Asesoría 2: Quicksort y Análisis Probabilístico

October 23, 2020

1 Quicksort

Ejercicio 2 (3 ptos). ¿Cual es el tiempo de ejecución de Quicksort en el peor caso? ¿Cual es el tiempo de ejecución de Quicksort en el mejor caso? Para ambas situaciones, de un ejemplo de un arreglo de 7 elementos.

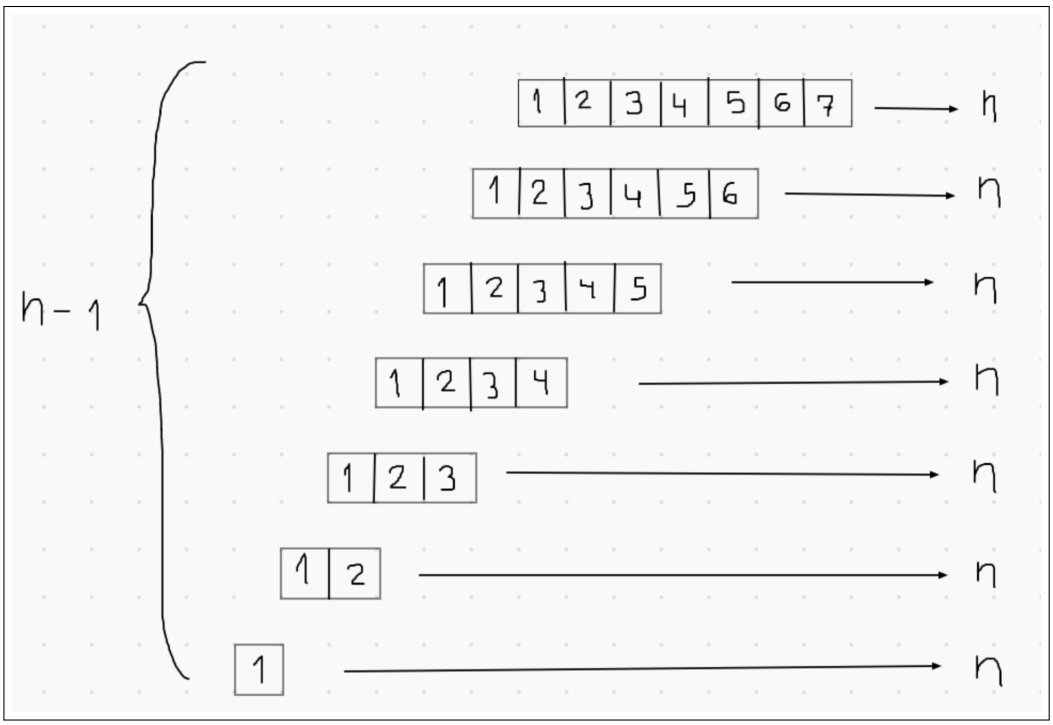
Tomando como referencia la implementación del Cormen:

```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)
```

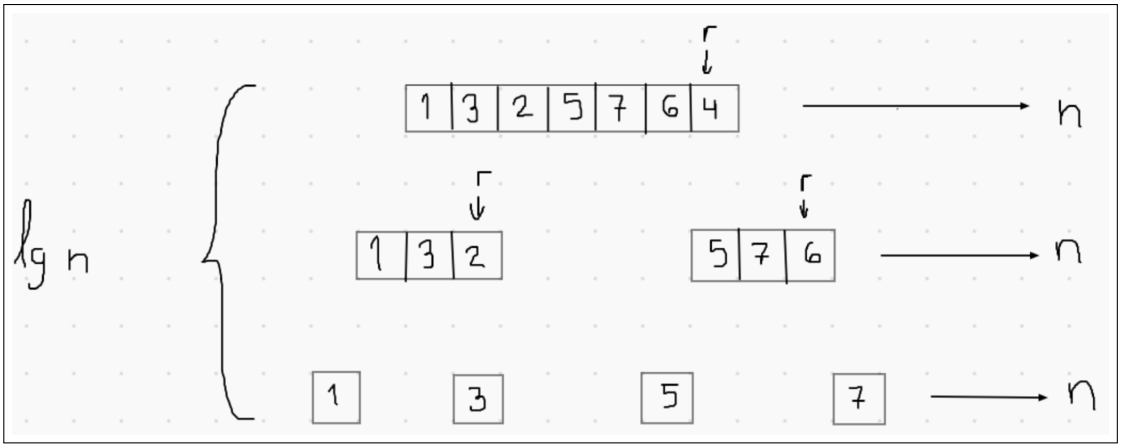
```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

¿Cuál es el tiempo de ejecución de Quicksort en el peor caso?

En peor caso de Quicksort ocurre cuando el arreglo ya está ordenado. En el primer paso, se elige el último elemento como pivote, el cual es el mayor elemento. La partición creará dos arreglos, uno de tamaño $n-1$ y otro de tamaño 0, ya que cada elemento que se compare con el pivote, será menor o igual a este. Y así para cada iteración, ya que el siguiente pivote que se elija, será el mayor elemento en esa partición del arreglo. Este procedimiento, generará $n-1$ llamadas a PARTITION el cual corre en $\Theta(n)$. Por lo tanto, nos da un tiempo de ejecución de $\Theta(n^2)$.



¿Cuál es el tiempo de ejecución de Quicksort en el mejor caso?



El mejor caso de Quicksort ocurre cuando cada pivote que se elige, es la mediana del arreglo. Esto causará que PARTITION cree dos arreglos de tamaño $\frac{n}{2}$ cada vez, lo cual hace que la recursión tenga $lg\ n$ niveles y como en cada nivel se hace un trabajo en $\Theta(n)$, el tiempo de Quicksort en este caso es de $\Theta(n\ lg\ n)$.

Ejercicio 13. ¿Cual es el tiempo de ejecución de QUICKSORT cuando todos los elementos del arreglo A tienen el mismo valor?

Cuando todos los elementos de A tienen el mismo valor, el tiempo de Quicksort es de $\Theta(n^2)$, ya que este es otro arreglo donde ocurre el peor caso de Quicksort. Cuando se llame a PARTITION, cada elemento que se compare con el pivote, será menor o igual a este, lo que causará que PARTITION genere dos subarreglos, uno de tamaño $n-1$ y otro de tamano 0.

Ejercicio 12. Escriba una función que reciba un vector con n letras A y B y, a través de intercambios, mueve todas las A al comienzo del vector. Su función deberá tener tiempo de ejecución $O(n)$.

```cpp
#include <iostream>
#include <vector>

using namespace std;

void swap(vector<char>& A, int p1, int p2) {
    char aux = A[p1];
    A[p1] = A[p2];
    A[p2] = aux;
}

void rearrange(vector<char>& A) {
    int p1 = 0;
    int p2 = A.size() - 1;

    while (p1 < p2) {
        if (A[p1] == 'A' && A[p2] == 'B') {
            ++p1;
            --p2;
        } else if (A[p1] == 'A' && A[p2] == 'A') {
            ++p1;
        } else if (A[p1] == 'B' && A[p2] == 'A') {
            ::swap(A, p1, p2);
            ++p1;
            --p2;
        } else if (A[p1] == 'B' && A[p2] == 'B') {
            --p2;
        }
    }
}

int main() {

    vector<char> A = {'A', 'A', 'B', 'B', 'A', 'A', 'A', 'B', 'A', 'B'};
    // vector<char> A = {'B', 'B', 'B', 'B', 'A', 'A', 'A', 'A', 'A', 'A'};
    // vector<char> A = {'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B'};
    // vector<char> A = {'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A'};

    cout << "Antes del swap:\n";
    for (auto& c : A) {
        cout << c << " ";
    }
    cout << "\n";

    rearrange(A);
```

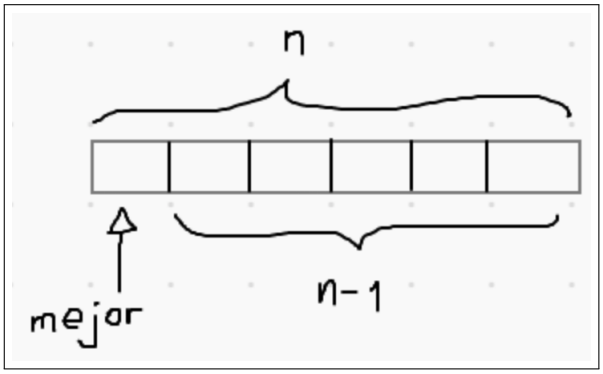
```
46
47      cout << "Despues del swap:\n";
48      for (auto& c : A) {
49          cout << c << " ";
50      }
51      cout << "\n";
52
53      return 0;
54  }
```

2 Análisis Probabilístico

Ejercicio 10. En el pseudocódigo de HIRE-ASSISTANT, suponiendo que los candidatos se presentan en de manera aleatoria uniforme, ¿cual es la probabilidad que se contrate exactamente una vez?, ¿cual es la probabilidad de que se contrate n veces?

Para facilitar la resolución de este ejercicio, veamos a los candidatos como un arreglo de números, donde cada número representa qué tan bueno es el candidato. Además, el arreglo tiene n números por los n candidatos.

¿Cuál es la probabilidad de que se contrate una sola vez? Esto significa que el mejor candidato, se presenta primero. Es decir, que el máximo elemento en el arreglo, esté en la primera posición.



Claramente, $n!$ es nuestro mar de posibilidades, ya que representa todos los arreglos posibles que se pueden formar con los n candidatos. De ese mar de posibilidades, tenemos $(n-1)!$ escenarios posibles donde el mejor candidato se presenta primero, es decir, el primer elemento del arreglo es el mayor. Por lo tanto, la probabilidad de que se contrate exactamente una vez es:

$$\frac{(n-1)!}{n!}$$

$$= \frac{1}{n}$$

¿Cuál es la probabilidad de contratar a todos los candidatos? Esto significa que cada candidato que se presenta es mejor que el anterior. Es decir, que el arreglo esté ordenado de manera ascendente.

Nuevamente, $n!$ es nuestro mar de posibilidades y dentro de ese mar de posibilidades, solo un arreglo está ordenado de manera ascendente. Por lo tanto, la probabilidad de que contratemos n veces es:

$$\frac{1}{n!}$$

Ejercicio 9. Sea X una variable aleatória que guarda el número de caras en dos lanzamientos de uma moneda justa. ¿Cuánto vale $E[X^2]$? ¿Cuánto vale $E[X]^2$?

Primero, recordar que:

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

Dado este experimento, nuestra variable aleatoria X puede tomar los valores $x = 0$, $x = 1$ o $x = 2$, ya que lanzando dos veces una moneda, nos pueden tomar 0 caras, 1 cara o 2 caras. Por lo tanto:

$$X = \{0, 1, 2\}$$

Entonces,

$$E[X]^2 = [\ \sum_{x=0}^{2} x.Pr(X = x)\ ]^2$$

$$= [\ 2.\frac{1}{4} + 1.\frac{1}{2} + 0.\frac{1}{4}\ ]^2$$

$$E[X]^2 = 1$$

$2.\frac{1}{4}$ porque la probabilidad de que toquen dos caras seguidas es la multiplicación de la probabilidad de que toque una cara (Pr=$\frac{1}{2}$) **y** que luego toque otra (Pr=$\frac{1}{2}$). La multiplicación se hace por la regla de la multiplicación: cuando son eventos independientes y queremos que ambos ocurran. Un caso análogo ocurre con $1.\frac{1}{2}$ y con $0.\frac{1}{4}$.

Ahora para el otro caso:

$$E[X^2] = [\ \sum_{x=0}^{2} x^2.Pr(X = x)\ ]$$

$$= 2^2.\frac{1}{4} + 1^2.\frac{1}{2} + 0^2.\frac{1}{4}$$

$$E[X^2] = \frac{3}{2}$$

Ejercicio 4 (4 ptos). Considere el siguiente algoritmo que determina el segundo mayor elemento de un vector $v[1 \ldots n]$ com $n \geq 2$ números positivos distintos.

```
ALGO(v, n)
 1: mayor = 0
 2: segundo_mayor = 0
 3: for i = 1 to n
 4:    if v[i] > mayor
 5:       segundo_mayor = mayor
 6:       mayor = v[i]
 7:    else
 8:       if v[i] > segundo_mayor
 9:          segundo_mayor = v[i]
10: return  segundo_mayor
```

Suponga que v es una permutación de 1 a n escogida de entre todas las permutaciones de 1 a n con distribución uniforme de probabilidad. Sea X la variable aleatória que guarda el número de veces que la variable *segundo_mayor* es alterada (osea, el número de ejecuciones de las líneas 5 y 9 del algoritmo) en una llamada a ALGO(v, n). Calcule el valor esperado de X.

$$X_i = \begin{cases} 1 : \text{la línea 5 o 9 es ejecutada en la iteración i} \\ \\ 0 : \text{caso contrario} \end{cases}$$

Como X_i es una variable aleatoria indicadora, su esperanza sería:

$$E[X_i] = Pr(X_i = 1)$$

$$= Pr(\text{"la línea 5 es ejecutada en la iteración i"}) + Pr(\text{"la línea 9 es ejecutada en la iteración i"})$$

$$= Pr(\text{"i es el mayor elemento en A[1...i]"}) + Pr(\text{"i es el segundo mayor elemento en A[1...i]"})$$

$$= \frac{1}{i} + \frac{1}{i}$$

$$E[X_i] = \frac{2}{i}$$

Ahora:

$$E[X] = \sum_{i=1}^{n} \frac{2}{i}$$

$$= 2 \sum_{i=1}^{n} \frac{1}{i}$$

$$= 2H_n$$

$$= 2 \ln n$$

Por lo tanto, el número esperado de veces en las que se alterará a la variable *segundo_mayor* es $2 \ln n$.

Ejercicio 3. Considere el siguiente algoritmo que determina el el mayor y menor elemento de un vector $v[1 \dots n]$ con números positivos distintos.

MAYORMENOR(v, n)

```
1: mayor = v[1]
2: menor = v[1]
3: for i = 2 to n
4:    if v[i] > mayor
5:       mayor = v[i]
6:    else
7:       if v[i] < menor
8:          menor = v[i]
9: return mayor, menor
```

Suponga que la entrada del algoritmo es una permutación de 1 a n escogida uniformemente dentre todas las permutaciones de 1 a n. ¿Cual es el número esperado de comparaciones ejecutadas en la línea 7 del algoritmo? ¿Cual es el número esperado de atribuiciones efectuadas en la línea 8 del algoritmo?

Notar que la línea 7 es ejecutada siempre y cuando el IF de la línea 4 evalúa a FALSE. Es decir, cuando el elemento en la posición i del arreglo no es el máximo en $A[1 \dots i]$.

Notar que la línea 8 es ejecutada si el elemento el la posición i del arreglo, es el menor en $A[1 \dots i]$.

Sea X la variable aleatoria que cuenta el número de ejecuciones de la línea 7. Sea X_i la variable indicadora que cuenta si la línea 7 es ejecutada en la iteración i para $i = 2 \dots n$.

$$E[X_i] = Pr(X_i = 1)$$

$$Pr(X_i = 1) = Pr(\text{"se ejecuta la línea 7 en la iteración i"})$$

$$= Pr(\text{"A[i] no es el elemento máximo en A[1...i]"})$$

$$= 1 - Pr(\text{"A[i] es el elemento máximo en A[1...i]"})$$

$$= 1 - \frac{1}{i}$$

Entonces:

$$E[X] = \sum_{i=2}^{n} E[X_i]$$

$$= \sum_{i=2}^{n} 1 - \frac{1}{i}$$

$$= \sum_{i=1}^{n} 1 - \frac{1}{i}$$

$$= n - H_n$$

$$= n - \ln n$$

Sea Y la variable aleatoria que cuenta el número de veces que se ejecuta la línea 8. Sea Y_i la variable indicadora que cuenta si la línea 8 es ejecutada en la iteración i para $i = 2 \ldots n$.

$$E[Y_i] = Pr(Y_i = 1)$$

$$Pr(Y_i = 1) = Pr(\text{"se ejecuta la línea 8 en la iteración i"})$$

$$= Pr(\text{"A[i] es mínimo en A}[1 \ldots \text{i]"})$$

$$= \frac{1}{i}$$

Entonces:

$$E[Y] = \sum_{i=2}^{n} E[Y_i]$$

$$= \sum_{i=2}^{n} \frac{1}{i}$$

$$= \sum_{i=1}^{n} \frac{1}{i} - 1$$

$$= H_n - 1$$

$$= \ln n - 1$$