Diseño de pruebas unitarias

|  |  |
| --- | --- |
| Caso de Prueba | # 1 |
| Algoritmo(s) | QuickSort()  RandomizeQS() |
| Objetivo | Probar el ordenamiento que debe realizar el algoritmo en arreglos pequeños |
| Arreglo de entrada | Un arreglo {10,9,8,7,6,5,4,3,2,1} con números enteros positivos |
| Arreglo de salida | Un arreglo {1,2,3,4,5,6,7,8,9,10} con números enteros positivos |

|  |  |
| --- | --- |
| Caso de Prueba | # 2 |
| Algoritmo(s) | Partition()  Rand-Parti() |
| Objetivo | Probar el ordenamiento que debe realizar el algoritmo en arreglos pequeños |
| Arreglo de entrada | Un arreglo {10,9,8,7,6,5,4,3,2,1} con números enteros positivos |
| Arreglo de salida | Un arreglo {1,2,3,4,5,6,7,8,9,10} con números enteros positivos |

|  |  |
| --- | --- |
| Caso de Prueba | # 3 |
| Algoritmo(s) | QuickSort()  RandomizeQS() |
| Objetivo | Probar el ordenamiento que debe realizar el algoritmo en arreglos grandes |
| Arreglo de entrada | Un arreglo {num1000,num999,….,num1} con números enteros positivos |
| Arreglo de salida | Un arreglo {result1, result2, result3,….. result1000} con números enteros positivos donde se cumple la condición que result sea menor o igual a result (n+1) |

|  |  |
| --- | --- |
| Caso de Prueba | # 4 |
| Algoritmo(s) | Partition()  Rand-Parti() |
| Objetivo | Probar el ordenamiento que debe realizar el algoritmo en arreglos grandes |
| Arreglo de entrada | Un arreglo {num1000,num999,….,num1} con números enteros positivos |
| Arreglo de salida | Un arreglo {result1, result2, result3,….. result1000} con números enteros positivos donde se cumple la condición que result sea menor o igual a result (n+1) |

|  |  |
| --- | --- |
| Caso de Prueba | # 5 |
| Algoritmo(s) | QuickSort()  RandomizeQS() |
| Objetivo | Probar el ordenamiento que debe realizar el algoritmo en arreglos aleatorios |
| Arreglo de entrada | Un arreglo {num1,num2,num3,…,num1000} donde num es un entero positivo aleatorio entre 1 y 10^9. |
| Arreglo de salida | Un arreglo {resul1,result2,result3,…..,resutl1000} donde result es un entero positivo aleatorio entre 1 y 10^9 cumpliéndose la condición que result en la posición n sea menor o igual a result en la posición (n+1) |

|  |  |
| --- | --- |
| Caso de Prueba | # 6 |
| Algoritmo(s) | Partition()  Rand-Parti() |
| Objetivo | Probar el ordenamiento que debe realizar el algoritmo en arreglos aleatorios |
| Arreglo de entrada | Un arreglo {num1,num2,num3,…,num1000} donde num es un entero positivo aleatorio entre 1 y 10^9. |
| Arreglo de salida | Un arreglo {resul1,result2,result3,…..,resutl1000} donde result es un entero positivo aleatorio entre 1 y 10^9 cumpliéndose la condición que result en la posición n sea menor o igual a result en la posición (n+1) |