Análisis de algoritmos

Erick Gonzalez Parada ID: 178145
Matemáticas discretas, Universidad de las Américas Puebla, Puebla, México

September 10, 2023

Abstract

Análisis de Quick Select Lomuto.

Keywords: búsqueda de posición, algoritmo

1 Introducción

El algoritmo *Quick Select* de Lomuto es una variante de esta en donde notablemente se utiliza *Quick Sort*, sin embargo, el objetivo del *Quick Select* sera encontrar el k-ésimo de un array no necesariamente ordenado.
IMPORTANTE: Se asume que el arreglo tiene al menos k posiciones (base de index 0).

2 Explicación del algoritmo

Nota: en pseudocódigo los comentarios son denotas con un '%' y no pueden contener acentos, las asignaciones son con el siguiente combo de símbolos ':=' y la comparación se hace con un solo signo de igual '='.

```
  funcion quick_select(arreglo, izquierda, derecha, k)
    % Si el rango solo contiene un elemento, regresar ese elemento
    si izquierda = derecha
        regresar arreglo[izquierda]
    % Dividir el arreglo en dos partes
    pIndex := particion(arreglo, izquierda, derecha)
    % Si el indice del pivote es igual a k, regresar el elemento en esa posicion
    si k = pIndex
        regresar arreglo[k]
    % Si k es menor que el indice del pivote, buscar en la parte izquierda del arreglo
    sino si k < pIndex
        regresar quick_select(arreglo, izquierda, pIndex - 1, k)
    % Si k es mayor que el indice del pivote, buscar en la parte derecha del arreglo
    sino
        regresar quick_select(arreglo, pIndex + 1, derecha, k)

funcion particion(arreglo, izquierda, derecha)
    % Seleccionar el ultimo elemento como pivote
    pivote := arreglo[derecha]
    % Inicializar el indice de particion con el valor de izquierda
    pIndex := izquierda
```

```
% Recorrer el arreglo desde izquierda hasta derecha - 1
para i desde izquierda hasta derecha - 1
    % Si un elemento es menor o igual al pivote, intercambiarlo con el elemento en
    si arreglo[i] <= pivote
        intercambiar(arreglo[i], arreglo[pIndex])
        pIndex := pIndex + 1
% Intercambiar el elemento en pIndex con el pivote y regresar pIndex
intercambiar(arreglo[pIndex], arreglo[derecha])
regresar pIndex
```

3 Análisis del algoritmo

mejor caso

Para el mejor caso solo necesitaríamos ejecutar un bloque del algoritmo,
el que se sale por que nuestro array es de tamaño 1, es decir, que tenga solo 1 elemento.

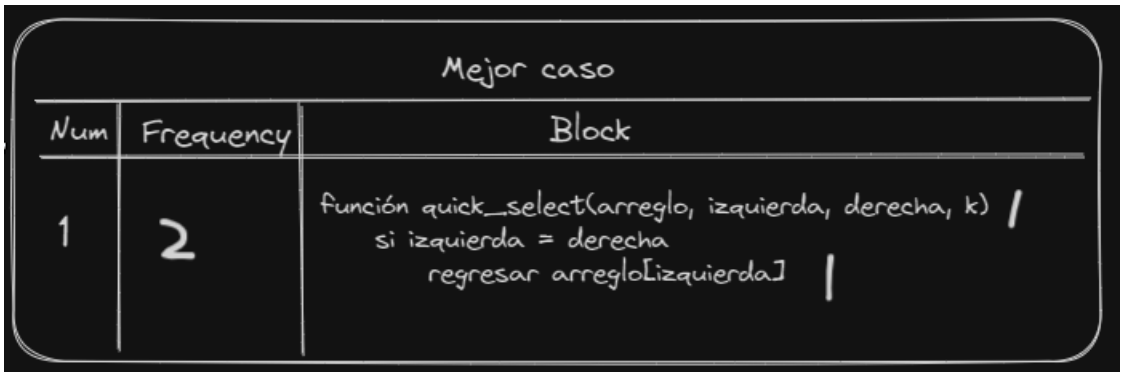

Figura 1: Mejor caso

Función característica:

$$2$$

Complejidad asintótica

$$\Omega(2) = \Omega(k)$$

Donde k es la idea/concepto de una constante

La gráfica de la complejidad asintótica es la siguiente:

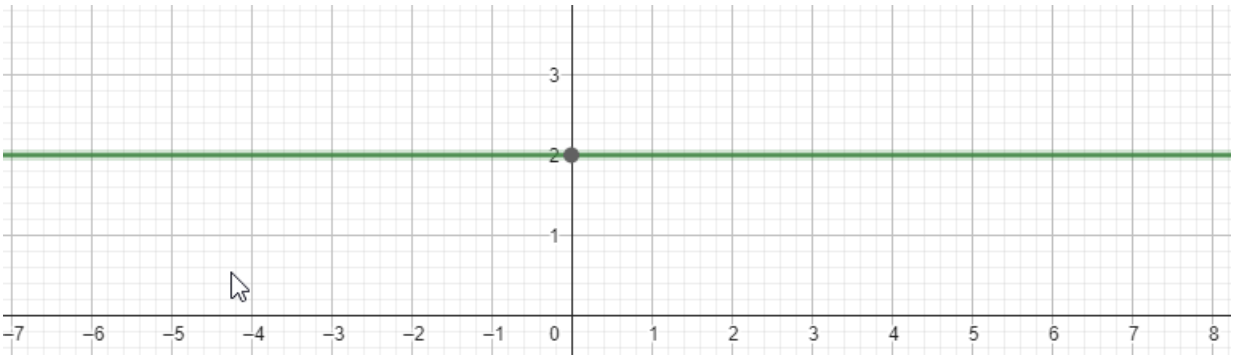

Figura 2: Gráfica de caso de la fig 1

Lo mejor del mundo en cuanto a velocidad (ver fig 2), una velocidad constante.

peor caso

Para el peor caso tenemos:

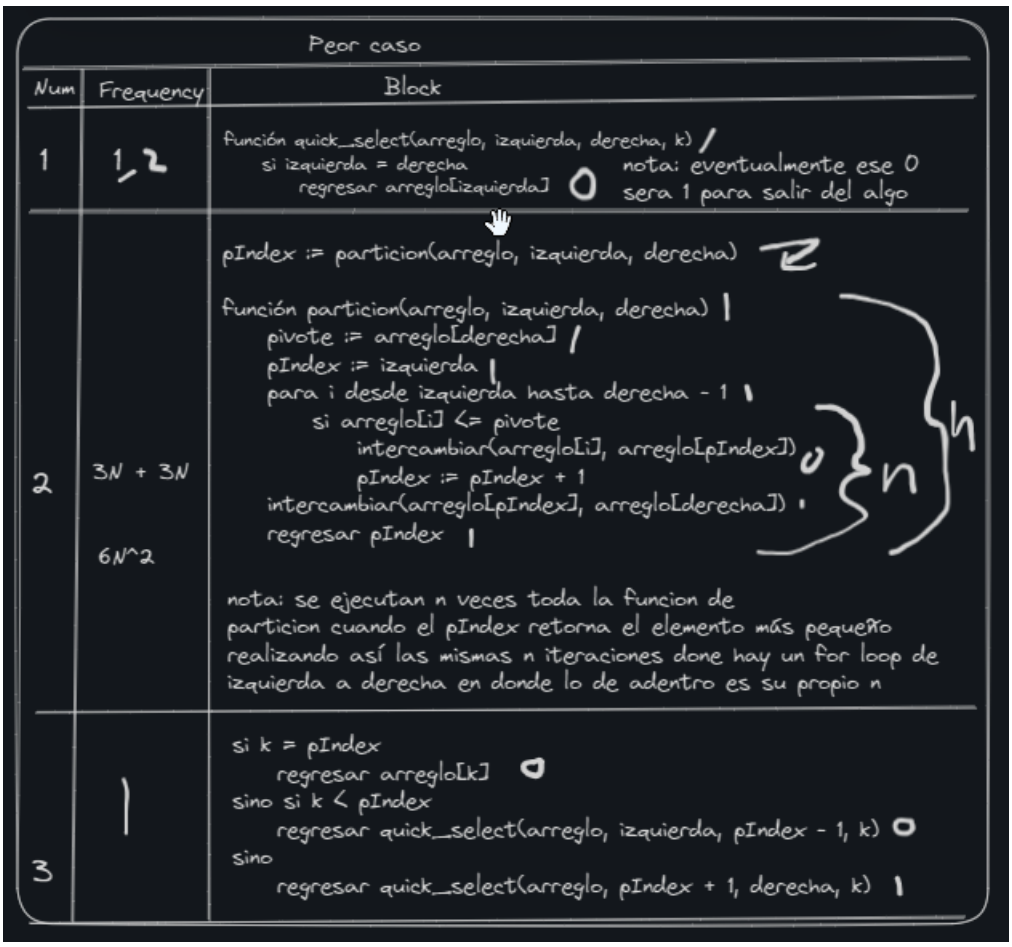

Figura 3: Peor caso

Función característica:

$$2 + 6n^2 + 1$$

Complejidad asintótica

$$O(n^2)$$

La gráfica de la complejidad asintótica es la siguiente:

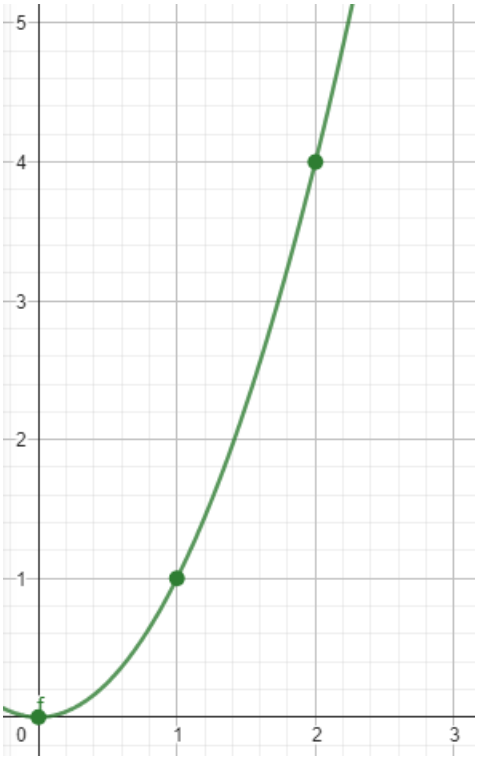

Figura 4: Gráfica de caso de la fig 3

4 Conclusiones

El algoritmo que evaluamos hoy *quick search Lomuto* tiene la curiosidad de funcionar mejor cuando el arreglo que se tiene esta de manera desordenada y funciona muy bien cuando tenemos muchos datos ya que la complejidad asintótica promedio sera linealmente de n sino en el pero caso tenemos n^2 como se puede observar en la fig 4 y esto no lo queremos.