

# Análisis de Algoritmos

# Qué es el análisis de algoritmos?

Son preguntas de interés:

- Cuánto tiempo tardará en ejecutarse un programa?
- Cuánta memoria requerirá?

De la respuesta depende que tan factible sea resolver un problema de cómputo en la práctica.

# Cómo responderlas?

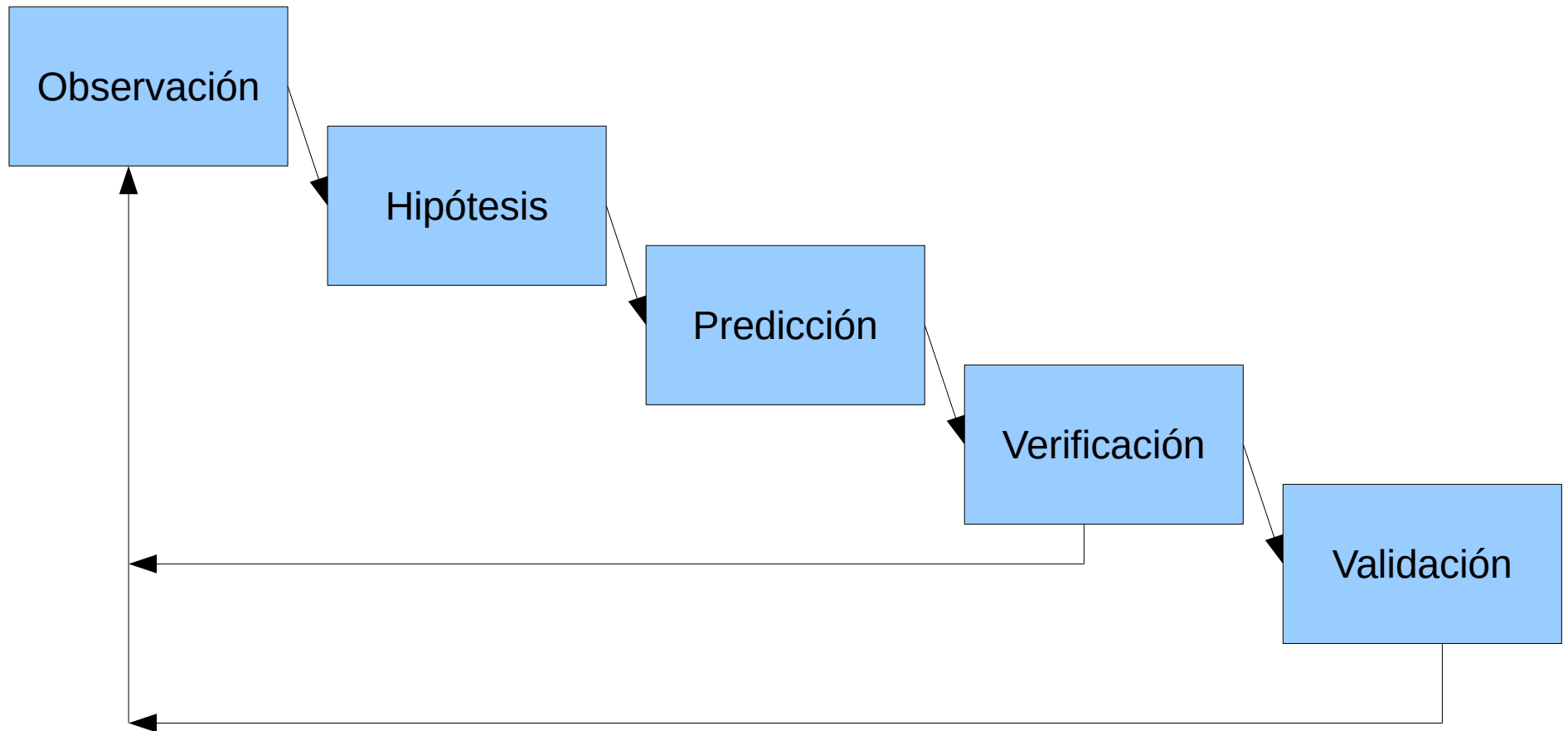
En el fondo interesa

- Medir : Cuando ya se tiene un programa
- Estimar : Qué tantos recursos se necesitaran?

Estos problemas se abordan por dos métodos

- Experimental
- Analítico

# Método experimental

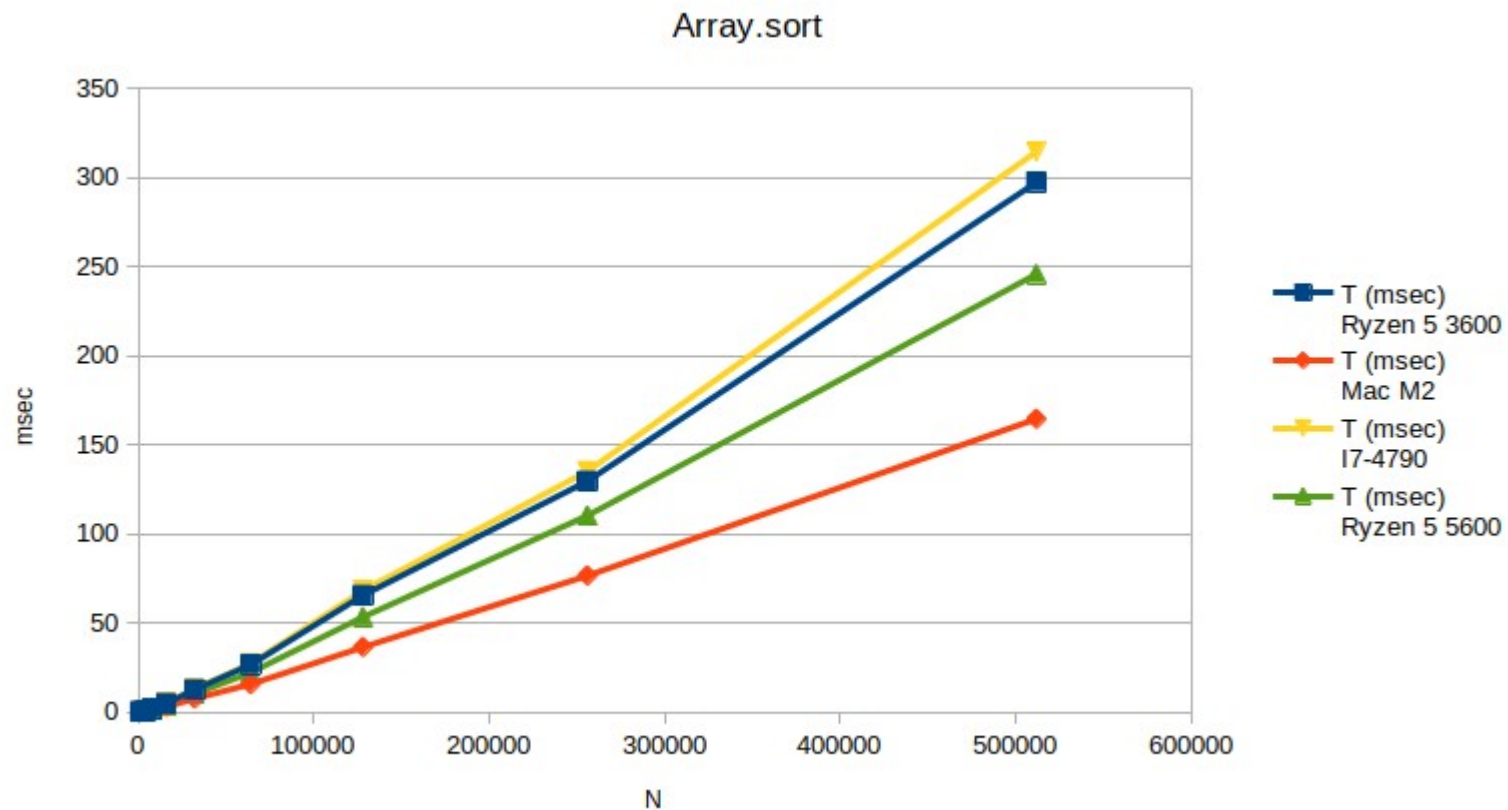


# Observación

- Es fácil medir el tiempo de ejecución de los programas
- Se encuentra que el tiempo generalmente es variable, depende de varios factores:
  - ♦ Características del hardware/software
  - ♦ El algoritmo utilizado
  - ♦ El tamaño del problema

# Ejemplo

- Desempeño del algoritmo de ordenación de arreglos en Javascript: [Array.sort](#)



# Ejemplo

## Medición de tiempo

- Programa **ThreeSum**: Encontrar todas las ternas de enteros que sumen cero.
- Clase **StopWatch**: Cronometrar la ejecución de un programa.
- Se observa que la dependencia tiempo vs tamaño generalmente no es lineal.

# Análisis de los datos experimentales

- Se tabulan los tiempos para distintos tamaños del problema de entrada
- Se grafican los datos: Observaciones
- Se busca una función que aproxime los datos: Hipótesis

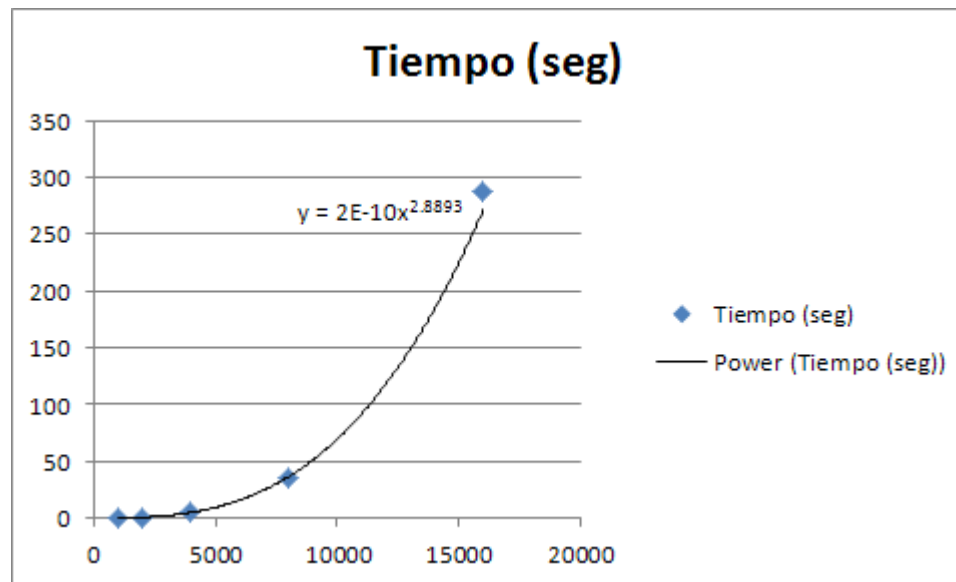


# Ejemplo: Observación

- Se ejecuta el programa y se tabulan los tiempos de ejecución

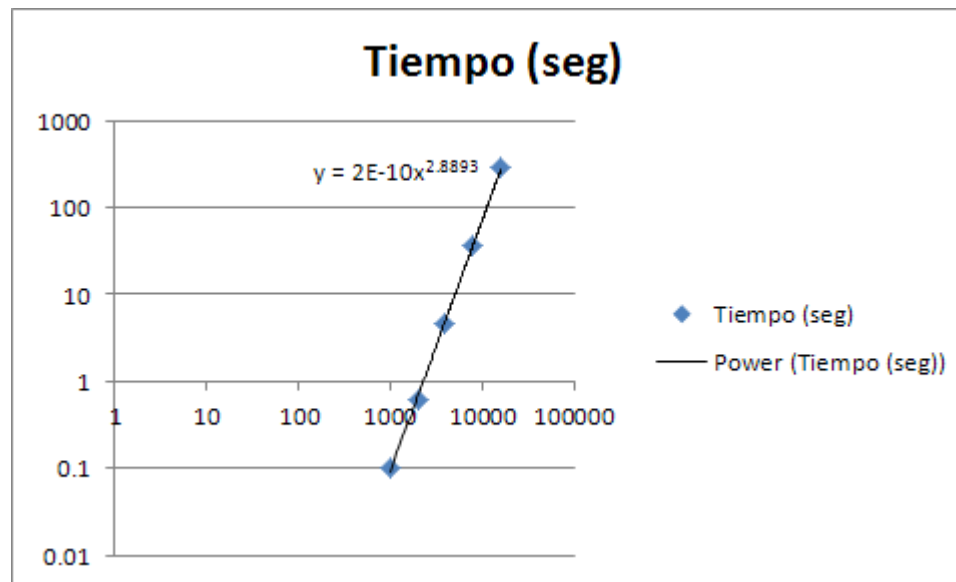
```
java -cp algs4.jar edu.princeton.cs.algs4.DoublingTest
  250    0.0
  500    0.0
 1000    0.4
 2000    0.6
 4000    4.7
 8000   36.3
```

# Ejemplo: Hipótesis



Hipótesis: El tiempo crece según una ley de potencia  $T(n) = a N^{2.89}$

# Ejemplo: Determinar el modelo



# Modelo matemático

Postulado de D. E. Knuth

El tiempo de ejecución de todo programa depende de dos factores:

- El tiempo de ejecución de cada instrucción,  $t_i$ .
- La frecuencia de ejecución de cada instrucción,  $f_i$ .

$$T(N) = \sum t_i \cdot f_i$$



Tamaño de la entrada

# Ejemplo:

## Modelo para ThreeSum

- El aspecto clave es determinar las frecuencias de las distintas sentencias.
- En el caso de ThreeSum, la condición del ciclo `if` se ejecuta

$$\binom{n}{3} = \frac{n!}{3!(n-3)!} = \frac{n(n-1)(n-2)}{3 \cdot 2 \cdot 1}$$

# Notación *tilde*

- Normalmente, el resultado del análisis es una función de muchos términos, pero uno de sus términos predomina para efectos de describir su crecimiento.

E.g.

$$\frac{N(N-1)(N-2)}{6} = \frac{N^3}{6} - \frac{N^2}{2} + \frac{N}{3}$$

Para valores grandes de  $N$ , predomina el término  $N^3/6$ .

# Notación *tilde*

- Se escribe  $\sim f(N)$  para denotar cualquier función que dividida por  $f(N)$  tienda a 1 para valores grandes de  $N$ .

e.g.

$$\frac{N(N-1)(N-2)}{6} \sim \frac{N^3}{6}$$

# Orden de crecimiento (order-of-grow)

- En el análisis algorítmico generalmente se obtienen funciones tilde de la forma

$$f(N) \sim a g(N)$$

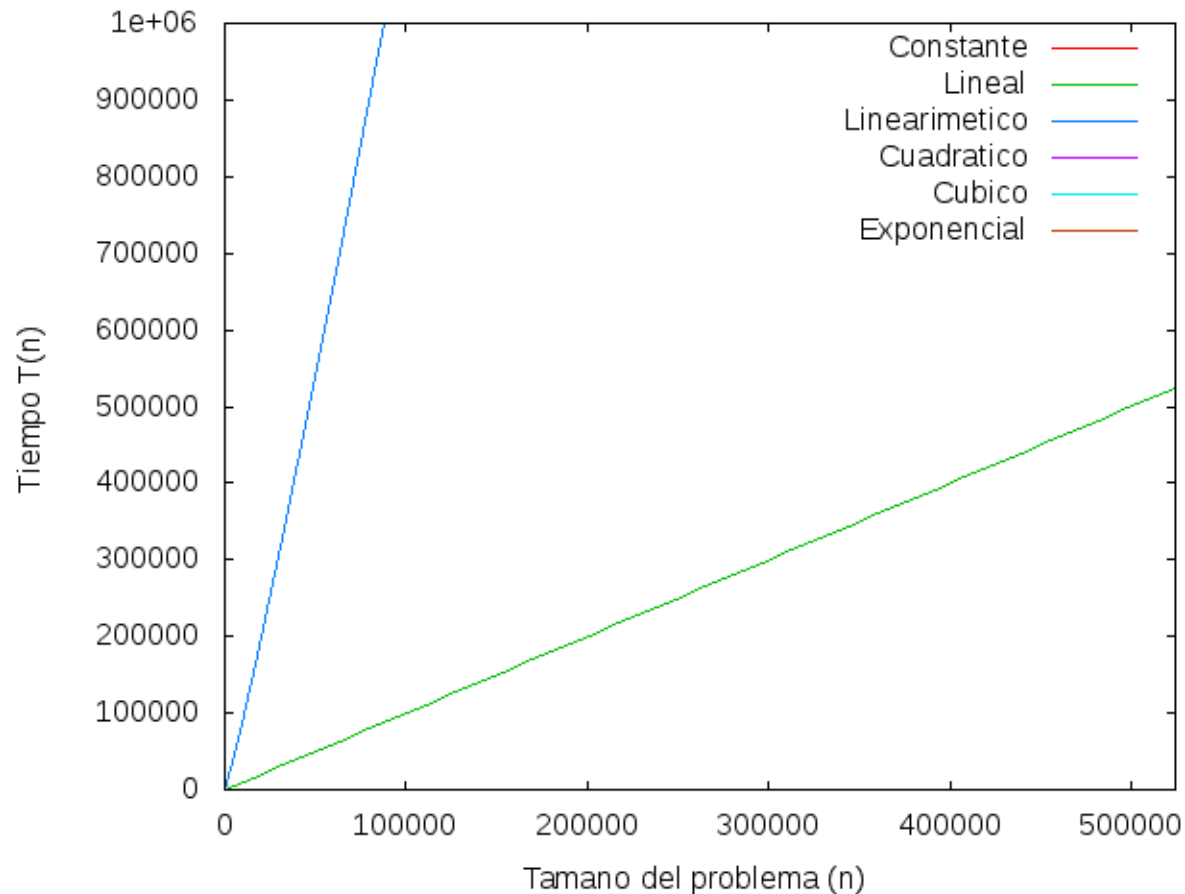
donde la  $a$  es una constante y  $g(N)$  son funciones de la forma

$$g(N) = N^b (\log N)^c$$

- A la función  $g(N)$  se le llama el **orden de crecimiento** de  $f(N)$ . (Se descarta la constante multiplicativa)



# Comparación de órdenes de crecimiento



# Comparación de órdenes (Gráfica con escalas log-log)

