INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

Análisis de Algoritmos

Sesión 4: Divide y Vencerás II

Edgar Adrián Nava Romo

Maestra: Sandra Díaz Santiago

Grupo: 3CM3

23 de Abril de 2020

Quicksort

Manual de Usuario

Para ejecutar la práctica es necesario tener instalado python 3 en cualquiera de sus versiones y en el Shell de la computadora escribir:

```
$ python3 quick.py
```

Al ejecutarse pide 3 números, el primero es para delimitar el arreglo, el segundo es para delimitar los números generados al azar por el lado inferior, el tercero para delimitar los números generados al azar por el lado superior.

Código Fuente

Las 3 partes del código funcionan parecido, cambiando en la función el pivote que se usará para poder comparar con los demás elementos, el contador sumará uno cada que se entre a la función de partition, mismo que va a regresar y se irá incrementando.

```
1.  def quicksort(quicktype, array, i, j, comparisonCount):
2.      if i >= j:
3.          return array
4.      if quicktype == 1:
5.          pivot, comps = partition(array, i, j, i) #First Pivot sending i
6.          comparisonCount = comparisonCount + comps
7.      if quicktype == 2:
8.          pivot, comps = partition(array, i, j, j) #Last Pivot sending j
9.          comparisonCount = comparisonCount + comps
10.     if quicktype == 3:
11.         pivot, comps = partition(array, i, j, random.randint(i, j))#Random
    Pivot sending a number between i, j
12.         comparisonCount = comparisonCount + comps
13.
14.     quicksort(quicktype, array, i, pivot - 1, comparisonCount)
15.     quicksort(quicktype, array, pivot + 1, j, comparisonCount)
16.     return comparisonCount
```

En la función partition es donde se irá partiendo el arreglo hasta que los valores se puedan comparar correctamente comparando A[i] < A[j]

```
1.  def partition(array, i, j, pivot):
2.      array[j], array[pivot] = array[pivot], array[j]
3.      stored_index = i
4.      comps = 0
5.      for i in range(i, j):
6.          if array[i] < array[j]:
7.              comps += 1
8.              array[stored_index], array[i] = array[i], array[stored_index]
9.              stored_index += 1
10.     array[j], array[stored_index] = array[stored_index], array[j]
11.     return stored_index, comps
```

```
[↳ python3 quick.py
Insert Limit of the Array:      10
Insert Inferior Limit:  20
Insert Superior Limit:  30
Original Array:  [25 28 29 25 20 20 21 27 26 29]

Sorted Array with First Pivot:  [20 20 21 25 25 26 27 28 29 29]
 3

Elapsed time:  0.00032799999999999496
Original Array:  [20 20 21 25 25 26 27 28 29 29]

Sorted Array with Last Pivot:  [20 20 21 25 25 26 27 28 29 29]
 8

Elapsed time:  0.00037400000000004097
Original Array:  [20 20 21 25 25 26 27 28 29 29]

Sorted Array with Random Pivot:  [20 20 21 25 25 26 27 28 29 29]
 5

Elapsed time:  0.00037200000000003897
```

```
[↳ python3 quick.py
Insert Limit of the Array:      10
Insert Inferior Limit:  1
Insert Superior Limit:  100
Original Array:  [38 13 73 10 76  6 80 65 17  2]

Sorted Array with First Pivot:  [ 2  6 10 13 17 38 65 73 76 80]
 5

Elapsed time:  0.00042700000000000107
Original Array:  [ 2  6 10 13 17 38 65 73 76 80]

Sorted Array with Last Pivot:  [ 2  6 10 13 17 38 65 73 76 80]
 9

Elapsed time:  0.00038599999999999746
Original Array:  [ 2  6 10 13 17 38 65 73 76 80]

Sorted Array with Random Pivot:  [ 2  6 10 13 17 38 65 73 76 80]
 8

Elapsed time:  0.00039099999999999747
```

```
↳ python3 quick.py
Insert Limit of the Array:      20
Insert Inferior Limit:  1
Insert Superior Limit:  100
Original Array:  [38 13 73 10 76  6 80 65 17  2 77 72  7 26 51 21 19 85 12 29]

Sorted Array with First Pivot:  [ 2  6  7 10 12 13 17 19 21 26 29 38 51 65 72 73 76 77 80 85]
 11

Elapsed time:  0.0006450000000000067
Original Array:  [ 2  6  7 10 12 13 17 19 21 26 29 38 51 65 72 73 76 77 80 85]

Sorted Array with Last Pivot:  [ 2  6  7 10 12 13 17 19 21 26 29 38 51 65 72 73 76 77 80 85]
 19

Elapsed time:  0.0010930000000000106
Original Array:  [ 2  6  7 10 12 13 17 19 21 26 29 38 51 65 72 73 76 77 80 85]

Sorted Array with Random Pivot:  [ 2  6  7 10 12 13 17 19 21 26 29 38 51 65 72 73 76 77 80 85]
 2

Elapsed time:  0.0006849999999999912
```

```
↳ python3 quick.py
Insert Limit of the Array:      20
Insert Inferior Limit:  100
Insert Superior Limit:  200
Original Array:  [137 112 172 109 175 105 179 164 116 101 176 171 106 125 150 120 118 184
 111 128]

Sorted Array with First Pivot:  [101 105 106 109 111 112 116 118 120 125 128 137 150 164 171 172 175 176
 179 184]
 11

Elapsed time:  0.0007199999999999984
Original Array:  [101 105 106 109 111 112 116 118 120 125 128 137 150 164 171 172 175 176
 179 184]

Sorted Array with Last Pivot:  [101 105 106 109 111 112 116 118 120 125 128 137 150 164 171 172 175 176
 179 184]
 19

Elapsed time:  0.0009319999999999884
Original Array:  [101 105 106 109 111 112 116 118 120 125 128 137 150 164 171 172 175 176
 179 184]

Sorted Array with Random Pivot:  [101 105 106 109 111 112 116 118 120 125 128 137 150 164 171 172 175 176
 179 184]
 19

Elapsed time:  0.0006510000000000127
```

```
↳ python3 quick.py
Insert Limit of the Array:      7
Insert Inferior Limit:  28
Insert Superior Limit:  97
Original Array:  [65 40 37 33 92 44 29]

Sorted Array with First Pivot:  [29 33 37 40 44 65 92]
 5

Elapsed time:  0.00041600000000002746
Original Array:  [29 33 37 40 44 65 92]

Sorted Array with Last Pivot:  [29 33 37 40 44 65 92]
 6

Elapsed time:  0.00037199999999998346
Original Array:  [29 33 37 40 44 65 92]

Sorted Array with Random Pivot:  [29 33 37 40 44 65 92]
 4

Elapsed time:  0.00038799999999999946
```

```
↳ python3 quick.py
Insert Limit of the Array:      30
Insert Inferior Limit:  1
Insert Superior Limit:  100
Original Array:  [38 13 73 10 76  6 80 65 17  2 77 72  7 26 51 21 19 85 12 29 30 15 51 69
 88 88 95 97 87 14]

Sorted Array with First Pivot:  [ 2  6  7 10 12 13 14 15 17 19 21 26 29 30 38 51 51 65 69 72 73 76 77 80
 85 87 88 88 95 97]
 14

Elapsed time:  0.0005269999999999997
Original Array:  [ 2  6  7 10 12 13 14 15 17 19 21 26 29 30 38 51 51 65 69 72 73 76 77 80
 85 87 88 88 95 97]

Sorted Array with Last Pivot:  [ 2  6  7 10 12 13 14 15 17 19 21 26 29 30 38 51 51 65 69 72 73 76 77 80
 85 87 88 88 95 97]
 29

Elapsed time:  0.0017579999999999818
Original Array:  [ 2  6  7 10 12 13 14 15 17 19 21 26 29 30 38 51 51 65 69 72 73 76 77 80
 85 87 88 88 95 97]

Sorted Array with Random Pivot:  [ 2  6  7 10 12 13 14 15 17 19 21 26 29 30 38 51 51 65 69 72 73 76 77 80
 85 87 88 88 95 97]
 9

Elapsed time:  0.0009069999999999911
```

```
↳ python3 quick.py
Insert Limit of the Array:      10
Insert Inferior Limit:  1
Insert Superior Limit:  100
Original Array:  [38 13 73 10 76  6 80 65 17  2]

Sorted Array with First Pivot:  [ 2  6 10 13 17 38 65 73 76 80]
 5

Elapsed time:  0.00035000000000001696
Original Array:  [ 2  6 10 13 17 38 65 73 76 80]

Sorted Array with Last Pivot:  [ 2  6 10 13 17 38 65 73 76 80]
 9

Elapsed time:  0.0005169999999999897
Original Array:  [ 2  6 10 13 17 38 65 73 76 80]

Sorted Array with Random Pivot:  [ 2  6 10 13 17 38 65 73 76 80]
 4

Elapsed time:  0.0004069999999999907
```

```
↳ python3 quick.py
Insert Limit of the Array:      40
Insert Inferior Limit:  1
Insert Superior Limit:  10
Original Array:  [6 9 6 1 1 2 8 7 3 5 6 3 5 3 5 8 8 2 8 1 7 8 7 2 1 2 9 9 4 9 8 4 7 6 2 4 5
 9 2 5]

Sorted Array with First Pivot:  [1 1 1 1 2 2 2 2 2 2 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 8 8 9 9
 9 9 9]
 21

Elapsed time:  0.0008479999999999599
Original Array:  [1 1 1 1 2 2 2 2 2 2 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 8 8 9 9
 9 9 9]

Sorted Array with Last Pivot:  [1 1 1 1 2 2 2 2 2 2 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 8 8 9 9
 9 9 9]
 35

Elapsed time:  0.0007710000000000217
Original Array:  [1 1 1 1 2 2 2 2 2 2 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 8 8 9 9
 9 9 9]

Sorted Array with Random Pivot:  [1 1 1 1 2 2 2 2 2 2 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 8 8 9 9
 9 9 9]
 16

Elapsed time:  0.0007570000000000077
```

```
↳ python3 quick.py
Insert Limit of the Array:      100
Insert Inferior Limit:  1
Insert Superior Limit:  100
Original Array:  [38 13 73 10 76  6 80 65 17  2 77 72  7 26 51 21 19 85 12 29 30 15 51 69
 88 88 95 97 87 14 10  8 64 62 23 58  2  1 61 82  9 89 14 48 73 31 72  4
 71 22 50 58  4 69 25 44 77 27 53 81 42 83 16 65 69 26 99 88  8 27 26 23
 10 68 24 28 38 58 84 39  9 33 35 11 24 16 88 26 72 93 75 63 47 33 89 24
 56 66 78  4]

Sorted Array with First Pivot:  [ 1  2  2  4  4  4  6  7  8  8  9  9 10 10 10 11 12 13 14 14 15 16 16 17
 19 21 22 23 23 24 24 24 25 26 26 26 26 27 27 28 29 30 31 33 33 35 38 38
 39 42 44 47 48 50 51 51 53 56 58 58 58 61 62 63 64 65 65 66 68 69 69 69
 71 72 72 72 73 73 75 76 77 77 78 80 81 82 83 84 85 87 88 88 88 88 89 89
 93 95 97 99]
 46

Elapsed time:  0.0017810000000000326
Original Array:  [ 1  2  2  4  4  4  6  7  8  8  9  9 10 10 10 11 12 13 14 14 15 16 16 17
 19 21 22 23 23 24 24 24 25 26 26 26 26 27 27 28 29 30 31 33 33 35 38 38
 39 42 44 47 48 50 51 51 53 56 58 58 58 61 62 63 64 65 65 66 68 69 69 69
 71 72 72 72 73 73 75 76 77 77 78 80 81 82 83 84 85 87 88 88 88 88 89 89
 93 95 97 99]

Sorted Array with Last Pivot:  [ 1  2  2  4  4  4  6  7  8  8  9  9 10 10 10 11 12 13 14 14 15 16 16 17
 19 21 22 23 23 24 24 24 25 26 26 26 26 27 27 28 29 30 31 33 33 35 38 38
 39 42 44 47 48 50 51 51 53 56 58 58 58 61 62 63 64 65 65 66 68 69 69 69
 71 72 72 72 73 73 75 76 77 77 78 80 81 82 83 84 85 87 88 88 88 88 89 89
 93 95 97 99]
 99

Elapsed time:  0.005927999999999989
Original Array:  [ 1  2  2  4  4  4  6  7  8  8  9  9 10 10 10 11 12 13 14 14 15 16 16 17
 19 21 22 23 23 24 24 24 25 26 26 26 26 27 27 28 29 30 31 33 33 35 38 38
 39 42 44 47 48 50 51 51 53 56 58 58 58 61 62 63 64 65 65 66 68 69 69 69
 71 72 72 72 73 73 75 76 77 77 78 80 81 82 83 84 85 87 88 88 88 88 89 89
 93 95 97 99]

Sorted Array with Random Pivot:  [ 1  2  2  4  4  4  6  7  8  8  9  9 10 10 10 11 12 13 14 14 15 16 16 17
 19 21 22 23 23 24 24 24 25 26 26 26 26 27 27 28 29 30 31 33 33 35 38 38
 39 42 44 47 48 50 51 51 53 56 58 58 58 61 62 63 64 65 65 66 68 69 69 69
 71 72 72 72 73 73 75 76 77 77 78 80 81 82 83 84 85 87 88 88 88 88 89 89
 93 95 97 99]
 54

Elapsed time:  0.0014279999999999848
```

```
↳ python3 quick.py
Insert Limit of the Array:      3
Insert Inferior Limit:  10
Insert Superior Limit:  100
Original Array:  [47 22 82]

Sorted Array with First Pivot:  [22 47 82]
 1

Elapsed time:  0.0003210000000000157
Original Array:  [22 47 82]

Sorted Array with Last Pivot:  [22 47 82]
 2

Elapsed time:  0.0003580000000002496
Original Array:  [22 47 82]

Sorted Array with Random Pivot:  [22 47 82]
 2

Elapsed time:  0.0002850000000000352
```

Conclusiones

En las imágenes previas se pudo observar que en el mismo arreglo hay veces que agarrando el pivote al azar hace el menor número de comparaciones, en otros daba un número de comparaciones cercano al menor, ya sea que el pivote final o pivote inicial tuvieran menos comparaciones, es importante resaltar que el tiempo usado usando un pivote al azar es muchas veces menor a usar el pivote inicial o el pivote final.

En conclusión, la diferencia de tiempos y comparaciones que se puede observar en los ejemplo, puede mejorar el algoritmo usando clases, más funciones o cambiar de lenguaje de programación para acortar tiempos, sin embargo para esta práctica es bastante útil y se puede observar que los tiempos no rebasan los .02 segundos, lo cuál es bastante bueno y se puede decir que se usa una complejidad O(n log n) en la implementación, sin embargo aunque la implementación en los 3 casos sea correcta, no siempre es mejor una que la otra, pueden variar dependiendo cómo estén ordenados al principio los datos del arreglo o el pivote elegido al azar.

Al implementar esta práctica se pudo observar que es bastante sencillo implementar este método de ordenamiento, y requiere muy pocos recursos a comparación de los otros métodos vistos y puede llegar a usarse en búsquedas binarias.

Bibliografía y Recursos Utilizados

Beach, J. (2020). PlanetB | Syntax Highlight Code in Word Documents. http://www.planetb.ca/syntax-highlight-word

Quicksort Algorithm - InterviewBit. (2020). https://www.interviewbit.com/tutorial/quicksort-algorithm/

Data Structure and Algorithms - Quick Sort - Tutorialspoint. (2020). https://www.tutorialspoint.com/data_structures_algorithms/quick_sort_algorithm.htm