Conceptos Fundamentales del Análisis de Algoritmos

Dr. Eduardo A. Rodríguez Tello

Cinvestav-Tamaulipas

15 de enero de 2018

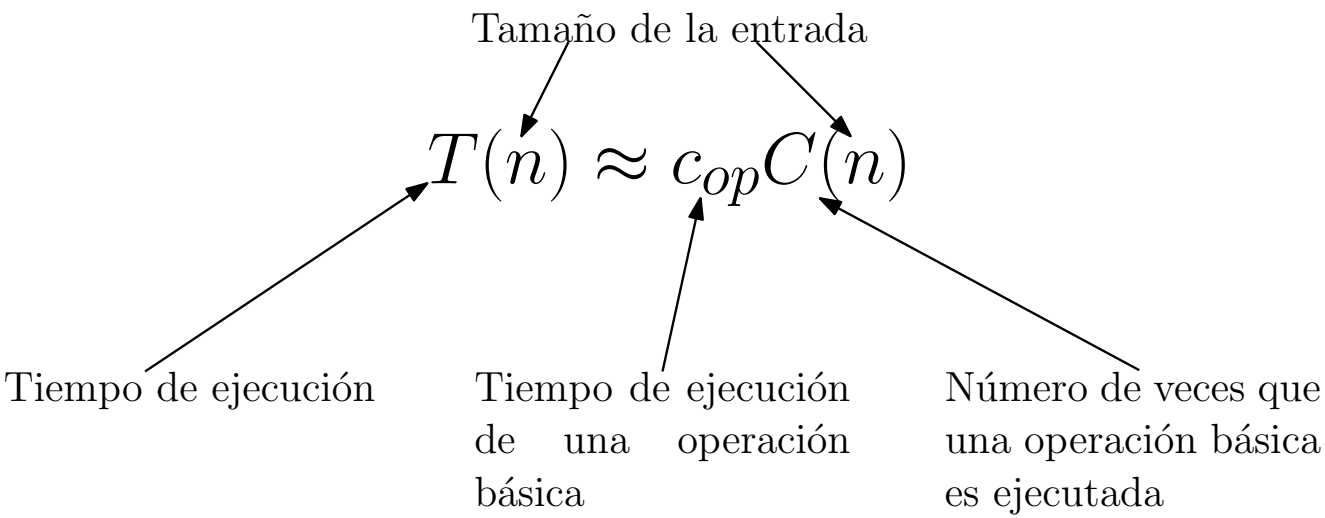
1. Conceptos fundamentales del análisis de algoritmos
   - Marco de análisis de algoritmos
   - Análisis teórico de la eficiencia temporal
   - Análisis empírico de la eficiencia temporal
   - Mejor caso, peor caso y caso promedio
   - Tipos de fórmulas para contar las operaciones básicas
   - Orden de crecimiento
   - Resumen
   - Tarea 2

Marco de análisis de algoritmos

- Cuestiones a analizar:
  - Corrección o correctitud (*correctness*)
  - Eficiencia temporal
  - Eficiencia espacial
  - Optimalidad
- Enfoques:
  - Análisis teórico
  - Análisis empírico

Análisis teórico de la eficiencia temporal

- La eficiencia temporal es analizada al determinar el número de repeticiones de la **operación básica** de un algoritmo como función del tamaño de la entrada.
- La **operación básica** de un algoritmo es aquella que contribuye de manera más importante en el tiempo de ejecución del algoritmo



Análisis teórico de la eficiencia temporal, ejemplos

Problema	Medida del tamaño de entrada	Operación básica
Buscar una valor en una lista de n elementos	Número de elementos en la lista, n	Comparación de valores
Verificar si un número n es primo	Tamaño de n (número de bits en binario)	División
Problema típico de grafos	Número de vértices y/o arcos	Visita a un vértice o recorrido de un arco

Análisis teórico de la eficiencia temporal, ejercicio

Multiplicación de dos matrices

Escriba el pseudocódigo del algoritmo para multiplicar dos matrices de tamaño $n \times n$. Responda las siguientes preguntas:

1. ¿Cuál es la operación básica de este algoritmo?
2. ¿Cuántas veces es ejecutada esta operación como función del orden n de la matriz de entrada?

Análisis teórico de la eficiencia temporal, ejercicio

Multiplicación de dos matrices

Escriba el pseudocódigo del algoritmo para multiplicar dos matrices de tamaño $n \times n$. Responda las siguientes preguntas:

1. ¿Cuál es la operación básica de este algoritmo?
2. ¿Cuántas veces es ejecutada esta operación como función del orden n de la matriz de entrada?

Respuestas:

1. Multiplicación de 2 números
2. Cada uno de los n^2 elementos del resultado es calculado como el producto escalar (punto) de dos vectores de tamaño n, lo que requiere n multiplicaciones. Por lo tanto el número total de multiplicaciones es $n \cdot n^2 = n^3$

Análisis empírico de la eficiencia temporal

- El procedimiento a seguir es:
  1. Seleccionar un ejemplo específico (representativo) de entradas
  2. Utilizar una unidad física de medida del tiempo (*e.g.*, milisegundos) o contar el número real de veces que la operación básica se ejecuta
  3. Analizar los datos empíricos obtenidos

Mejor caso, peor caso y caso promedio

- En algunos algoritmos la eficiencia depende del tipo de entradas:
  - Mejor caso: $C_{best}(n)$, el mínimo entre las entradas de tamaño n
  - Peor caso: $C_{worst}(n)$, el máximo entre las entradas de tamaño n
  - Caso promedio: $C_{avg}(n)$, el "promedio" entre las entradas de tamaño n
    - Número de veces que la operación básica se ejecuta en una entrada típica
    - NO es el promedio de $C_{worst}(n)$ y $C_{best}(n)$
    - Número esperado de operaciones básicas como una variable aleatoria bajo algunas suposiciones acerca de la distribución de probabilidad de las posibles entradas

Mejor caso, peor caso y caso promedio, ejemplo

ALGORITHM *SequentialSearch*$(A[0..n-1], K)$

```
//Searches for a given value in a given array by sequential search
//Input: An array A[0..n − 1] and a search key K
//Output: The index of the first element in A that matches K
//        or −1 if there are no matching elements
i ← 0
while i < n and A[i] ≠ K do
    i ← i + 1
if i < n return i
else return −1
```

- $C_{worst}(n) = n$, si el elemento buscado no existe o está en la última posición
- $C_{best}(n) = 1$, si el elemento buscado es el primero de la lista

Mejor caso, peor caso y caso promedio, ejemplo

ALGORITHM *SequentialSearch*$(A[0..n-1], K)$

```
//Searches for a given value in a given array by sequential search
//Input: An array A[0..n − 1] and a search key K
//Output: The index of the first element in A that matches K
//        or −1 if there are no matching elements
i ← 0
while i < n and A[i] ≠ K do
    i ← i + 1
if i < n return i
else return −1
```

- Probabilidad de tener éxito en la búsqueda, $0 \leq p \leq 1$
- Probabilidad de encontrar la primera ocurrencia en la posición i es igual para cada elemento de la lista

$$
\begin{aligned}
C_{avg}(n) &= \left[1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n} + \cdots + n \cdot \frac{p}{n}\right] + n(1-p) \\
&= \frac{p}{n}\left[1 + 2 + \cdots + i + \cdots + n\right] + n(1-p) \\
&= \frac{p}{n}\left[\frac{n(n+1)}{2}\right] + n(1-p) \\
&= \frac{p(n+1)}{2} + n(1-p)
\end{aligned}
$$

Mejor caso, peor caso y caso promedio, ejercicio

Ordenamiento por inserción

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

- Calcule lo siguiente:
  1. $C_{best}(n)$
  2. $C_{worst}(n)$

Mejor caso, peor caso y caso promedio, ejercicio

Ordenamiento por inserción

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

- Calcule lo siguiente:
  1. $C_{best}(n)$
  2. $C_{worst}(n)$

Respuestas:

1. Lista ordenada ascendentemente, 1 comparación para cada posición ($1 \leq i \leq n-1$), en total n
2. Lista ordenada descendentemente, i comparaciones para cada posición ($1 \leq i \leq n-1$), para un total de $1 + 2 + \cdots + (n-1) = \frac{n(n-1)}{2}$

Mejor caso, peor caso y caso promedio, ejercicio

Selección de guantes

Hay 22 guantes en una caja: 5 pares de guantes rojos, 4 pares de guantes amarillos, y 2 pares de guantes verdes. Se eligen los guantes a ciegas y se pueden verificar solamente después de haberlos seleccionado.

1. ¿Cuál es el número de guantes que se requiere elegir para tener al menos un par completo en el mejor de los casos?
2. ¿Cuál es el número de guantes que se requiere elegir para tener al menos un par completo en el peor de los casos?

Mejor caso, peor caso y caso promedio, ejercicio

Selección de guantes

Hay 22 guantes en una caja: 5 pares de guantes rojos, 4 pares de guantes amarillos, y 2 pares de guantes verdes. Se eligen los guantes a ciegas y se pueden verificar solamente después de haberlos seleccionado.

1. ¿Cuál es el número de guantes que se requiere elegir para tener al menos un par completo en el mejor de los casos?
2. ¿Cuál es el número de guantes que se requiere elegir para tener al menos un par completo en el peor de los casos?

Respuestas:

1. Obviamente es 2
2. Doce guantes: dado que son 22 guantes (11 pares), después de elegir 11 guantes el número 12 debe formar un par completo

Tipos de fórmulas para contar las operaciones básicas

- Fórmula exacta

$$C(n) = \frac{n(n-1)}{2}$$

- Fórmula indicando el orden de crecimiento con una constante multiplicativa conocida

$$C(n) \approx 0.5n^2$$

- Fórmula indicando el orden de crecimiento con una constante multiplicativa no conocida

$$C(n) \approx cn^2$$

Orden de crecimiento

- Importante: el orden de crecimiento para un múltiplo constante cuando $n \to \infty$

- Ejemplos:
  - ¿Cuánto más rápido correrá un algoritmo en una computadora dos veces más rápida?
  - ¿Cuánto más tiempo tomará resolver un problema cuya entrada es dos veces más grande?

Orden de crecimiento

TABLE 2.1 Values (some approximate) of several functions important for analysis of algorithms

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

Tabla tomada del libro de A. Levitin. *Introduction to the Design and Analysis of Algorithms*. Addison-Wesley; 3rd edition (October 9, 2011), ISBN-10: 0132316811.

Resumen

- La eficiencia temporal y espacial se miden como funciones del tamaño de la entrada del algoritmo
- La eficiencia temporal se mide contando el número de veces que una operación básica es ejecutada, la espacial contando el número de unidades de memoria consumidas
- La eficiencia de algunos algoritmos puede diferir significativamente para entradas del mismo tamaño, distinguir $C_{best}(n)$, $C_{worst}(n)$ y $C_{avg}(n)$
- El principal interés del marco de análisis presentado radica en el orden de crecimiento del tiempo de ejecución (de la memoria consumida) por un algoritmo a medida que el tamaño de su entrada tiende a infinito

Tarea 2

- Leer la sección 1.4 del libro *Algorithms* de Sedgewick y Wayne
- Para cada uno de los 4 algoritmos mencionados en la sección (*TwoSum, TwoSumFast, ThreeSum, ThreeSumFast*) aplique el método descrito en el apartado “Doubling ratio experiments” (página 192) y a partir de los datos generados en sus experimentos genere gráficas similares a las presentadas en la página 177.
- Genere un reporte en Latex donde presente los detalles del trabajo realizado, así como sus conclusiones respecto al orden de crecimiento de cada algoritmo.
- Fecha de entrega: lunes 22 de enero antes de las 8 AM.