing.

No entraremos en detalles sobre la arquitectura o programación de este máquina, pero si sacaremos provecho sobre la simple utilización de recursos que permite.

Con respecto a la utilización de tiempo el modelo RAM hace una diferenciación entre operaciones simples y complejas. Cada operación simple –por ejemplo las operaciones aritméticas, las operaciones lógicas o el condicional `if`– toma exactamente una unidad de tiempo en ejecutarse, a esta unidad le llamaremos *paso*.

Las operaciones complejas corresponden a ciclos y procedimientos y podemos entenderlas como composiciones de operaciones simples y complejas. El tiempo, o número de pasos, requeridos para completar una operación compleja será equivalente a la cantidad de pasos requeridos para completar cada una de las operaciones que la componen.

Cada acceso a memoria, ya sea de lectura o de escritura, toma exactamente un paso en ejecutarse.

Este modelo pasa por alto detalles comúnes sobre las máquinas reales, por ejemplo la multiplicación es usualmente una operación más compleja que la suma y por tanto toma más tiempo en ser ejecutada; el tiempo de los accesos a memoria podrían variar dependiendo de si los datos están en caché o en memoria principal.

Podemos aplicar este modelo simplificado de complejidad temporal y obtener conclusiones válidas gracias a que lo utilizaremos en conjunto con un modelo matemático de análisis asintótico donde lo importante es la tasa de crecimiento de las funciones más que su forma exacta. Más adelante hablaremos más en detalle sobre análisis asintótico.

(1) **Ejemplo**. Factorial.

```
def factorial(x):
    f = 1
    while x > 0:
        f = f * x
        x = x - 1

    return f
```

Table 1: Análisis de tiempo para `factorial` utilizando el modelo RAM

Línea	Costo	Veces
2	1	1
3	1	$n+1$
4	2	n
5	2	n
7	1	1

El ejemplo (1) ilustra una aplicación del modelo RAM para el análisis de la complejidad temporal de una versión iterativa del algoritmo para calcular el factorial de un número n.

La tabla 1 muestra un análisis de tiempo línea por línea. La columna *Línea* indica el número de línea en el algoritmo; *Costo* indica el número de pasos involucrados en esa línea de acuerdo con el model RAM; y *Veces* indica cuántas veces se ejecutará esa línea cuando se corra el algoritmo.

La cantidad de veces que se ejecuten las líneas 3, 4 y 5 dependerá del número que entre como argumento a la función, y por tanto el valor se especifica como una variable n dependiente de la entrada más que como una constante predeterminada. En este caso particular, el tamaño de la entrada es el número x que entra como argumento, pues el ciclo está condicionado en términos de x.

$$n = x \tag{1}$$

Las líneas 4 y 5 tienen un costo de 2 pasos pues cada línea realiza una operación aritmética y una asignación de memoria.

La línea 3 se ejecutará $n+1$ veces pues va de n a 0, por ejemplo si se ejecuta la función factorial con $n = 5$, la línea 3 se ejecutará 6 veces con los valores 5, 4, 3, 2, 1 y 0 en cada respectiva iteración, y es hasta la última iteración cuando $n = 0$ que la condición se dejará de cumplir y por tanto se romperá el ciclo para posteriormente ejecutar la línea 7.

Gracias a este análisis línea por línea, podemos expresar la complejidad temporal del algoritmo **??** como una función $T(n)$, donde n corresponde al tamaño de la entrada. Podemos definir a $T(n)$ como la suma de cada fila de la tabla 1, donde cada fila será un término definido como $(Veces \times Costo)$.

$$T(n) = (1 \times 1) + ((n+1) \times 1) + (n \times 2) + (n \times 2) + (1 \times 1) \tag{2}$$

$$T(n) = 1 + (n+1) + 2n + 2n + 1 \tag{3}$$

$$T(n) = 5n + 3 \tag{4}$$

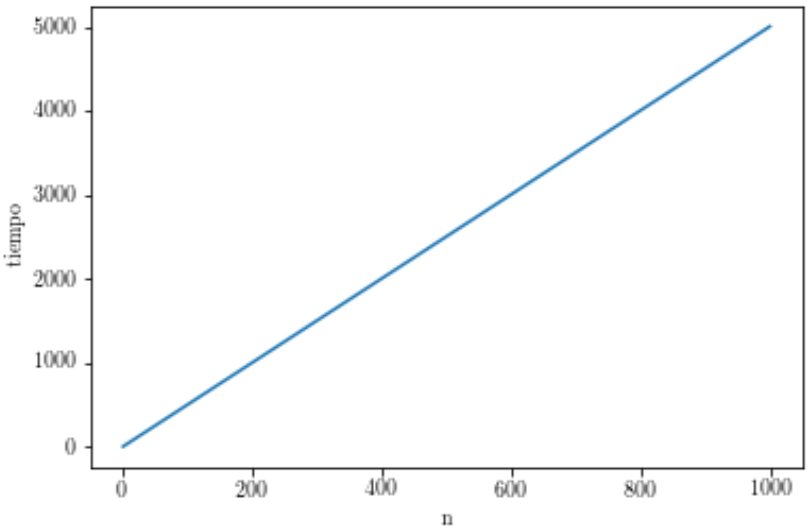

Figure 1: Complejidad de tiempo alg. Factorial