

# Analisis y diseño de algoritmos. Análisis probabilístico y Quicksort

Juan Gutiérrez

September 2019

## 1 Repaso de probabilidades

Una *variable aleatoria*  $X$  es una función  $X : S \rightarrow \mathbb{R}$ . Esta función asocia un número real a cada posible salida de un experimento.

Por ejemplo, si el experimento es lanzar un dado, tendremos  $S = \{1, 2, 3, 4, 5, 6\}$ .

Una variable aleatoria podría ser  $X(1) = 1, X(2) = 2, X(3) = 3, X(4) = 4, X(5) = 5, X(6) = 6$  que representa el *número que sale en el dado*.

Otra variable aleatoria podría ser  $X(1) = 0, X(2) = 0, X(3) = 0, X(4) = 1, X(5) = 1, X(6) = 1$  que responde la pregunta si el *número que sale en el dado es mayor que 3*.

Por ejemplo, si el experimento es lanzar dos dados, tendremos  $S = \{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\} = \{11, 12, 13, \dots, 46, 56, 66\}$ .

Una variable aleatoria podría ser  $X(ij) = i + j$  que representa el *el resultado total en ambos dados*.

Otra variable aleatoria podría ser  $X(ij) = \max\{i, j\}$  que representa el *máximo entre los dos valores mostrados en los dados*.

Notación: El conjunto

$$\{s \in S : X(s) = x\}$$

se denomina *evento* y se denota por  $X = x$ .

Por ejemplo, en el último ejemplo (lanzar dos dados y ver el máximo) tendríamos que  $X = 3$  denota al evento  $\{13, 23, 33, 31, 32\}$

Definimos entonces la probabilidad de un evento de la siguiente manera.

$$Pr\{X = x\} = Pr\{s \in S : X(s) = x\} = \sum_{s \in S : X(s) = x} Pr\{s\}.$$

En el ejemplo, si suponemos que  $Pr(ij) = 1/36$  para todo  $ij$ , tenemos que  $Pr\{X = 3\} = \sum_{s \in S : X(s) = 3} Pr\{s\} = Pr\{13, 23, 33, 31, 32\} = Pr\{13\} + Pr\{23\} + Pr\{33\} + Pr\{32\} + Pr\{31\} = \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{5}{36}$ .

En palabras, esto quiere decir que la probabilidad de que al lanzar dos dados, el resultado máximo sea exactamente 3, es  $\frac{5}{36}$ .

Definimos la *función de probabilidad* de la variable aleatoria  $X$  como

$$f(x) = \Pr\{X = x\}.$$

Se sabe, por lo axiomas de probabilidad, que  $f(x) \geq 0$  y que  $\sum_x f(x) = 1$ . En el ejemplo, tendríamos que  $f(1) + f(2) + f(3) + f(4) + f(5) + f(6) = 1$  (compruebe manualmente).

El *valor esperado (esperanza, media)* de una variable aleatoria  $X$  es

$$E[X] = \sum_x x f(x).$$

En el ejemplo anterior, tendríamos que

$$\begin{aligned} E[X] &= \sum_x x \cdot f(x) \\ &= 1 \cdot f(1) + 2 \cdot f(2) + 3 \cdot f(3) + 4 \cdot f(4) + 5 \cdot f(5) + 6 \cdot f(6) \\ &= 1 \cdot \Pr\{11\} + 2 \cdot \Pr\{12, 21, 22\} + 3 \cdot \Pr\{13, 23, 33, 32, 31\} \\ &\quad + 4 \cdot \Pr\{14, 24, 34, 44, 43, 42, 41\} + 5 \cdot \Pr\{15, 25, 35, 45, 55, 54, 53, 52, 51\} \\ &\quad + 6 \cdot \Pr\{16, 26, 36, 46, 56, 66, 65, 64, 63, 62, 61\} \\ &= 1 \cdot \frac{1}{36} + 2 \cdot \frac{3}{36} + 3 \cdot \frac{5}{36} + 4 \cdot \frac{7}{36} + 5 \cdot \frac{9}{36} + 6 \cdot \frac{11}{36} \\ &= \frac{161}{36}. \end{aligned}$$

Eso quiere decir, que si lanzamos dos dados, se espera que el valor máximo de ambos dados sea  $\frac{161}{36} = 4.4722\dots$

Propiedad de linealidad: si  $X$  e  $Y$  son variables aleatorias, entonces

$$E[X + Y] = E[X] + E[Y].$$

## 2 El problema de contratación (hiring problem)

Suponga que usted desea contratar un candidato de una lista de  $n$  candidatos, de manera tal que debe pagar un costo (de tiempo o dinero) por entrevistar o por contratar algún candidato. Suponga también que solo puede analizar los candidatos uno por vez, y tiene la opción de despedir cuantas veces desee.

```
HIRE-ASSISTANT( $n$ )
1   $best = 0$            // candidate 0 is a least-qualified dummy candidate
2  for  $i = 1$  to  $n$ 
3      interview candidate  $i$ 
4      if candidate  $i$  is better than candidate  $best$ 
5           $best = i$ 
6          hire candidate  $i$ 
```

Figure 1: Tomada del libro Cormen, Introduction to Algorithms

Sea  $c_e$  el costo de entrevistar una persona y sea  $c_d$  el costo de contratar una persona. Sea  $m$  el número de personas contratadas. Tenemos que el costo total es  $O(c_e n + c_d m)$ .

Note que el costo variable depende de  $m$ . En el peor caso,  $m = n$  y el tiempo es  $O(c_e n + c_d n)$ . Analizaremos ahora que sucede en el caso promedio haciendo uso de técnicas de probabilidad.

Queremos estimar el número esperado de veces que contratamos a un nuevo empleado. El espacio de experimentos es todas las posibles  $n!$  permutaciones posibles en las cuales los candidatos pueden llegar.

Sea  $X$  la variable aleatoria que guarda el número de contrataciones. Podemos hallar el esperado de  $X$  según

$$E[X] = \sum_{x=1}^n x \cdot Pr\{X = x\}.$$

Pero es un poco complicado hacerlo así. Lo calcularemos con variables aleatorias auxiliares. Este tipo de variables aleatorias también se suelen llamar *indicadoras*.

Para cada  $i \in \{1, 2, \dots, n\}$ , definimos la variable aleatoria  $X_i$ , según

$$X_i(p) = \begin{cases} 1 & \text{si el candidato } i \text{ es contratado en la permutación } p \\ 0 & \text{si el candidato } i \text{ no es contratado en la permutación } p \end{cases}$$

Por ejemplo, si  $n = 3$  y los candidatos son  $\{1, 2, 3\}$ , el espacio es  $\{123, 132, 231, 213, 312, 321\}$ . Y tenemos que  $X_2(123) = 1, X_2(132) = 1, X_2(213) = 0, X_2(231) = 1, X_2(312) = 0, X_2(321) = 0$ .

**Ejercicio 2.1.** Hallar  $X_1, X_2, X_3$  cuando  $n = 3$  y los candidatos son  $\{1, 2, 3\}$ .

**Ejercicio 2.2.** Hallar  $X_1, X_2, X_3, X_4$  cuando  $n = 4$  y los candidatos son  $\{1, 2, 3, 4\}$ .

Note entonces que  $X$ , la variable aleatoria que guarda el número de contrataciones, puede ser expresada según

$$X = X_1 + X_2 + \dots + X_n.$$

Note también que  $E[X_i] = 1 \cdot Pr\{X_i = 1\} + 0 \cdot Pr\{X_i = 0\} = Pr\{X_i = 1\} = Pr\{\text{el candidato } i \text{ es contratado}\}$

Entonces debemos responder, cual es la probabilidad de que el candidato  $i$ -ésimo sea contratado?

Sea  $a_1, a_2, \dots, a_i, a_{i+1}, \dots, a_n$  el orden en el que llegan los candidatos. Note que

$$\begin{aligned} Pr\{\text{el candidato } i \text{ es contratado}\} &= Pr\{a_i \text{ es máximo en la secuencia } a_1, a_2, \dots, a_i\} \\ &= Pr\{\text{el máximo en la secuencia } a_1, a_2, \dots, a_i \text{ aparece al final}\} \\ &= \frac{(i-1)!}{i!} \\ &= \frac{1}{i}. \end{aligned}$$

Luego

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^n X_i\right] \\ &= \sum_{i=1}^n E[X_i] \\ &= \sum_{i=1}^n (1/i) \\ &= \ln n + O(1). \end{aligned}$$

Es decir, en promedio, el estimado del costo de contratación es  $O(c_d \ln n)$ .

### 3 Quicksort

Presentaremos un algoritmo para el problema de ordenación. Este algoritmo tiene tiempo de ejecución de  $\Theta(n^2)$  en el peor caso. Sin embargo, en el caso promedio, su tiempo de ejecución es  $\Theta(n \lg n)$ .

El algoritmo QUICKSORT aplica el paradigma de división y conquista. A continuación mostramos estos tres pasos para un arreglo  $A[p..r]$ .

- **Dividir:** Reordenamos el arreglo de manera tal que exista un índice  $q$  que cumple  $A[p..q-1] \leq A[q] < A[q+1..r]$ . Los subarreglos  $A[p..q-1]$  y  $A[q+1..r]$  son los subproblemas.
- **Conquistar:** Ordenar los subarreglos  $A[p..q-1]$  y  $A[q+1..r]$  con una llamada al mismo algoritmo.
- **Combinar:** Como los subarreglos ya están ordenados, no hay necesidad de hacer operaciones de combinación.

Recibe: Un arreglo  $A[p..r]$  de números enteros.  
Ordena el arreglo de manera creciente.

```
QUICKSORT( $A, p, r$ )  
1  if  $p < r$   
2       $q = \text{PARTITION}(A, p, r)$   
3      QUICKSORT( $A, p, q - 1$ )  
4      QUICKSORT( $A, q + 1, r$ )
```

Figure 2: Tomada del libro Cormen, Introduction to Algorithms

Recibe: Un arreglo  $A[p..r]$  de números enteros.  
Reorganiza el arreglo  $A$  y devuelve un índice  $q$  tal que  
 $A[p..q - 1] \leq A[q] < A[q + 1..r]$ .

```
PARTITION( $A, p, r$ )  
1   $x = A[r]$   
2   $i = p - 1$   
3  for  $j = p$  to  $r - 1$   
4      if  $A[j] \leq x$   
5           $i = i + 1$   
6          exchange  $A[i]$  with  $A[j]$   
7  exchange  $A[i + 1]$  with  $A[r]$   
8  return  $i + 1$ 
```

Figure 3: Tomada del libro Cormen, Introduction to Algorithms

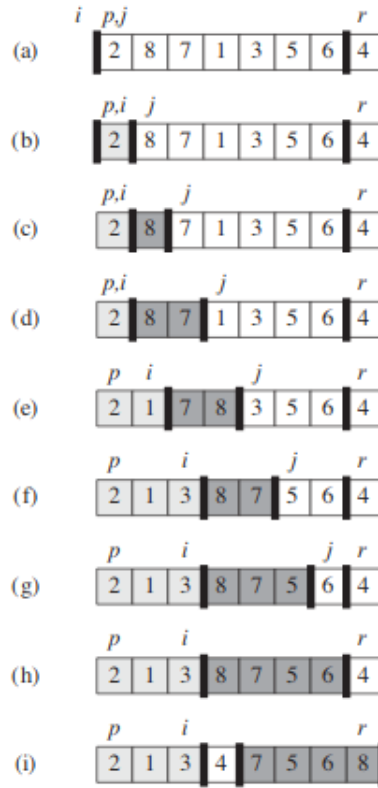


Figure 4: Tomada del libro Cormen, Introduction to Algorithms

Tenemos la siguiente invariante para la subrutina PARTITION:  
Al inicio de cada iteración del bucle de las líneas 3-6,

1.  $A[p..i] \leq x$
2.  $A[i+1..j-1] > x$
3.  $A[r] = x$

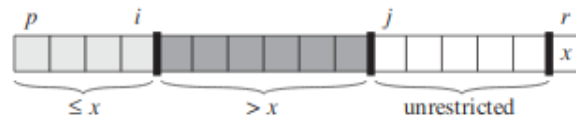


Figure 5: Tomada del libro Cormen, Introduction to Algorithms

- Inicialización. Antes de comenzar el primer for, tenemos  $i = p - 1$  y  $j = p$ . Luego, las dos primeras condiciones de la invariante son trivialmente satisfechas. Además, por causa de la línea 1, tenemos que  $A[r] = x$ .
- Manutención. Dependiendo del if de la línea 4, tenemos dos posibilidades. Si  $A[j] > x$  entonces simplemente se incrementa  $j$  y las condiciones 1 y 2 valen para  $j + 1$  en lugar de  $j$ .

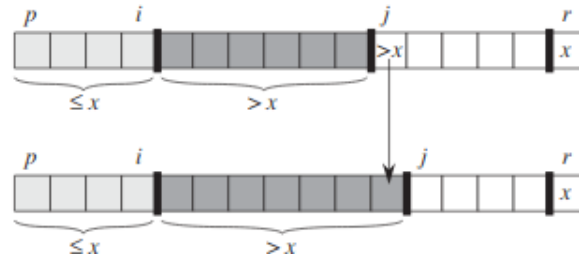


Figure 6: Tomada del libro Cormen, Introduction to Algorithms

Si  $A[j] \leq x$  entonces se incrementa  $i$ , se intercambia  $A[i]$  con  $A[j]$ , y se incrementa  $j$ . Como  $A[i] > x$ , la condición 1 será satisfecha. Como  $A[j - 1] \leq x$ , la condición 2 será también satisfecha.

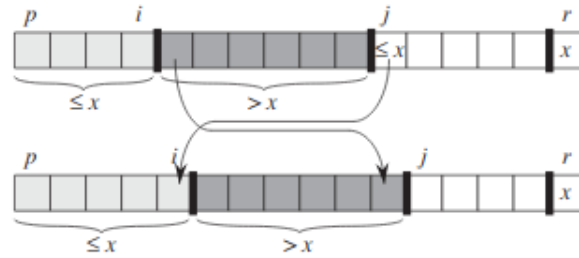


Figure 7: Tomada del libro Cormen, Introduction to Algorithms

- Terminación. Al terminar el for, tenemos  $j = r$  y portanto,  $A[1..i] \leq x < A[i + 1..r - 1]$  y  $A[r] = x$ .  
Luego de esto, debido a la línea 7, se tendrá un índice  $q$  tal que  $A[1..q] \leq A[q] < A[q + 1..r]$ .

Es relativamente fácil ver que el tiempo de ejecución de la subrutina PARTITION en el subarreglo  $A[p..r]$  es  $\Theta(n)$ , con  $n = r - p + 1$ .

**Ejercicio 3.1.** Ilustre la operación PARTITION en el arreglo  $A = [13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11]$ .

## 4 Análisis de tiempo del Quicksort

Sea  $T(n)$  una cota superior para el tiempo de ejecución del QUICKSORT al recibir  $n$  elementos, entonces.

$$T(n) \leq \max_{0 \leq q \leq n-1} \{T(q) + T(n-1-q)\} + O(n).$$

Por otro lado, si  $T(n)$  es una cota superior para el tiempo de ejecución del QUICKSORT al recibir  $n$  elementos, entonces.

$$T(n) \geq \min_{0 \leq q \leq n-1} \{T(q) + T(n-1-q)\} + O(n).$$

En líneas generales, el tiempo de ejecución del Quicksort depende de si la partición es balanceada o no balanceada. Si la partición es balanceada, es tan rápido como el MERGESORT ( $\Theta(n \lg n)$ ); si la partición no lo es, puede ser tan lento como el INSERTIONSORT ( $\Theta(n^2)$ ).

### 4.1 Peor caso

Intuitivamente, ocurre cuando la rutina de partición produce un subproblema con  $n-1$  elementos y otro con 0 elementos. Esta situación puede darse en cada llamada recursiva. En ese caso, suponiendo  $T(0) = 0$ , tenemos

$$T(n) = T(n-1) + T(0) + O(n) = T(n-1) + O(n).$$

Portanto  $T(n) = O(n^2)$  para este caso.

A continuación, probaremos que, para el caso general,  $T(n) = O(n^2)$  y portanto el caso anterior es efectivamente el peor caso.

Sea  $T(n)$  el tiempo de ejecución para el Quicksort (en un caso cualquiera). Probaremos por inducción que  $T(n) \leq cn^2$  (para todo  $n \geq 0$ , con algún  $c > 0$ ). Si  $n = 0$  entonces, como suponemos que  $T(0) = 0$ , la desigualdad se cumple. Suponga ahora que  $n > 0$ . Tenemos que, cuando  $c \geq k$ ,

$$\begin{aligned} T(n) &\leq \max_{0 \leq q \leq n-1} \{T(q) + T(n-1-q)\} + kn \\ &\leq \max_{0 \leq q \leq n-1} \{cq^2 + c(n-q-1)^2\} + kn \\ &\leq c \cdot \max_{0 \leq q \leq n-1} \{q^2 + (n-q-1)^2\} + kn \\ &\leq c \cdot (n-1)^2 + kn \\ &\leq cn^2 - c(2n-1) + kn \\ &\leq cn^2. \end{aligned}$$

Portanto, en el peor caso, el QUICKSORT tiene tiempo de ejecución  $\Theta(n^2)$ .

**Ejercicio 4.1.** Pruebe que  $\max_{0 \leq q \leq n-1} \{q^2 + (n-q-1)^2\} = (n-1)^2$



## 4.2 Mejor caso

Intuitivamente, el mejor caso ocurre cuando ambos subproblemas tienen aproximadamente el mismo tamaño  $((n-1)/2)$ . En ese caso tenemos

$$T(n) = T(\lceil (n-1)/2 \rceil) + T(\lfloor (n-1)/2 \rfloor) + kn$$

Portanto  $T(n) = \Theta(n \lg n)$  para este caso.

A continuación, probaremos que, en el caso general,  $T(n) = \Omega(n \lg n)$  y portanto el caso anterior es efectivamente el mejor caso.

Sea  $T(n)$  el tiempo de ejecución para el Quicksort (en un caso cualquiera). Probaremos por inducción que  $T(n) \geq cn \lg n$  para algún  $c > 0$ . Si  $n = 1$  entonces, como  $T(1) \geq 0$ , la desigualdad se cumple.

Suponga ahora que  $n > 1$ . Tenemos que, cuando  $c \leq k/3$  y  $n \geq 2$ ,

$$\begin{aligned} T(n) &\geq \min_{0 \leq q \leq n-1} \{T(q) + T(n-1-q)\} + kn \\ &\geq \min_{0 \leq q \leq n-1} \{cq \lg q + c(n-q-1) \lg (n-q-1)\} + kn \\ &\geq c \cdot \min_{0 \leq q \leq n-1} \{q \lg q + (n-q-1) \lg (n-q-1)\} + kn \\ &= c \cdot (n-1) \lg \left( \frac{n-1}{2} \right) + kn \\ &= c(n-1) \lg (n-1) - c(n-1) + kn \\ &= cn \lg (n-1) - c \lg (n-1) - c(n-1) + kn \\ &\geq cn \lg (n/2) - c \lg (n-1) - c(n-1) + kn \\ &= cn(\lg n - 1) - c \lg (n-1) - c(n-1) + kn \\ &= cn \lg n - cn - c \lg (n-1) - c(n-1) + kn \\ &= cn \lg n - c(n + \lg (n-1) + (n-1)) + kn \\ &= cn \lg n - c(2n + \lg (n-1) - 1) + kn \\ &\geq cn \lg n - c(2n + \lg (n-1) - 1) + 3cn \\ &\geq cn \lg n + c(n - \lg (n-1) + 1) \\ &\geq cn \lg n. \end{aligned}$$

Portanto, en el mejor caso, el Quicksort tiene tiempo de ejecución  $\Theta(n \lg n)$ .

## 4.3 Caso promedio

El tiempo de ejecución de QUICKSORT es dominado por el tiempo del procedimiento PARTITION.

Recordemos el procedimiento PARTITION:

```

PARTITION( $A, p, r$ )
1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4      if  $A[j] \leq x$ 
5           $i = i + 1$ 
6          exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 

```

Figure 8: Tomada del libro Cormen, Introduction to Algorithms

Observe que cada vez que este procedimiento es llamado como máximo  $n$  veces. Además, cada llamada a partition toma tiempo de ejecución  $O(r - p + 1)$  (número de veces que es ejecutado el for de las líneas 3–6). Este tiempo de ejecución también se puede expresar por el número de veces en que la línea 4 es ejecutada.

Sea  $X$  la variable aleatoria que cuenta el número de veces que la línea 4 del procedimiento PARTITION es ejecutada durante toda la ejecución de QUICK-SORT. Mostraremos que  $E[X] = O(n \lg n)$ .

Denotamos por  $z_1, z_2, \dots, z_n$  a los elementos del arreglo  $A$ , con  $z_1 \leq z_2 \leq \dots \leq z_n$ . También, para cada  $i, j$ , sea  $Z_{ij} = \{z_i, \dots, z_j\}$ . Y para cada  $ij$  definimos las siguientes variables aleatorias:

$$X_{ij} = \begin{cases} 1 & \text{si } z_i \text{ es comparado con } z_j \\ 0 & \text{si } z_i \text{ no es comparado con } z_j \end{cases}$$

Note que

$$\begin{aligned}
E[X] &= \sum_{1 \leq i < j \leq n} E[X_{ij}] \\
&= \sum_{1 \leq i < j \leq n} \Pr[z_i \text{ es comparado con } z_j] \\
&= \sum_{1 \leq i < j \leq n} \Pr[z_i \text{ o } z_j \text{ es el primer pivote escogido de } Z_{ij}] \\
&= \sum_{1 \leq i < j \leq n} \frac{2}{j - i + 1} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j - i + 1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k + 1} \\
&\leq \sum_{i=1}^{n-1} \sum_{k=1}^n \frac{2}{k} \\
&= \sum_{i=1}^{n-1} O(\lg n) \\
&= O(n \lg n)
\end{aligned}$$