Time Complexity:

$n = r - p$

Line	Instruction	# of execution times
	Partition(A,p,r)	
1	x = A[r]	1
2	i = p - 1	1
3	**for** j = p to r - 1	n+1
4	**if** A[j] ≤ x	n
5	i = i + 1	n
6	A[i] ↔ A[j]	n
7	A[i + 1] ↔ A[r]	1
8	**return** i + 1	1

Worst, Average & Best:

$T(n) = 1 + 1 + (n+1) + n + n + n + 1 + 1 = 5 + 4n = O(n)$

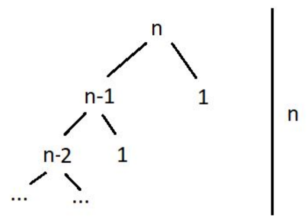
$O(n)$, $\Omega(1)$, $\Theta(n)$

Line	Instruction	# of execution times
	QuickSort(A,p,r)	
1	**if** p < r	1
2	q = Partition(A,p,r)	O(n)
3	QuickSort(A,p,q-1)	T(n/2)
4	QuickSort(A,q+1,r)	T(n/2)

Worst:

$T(n) = 1 + O(n) + T(n-1) + T(1) \simeq O(n) + T(n-1) + T(1)$ 1

$n - i2 = 1 \to i = 2(n-1) = 2n - 2 \simeq n$



$\sum_{i=0}^{n} n = O(n^2)$

$O(n^2)$

Average & Best:

$T(n) = 1 + O(n) + 2T(n/2) \simeq O(n) + 2T(n/2)$

by the Master method:

$T(1) \rightarrow \Theta(1),\ a = 2,\ b = 2,\ a = b^1 \ is\ true$

$T(n) = \Theta(n^1 log\ n) = \Theta(n\ log\ n)$

$\Theta(n\ log\ n)\ ,\ \Omega(n\ log\ n)$

Line	Instruction	# of execution times
	Rand-Parti(A,p,r)	
1	i = Random(p,r)	1
2	A[r] ↔ A[i]	1
3	**return** Partition(A,p,r)	O(n)

Worst, Average & Best:

$T(n) = 1 + 1 + O(n) = O(n)$

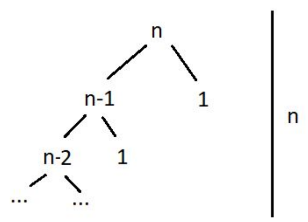
$O(n)\ ,\Omega(n)\ ,\Theta(n)$

Line	Instruction	# of execution times
	Randomized-QS(A,p,r)	
1	**if** p < r	1
2	q = Rand-Parti(A,p,r)	O(n)
3	Randomized-QS(A,p,q-1)	T(n/2)
4	Randomized-QS(A,q+1,r)	T(n/2)

Worst:

$T(n) = 1 + O(n) + T(n-1) + T(1) \simeq O(n) + T(n-1) + T(1)$

$n - i2 = 1 \longrightarrow i = 2(n-1) = 2n - 2 \simeq n$



$\sum_{i=0}^{n} n = O(n^2)$

$O(n^2)$

Average & Best:

$T(n) = 1 + O(n) + 2T(n/2) \simeq O(n) + 2T(n/2)$

by the Master method:

$T(1) \to \Theta(1),\ a = 2,\ b = 2,\ a = b^1 \ is\ true$

$T(n) = \Theta(n^1 log\ n) = \Theta(n\ log\ n)$

$\Theta(n\ log\ n)$, $\Omega(n\ log\ n)$

Theoretical treatments:

Prueba	Variante	Estado	Tamaño (n)	Tiempo (ms)
1			10	$O(n\ log\ n)$
2			100	$O(n\ log\ n)$
3			1000	$O(n\ log\ n)$
4		Ascending	10000	$O(n\ log\ n)$
6			10	$O(n^2)$
7			100	$O(n^2)$
8			1000	$O(n^2)$
9		Descending	10000	$O(n^2)$
11			10	$\Theta(n\ log\ n)$
12			100	$\Theta(n\ log\ n)$
13			1000	$\Theta(n\ log\ n)$
14	Normal	Random	10000	$\Theta(n\ log\ n)$
16			10	$\Theta(n\ log\ n)$
17			100	$\Theta(n\ log\ n)$
18			1000	$\Theta(n\ log\ n)$
19		Ascending	10000	$\Theta(n\ log\ n)$
21			10	$\Theta(n\ log\ n)$
22			100	$\Theta(n\ log\ n)$
23			1000	$\Theta(n\ log\ n)$
24		Descending	10000	$\Theta(n\ log\ n)$
26	Random	Random	10	$\Theta(n\ log\ n)$

27			100	$\Theta(n \log n)$
28			1000	$\Theta(n \log n)$
29			10000	$\Theta(n \log n)$

Experimental Unit:
-Quicksort Algorithm

Response Values:
-Execution time of the QuickSort method

Experimental Factors:

- **Studied:**

- Array status.
- Array size.
- Algorithm variant.

- **Not studied:**

-Number of programs being executed
-RAM capacity
-Programing Language

Observational Factors:
-Program execution in the computer
-Compiler performance and optimization
-The general condition and health of the programmers

Factor Levels:

- **Variant:** Normal, Random
- **Status:** Ascending, Descending, Random.
- **Size:** 10^1 , 10^2 , 10^3 , 10^4

Treatment:

Treatment	Variant	Status	Size(n)	Time (ms)
1			10	
2			100	
3			1000	
4		Ascending	10000	
6			10	
7			100	
8			1000	
9		Descending	10000	
11	Normal	Random	10	

12			100	
13			1000	
14			10000	
16			10	
17			100	
18			1000	
19		Ascending	10000	
21			10	
22			100	
23			1000	
24		Descending	10000	
26			10	
27			100	
28			1000	
29	Random	Random	10000	

1000 repetitions per treatment.

1.b. Hitherto the stages of study and experiment design we have completed are planning and realization. The stages that we are missing are analysis, interpretation, and control & final conclusions.

1.c. The program objective is to compare two or more treatments, since in this experiment we created many different treatments to see how the response variables change and to analyze the results and the variance using ANOVA. All of this because we want to understand the behavior of the QuickSort algorithm and we want to draw a valid conclusion.

1.d. Analysis:

<u>Ascending arrays:</u>

In the case of ascending arrays we have the following null hypothesis, alternate hypothesis and alpha value. (Group 1 is the QuickSort algorithm and Group 2 is the Randomized QuickSort algorithm)

$H_0 : \mu_1 = \mu_2$

$H_a : \mu_1 > \mu_2$

$\alpha = 0.05$

This is the resulting ANOVA table:

Variation Source	Sum of Squares	Degrees of Freedom	S.S. Average	F	P-Value	F Crit
Between Groups	567842,0792	1	567842,0792	6,07694	0,014538618	3,888374717
Within Groups	18688422,26	200	93442,11129			
Total	19256264,34	201				

Since the P-Value is 0.015, it is lower than alpha. Therefore we must reject the null hypothesis in this particular case.

Descending arrays:

In the case of descending arrays we have the following null hypothesis, alternate hypothesis and alpha value. (Group 1 is the QuickSort algorithm and Group 2 is the Randomized QuickSort algorithm)

$H_0 : \mu_1 = \mu_2$

$H_a : \mu_1 > \mu_2$

$\alpha = 0.05$

This is the resulting ANOVA table:

Variation Source	Sum of Squares	Degrees of Freedom	S.S. Average	F	P-Value	F Crit
Between Groups	113395,0297	1	113395,0297	76,21889	1,00616E-15	3,888374717
Within Groups	297550,9901	200	1487,75495			
Total	410946,0198	201				

Since the P-Value is $1,006 \times 10^{-15}$, it is lower than alpha. Therefore we must reject the null hypothesis in this particular case.

Random arrays:

In the case of random arrays we have the following null hypothesis, alternate hypothesis and alpha value. (Group 1 is the QuickSort algorithm and Group 2 is the Randomized QuickSort algorithm)

$H_0 : \mu_1 = \mu_2$

$H_a : \mu_1 > \mu_2$

$\alpha = 0.05$

This is the resulting ANOVA table:

Variation Source	Sum of Squares	Degrees of Freedom	S.S. Average	F	P-Value	F Crit
Between Groups	227834,9307	1	227834,9307	23,82808	2,14957E-06	3,888374717
Within Groups	1912323,168	200	9561,615842			
Total	2140158,099	201				

Since the P-Value is $2,150 \times 10^{-6}$, it is lower than alpha. Therefore we must reject the null hypothesis in this particular case.

Conclusion:

After aggregating all the data and performing an ANOVA analysis, we can conclude that the randomized QuickSort algorithm takes less time than the QuickSort algorithm.