<https://github.com/eugenp/tutorials/tree/master/algorithms-sorting/src/main/java/com/baeldung/algorithms>

Counting Sort

**import** java.util.Arrays;

**public** **class** CountSortDemo {

**public** **static** **void** main(String[] args) {

CountSortDemo obj = **new** CountSortDemo();

**int**[] arr = { 2, 1, 1, 0, 2, 5, 4, 0, 2, 8, 7, 7, 9, 2, 0, 1, 9 };

System.***out***.println(Arrays.*toString*(arr));

**int**[] result = obj.countSort(arr);

System.***out***.println(Arrays.*toString*(result));

}

**public** **int**[] countSort(**int**[] arr) {

// k is the max element in arr

// range is from min to max

**int** max = Integer.***MIN\_VALUE***;

**int** min = Integer.***MAX\_VALUE***;

**int** n = arr.length;

**for** (**int** i = 0; i < n; i++) {

**if** (arr[i] > max) {

max = arr[i];

}

**if** (arr[i] < min) {

min = arr[i];

}

}

**int** range = max - min + 1;

**int**[] count = **new** **int**[range];

**int**[] result = **new** **int**[arr.length];

**for** (**int** i = 0; i < arr.length; i++) {

count[arr[i] - min]++;

}

**for** (**int** i = 1; i < count.length; i++) {

count[i] = count[i] + count[i - 1];

}

**for** (**int** i = arr.length - 1; i >= 0; i--) {

result[count[arr[i] - min] - 1] = arr[i];

count[arr[i] - min]--;

}

**for** (**int** i = 0; i < arr.length; i++) {

arr[i] = result[i];

}

**return** arr;

}

}

Merge Sort

**package** test;

**import** java.util.Arrays;

**public** **class** MergeSortDemo {

**public** **static** **void** main(String[] args) {

MergeSortDemo obj = **new** MergeSortDemo();

**int**[] arr = { 5, 1, 6, 2, 3, 4 };

**int** n = arr.length;

System.***out***.println(Arrays.*toString*(arr));

obj.mergeSort(arr, n);

System.***out***.println(Arrays.*toString*(arr));

}

**public** **void** mergeSort(**int**[] a, **int** n) {

**if** (n < 2)

**return**;

**int** mid = n / 2;

**int**[] l = **new** **int**[mid];

**int**[] r = **new** **int**[n - mid];

**for** (**int** i = 0; i < mid; i++) {

l[i] = a[i];

}

**for** (**int** i = mid; i < n; i++) {

r[i - mid] = a[i];

}

mergeSort(l, mid);

mergeSort(r, n - mid);

merge(a, l, r, mid, n - mid);

}

**public** **void** merge(**int**[] a, **int**[] l, **int**[] r, **int** left, **int** right) {

**int** i = 0, j = 0, k = 0;

**while** (i < left && j < right) {

**if** (l[i] <= r[j])

a[k++] = l[i++];

**else**

a[k++] = r[j++];

}

**while** (i < left)

a[k++] = l[i++];

**while** (j < right)

a[k++] = r[j++];

}

}

Heap Sort

Quick Sort

Bubble Sort Iterative Approach

**package** test;

**import** java.util.Arrays;

**public** **class** BubbleSortIterative {

**public** **static** **void** main(String[] args) {

BubbleSortIterative obj = **new** BubbleSortIterative();

**int**[] arr = { 29, 10, 14, 37, 13 };

**int** n = arr.length;

System.***out***.println(Arrays.*toString*(arr));

**int**[] result = obj.bubbleSortIterativeApproach(arr, n);

System.***out***.println(Arrays.*toString*(result));

}

// Complexity : n^2

**public** **int**[] bubbleSortIterativeApproach(**int**[] arr, **int** n) {

**for** (**int** i = 0; i < n - 1; i++) {

**for** (**int** j = 0; j < n - 1; j++) {

**if** (arr[j] > arr[j + 1]) {

**int** temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

**return** arr;

}

}

Bubble Sort Improved Approach

**package** test;

**import** java.util.Arrays;

**public** **class** BubbleSortImproved {

**public** **static** **void** main(String[] args) {

BubbleSortImproved obj = **new** BubbleSortImproved();

**int**[] arr = { 29, 10, 14, 37, 13 };

**int** n = arr.length;

System.***out***.println(Arrays.*toString*(arr));

**int**[] result = obj.bubbleSortImproved(arr,n);

System.***out***.println(Arrays.*toString*(result));

}

// Complexity :

// If array is sorted do not enter

// Remove i elements - i

**public** **int**[] bubbleSortImproved(**int**[] arr, **int** n) {

**boolean** sortedFlag = **false**;

**for** (**int** i = 0; i < n - 1; i++) {

**for** (**int** j = 0; j < n - 1 - i; j++) {

**if** (arr[j] > arr[j + 1]) {

**int** temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

sortedFlag = **true**;

}

}

**if** (sortedFlag == **false**) {

**break**;

}

}

**return** arr;

}

}

Bubble Sort Recursive Approach

**package** test;

**import** java.util.Arrays;

**public** **class** BubbleSortRecursive {

**public** **static** **void** main(String[] args) {

BubbleSortRecursive obj = **new** BubbleSortRecursive();

**int**[] arr = { 29, 10, 14, 37, 13 };

**int** n = arr.length;

System.***out***.println(Arrays.*toString*(arr));

obj.bubbleSortRecursive(arr, n);

System.***out***.println(Arrays.*toString*(arr));

}

**public** **void** bubbleSortRecursive(**int**[] arr, **int** n) {

**if** (n == 1) {

**return**;

}

**for** (**int** i = 0; i < n - 1; i++) {

**if** (arr[i] > arr[i + 1]) {

**int** temp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = temp;

}

bubbleSortRecursive(arr, n - 1);

}

}

}

Insertion Sort Iterative Approach

Insertion Sort Recursive Approach

Selection Sort Iterative Approach

Selection Sort Recursive Approach