

MergeSort(o, v.length, v)

p = 0, n = 6, q = 3

MergeSort(p, q, v)

p = 0, n = 3, q = 1

MergeSort(p, q, v)

p = 0, n = 1

return

MergeSort(q, n, v)

p = 1, n = 3, q = 2

MergeSort(p, q, v)

p = 1, n = 2

return

MergeSort(q, n, v)

p = 2, n = 3

return

return

return

MergeSort(q, n, v)

p = 3, n = 6, q = 4

MergeSort(p, q, v)

p = 3, n = 4

return

MergeSort(q, n, v)

p = 3, n = 6, q = 5

MergeSort(p, q, v)

p = 4, n = 5

return

MergeSort(q, n, v)

p = 5, n = 6

return

return

return

return

int[] v = {5,8,2,1,7,4};

```

public static void mergeSort(int p, int n, int[] v){
    if (p < n - 1) {
        int q = (p + n) / 2;

        mergeSort(p, q, v);
        mergeSort(q, n, v);
        intercala(p, q, n, v);
    }
}

```