DISEÑO DE PRUEBAS UNITARIAS

Diseño de pruebas para la clase CombSort

Escenario	1 (entradas pequeñas)
Nombre	combSortRandom1Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 10 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 30, 12, 22, 10, 26, 35, 8, 18, 40, 15}
Resultado esperado	arr = { 8, 10, 12, 15, 18, 22, 26, 30, 35, 40}

Escenario	2 (entradas pequeñas)
Nombre	combSortRandom2Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 50 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = {19, 92, 82, 81, 9, 98, 12, 34, 87, 11, 51, 53, 43, 91, 74, 78, 62, 90, 18, 58, 6, 1, 100, 80, 70, 32, 130, 27, 10, 5, 65, 8, 19, 40, 49, 2, 28, 37, 89, 14, 48, 31, 77, 33, 25, 66, 39, 21, 108, 102}
Resultado esperado	arr = {1, 2, 5, 6, 8, 9, 10, 11, 12, 14, 18, 19, 21, 25, 27, 28, 31, 32, 33, 34, 37, 39, 40, 43, 48, 49, 51, 53, 58, 62, 65, 66, 70, 74, 77, 78, 80, 81, 82, 87, 89, 90, 91, 92, 100, 102, 108, 130}

Escenario	3 (entradas medianas)
Nombre	combSortRandom3Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 160000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 160000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr incial ordenado}

Escenario	4 (entradas grandes)
Nombre	combSortRandom4Test()
Método a probar	sort()

Descripción	Se tiene un arreglo de tamaño n = 800000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 800000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Escenario	5 (entradas grandes)
Nombre	combSortRandom4Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 2000000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 2000000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Escenario	6 (entradas pequeñas, orden inverso)
Nombre	combSortInverse1Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 10 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 100, 90, 80, 70, 60, 50, 40, 30, 20, 10,}
Resultado esperado	arr = {10,20,30,40,50,60,70,80,90,100}

Escenario	7 (entradas pequeñas, orden inverso)
Nombre	combSortInverse2Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 800000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 800000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Diseño de pruebas para la clase HeapSort

Escenario	1 (entradas pequeñas)
Nombre	heapSortRandom1Test()

Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 10 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 30, 12, 22, 10, 26, 35, 8, 18, 40, 15}
Resultado esperado	arr = { 8, 10, 12, 15, 18, 22, 26, 30, 35, 40}

Escenario	2 (entradas pequeñas)
Nombre	heapSortRandom2Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 50 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = {19, 92, 82, 81, 9, 98, 12, 34, 87, 11, 51, 53, 43, 91, 74, 78, 62, 90, 18, 58, 6, 1, 100, 80, 70, 32, 130, 27, 10, 5, 65, 8, 19, 40, 49, 2, 28, 37, 89, 14, 48, 31, 77, 33, 25, 66, 39, 21, 108, 102}
Resultado esperado	arr = {1, 2, 5, 6, 8, 9, 10, 11, 12, 14, 18, 19, 21, 25, 27, 28, 31, 32, 33, 34, 37, 39, 40, 43, 48, 49, 51, 53, 58, 62, 65, 66, 70, 74, 77, 78, 80, 81, 82, 87, 89, 90, 91, 92, 100, 102, 108, 130}

Escenario	3 (entradas medianas)
Nombre	heapSortRandom3Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 160000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 160000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr incial ordenado}

Escenario	4 (entradas grandes)
Nombre	heapSortRandom4Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 800000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 800000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Escenario	5 (entradas grandes)
Nombre	heapSortRandom4Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 2000000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 2000000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Escenario	6 (entradas pequeñas, orden inverso)
Nombre	heapSortInverse1Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 10 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 100, 90, 80, 70, 60, 50, 40, 30, 20, 10,}
Resultado esperado	arr = {10,20,30,40,50,60,70,80,90,100}

Escenario	7 (entradas pequeñas, orden inverso)
Nombre	heapSortInverse2Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 800000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 800000 valores en orden aleatorio }
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Diseño de pruebas para la clase MergeSort

Escenario	1 (entradas pequeñas)
Nombre	mergeSortRandom1Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 10 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 30, 12, 22, 10, 26, 35, 8, 18, 40, 15} i inicial = 0

	i final = 9
Resultado esperado	arr = { 8, 10, 12, 15, 18, 22, 26, 30, 35, 40}

Escenario	2 (entradas pequeñas)
Nombre	mergeSortRandom2Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 50 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = {19, 92, 82, 81, 9, 98, 12, 34, 87, 11, 51, 53, 43, 91, 74, 78, 62, 90, 18, 58, 6, 1, 100, 80, 70, 32, 130, 27, 10, 5, 65, 8, 19, 40, 49, 2, 28, 37, 89, 14, 48, 31, 77, 33, 25, 66, 39, 21, 108, 102} i inicial = 0 i final = 49
Resultado esperado	arr = {1, 2, 5, 6, 8, 9, 10, 11, 12, 14, 18, 19, 21, 25, 27, 28, 31, 32, 33, 34, 37, 39, 40, 43, 48, 49, 51, 53, 58, 62, 65, 66, 70, 74, 77, 78, 80, 81, 82, 87, 89, 90, 91, 92, 100, 102, 108, 130}

Escenario	3 (entradas medianas)
Nombre	mergeSortRandom3Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 160000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 160000 valores en orden aleatorio } i inicial = 0 i final = 159999
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Escenario	4 (entradas grandes)
Nombre	mergeSortRandom4Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 800000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 800000 valores en orden aleatorio } i inicial = 0 i final = 799999
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Escenario	5 (entradas grandes)

Nombre	mergeSortRandom4Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 2000000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 2000000 valores en orden aleatorio } i inicial = 0 i final = 1999999
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}

Escenario	6 (entradas pequeñas, orden inverso)
Nombre	mergeSortInverse1Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 10 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 100, 90, 80, 70, 60, 50, 40, 30, 20, 10,} i inicial = 0 i final = 9
Resultado esperado	arr = {10,20,30,40,50,60,70,80,90,100}

Escenario	7 (entradas pequeñas, orden inverso)
Nombre	mergeSortInverse2Test()
Método a probar	sort()
Descripción	Se tiene un arreglo de tamaño n = 800000 que contiene valores generados aleatoriamente y se llama al método sort() para que se ordene el arreglo, de menor a mayor.
Valores de entrada	arr = { 800000 valores en orden aleatorio } i inicial = 0 i final = 799999
Resultado esperado	Arreglo ordenado de menor a mayor arr = {arr inicial ordenado}