Sorting_Algos

January 2, 2025

[1]:
```scala
def swap(arr: Array[Int], i: Int, j: Int): Unit = {
  val temp = arr(i)
  arr(i) = arr(j)
  arr(j) = temp
}

def bubbleSort(arr: Array[Int]): Array[Int] = {
  val n = arr.length
  for (i <- 0 until n - 1) {
    var swapped = false
    for (j <- 0 until n - i - 1) {
      if (arr(j) > arr(j + 1)) {
        swap(arr, j, j + 1)
        swapped = true
      }
    }
    if (!swapped) return arr
  }
  arr
}

val bubbleResult = bubbleSort(Array(9, 7, 5, 3, 1))
println("Bubble Sorted Output:")
bubbleResult.foreach(x => print(s"$x "))
```

```
Bubble Sorted Output:
1 3 5 7 9

bubbleResult = Array(1, 3, 5, 7, 9)


swap: (arr: Array[Int], i: Int, j: Int)Unit
bubbleSort: (arr: Array[Int])Array[Int]
```

[1]:
```
Array(1, 3, 5, 7, 9)
```

[2]:
```scala
def insertionSort(arr: Array[Int]): Array[Int] = {
  for (i <- 1 until arr.length) {
    val key = arr(i)
    var j = i - 1
    while (j >= 0 && arr(j) > key) {
      arr(j + 1) = arr(j)
      j -= 1
    }
    arr(j + 1) = key
  }
  arr
}

val insertionResult = insertionSort(Array(8, 4, 2, 6, 9))
println("Insertion Sorted Output:")
insertionResult.foreach(x => print(s"$x "))
```

```
Insertion Sorted Output:
2 4 6 8 9

insertionResult = Array(2, 4, 6, 8, 9)


insertionSort: (arr: Array[Int])Array[Int]
```

[2]:
```
Array(2, 4, 6, 8, 9)
```

[3]:
```
def quickSort(arr: Array[Int]): Array[Int] = {
  if (arr.length <= 1) return arr
  val pivot = arr(arr.length / 2)
  quickSort(arr.filter(_ < pivot)) ++ arr.filter(_ == pivot) ++ quickSort(arr.
 ↪filter(_ > pivot))
}

val quickResult = quickSort(Array(10, 7, 8, 9, 1))
println("Quick Sorted Output:")
quickResult.foreach(x => print(s"$x "))
```

```
Quick Sorted Output:
1 7 8 9 10

quickResult = Array(1, 7, 8, 9, 10)


quickSort: (arr: Array[Int])Array[Int]
```

[3]:
```
Array(1, 7, 8, 9, 10)
```

[4]:
```
def heapify(arr: Array[Int], n: Int, i: Int): Unit = {
  var largest = i
  val left = 2 * i + 1
  val right = 2 * i + 2

  if (left < n && arr(left) > arr(largest)) largest = left
  if (right < n && arr(right) > arr(largest)) largest = right

  if (largest != i) {
    swap(arr, i, largest)
    heapify(arr, n, largest)
  }
}

def heapSort(arr: Array[Int]): Array[Int] = {
  val n = arr.length
  for (i <- n / 2 - 1 to 0 by -1) heapify(arr, n, i)
  for (i <- n - 1 to 0 by -1) {
    swap(arr, 0, i)
    heapify(arr, i, 0)
  }
  arr
}

val heapResult = heapSort(Array(3, 1, 4, 1, 5, 9))
println("Heap Sorted Output:")
heapResult.foreach(x => print(s"$x "))
```

```
Heap Sorted Output:
1 1 3 4 5 9
```

```
heapResult = Array(1, 1, 3, 4, 5, 9)


heapify: (arr: Array[Int], n: Int, i: Int)Unit
heapSort: (arr: Array[Int])Array[Int]
```

```
[4]: Array(1, 1, 3, 4, 5, 9)
```

```
[5]: def selectionSort(arr: Array[Int]): Array[Int] = {
       for (i <- arr.indices) {
         var minIdx = i
         for (j <- i + 1 until arr.length) {
           if (arr(j) < arr(minIdx)) minIdx = j
         }
         swap(arr, i, minIdx)
       }
       arr
     }
```

```
val selectionResult = selectionSort(Array(6, 3, 8, 5, 2))
println("Selection Sorted Output:")
selectionResult.foreach(x => print(s"$x "))
```

```
Selection Sorted Output:
2 3 5 6 8

selectionResult = Array(2, 3, 5, 6, 8)


selectionSort: (arr: Array[Int])Array[Int]
```

[5]:
```
Array(2, 3, 5, 6, 8)
```

[6]:
```
def printArray(arr: Array[Int], label: String): Unit = {
  println(s"$label: ${arr.mkString(", ")}")
}

val testArray = Array(5, 4, 3, 2, 1)
printArray(testArray, "Unsorted Array")
```

```
Unsorted Array: 5, 4, 3, 2, 1

testArray = Array(5, 4, 3, 2, 1)


printArray: (arr: Array[Int], label: String)Unit
```

[6]:
```
Array(5, 4, 3, 2, 1)
```

[ ]:
```

```

[ ]:
```

```