mergeSort(0, v.length, v)

p = 0, n = 6, v = {5, 8, 2, 1, 7, 4}, q = 3

mergeSort(p, q, v)

p = 0, n = 3, v = {5, 8, 2, 1, 7, 4}, q = 1

mergeSort(p, q, v)

p = 0, n = 1, v = {5, 8, 2, 1, 7, 4}, q = 0

intercala(p, q, n, v)

v = {**5**, 8, 2, 1, 7, 4}

mergeSort(q, n, v)

p = 1, n = 3, v = {5, 8, 2, 1, 7, 4}, q = 2

intercala(p, q, n, v)

v = {5, **2**, **8**, 1, 7, 4}

intercala(p, q, n, v)

v = {**2**, **5**, **8**, 1, 7, 4}

mergeSort(q, n, v)

p = 3, n = 6, v = {2, 5, 8, 1, 7, 4}, q = 4

mergeSort(p, q, v)

p = 3, n = 4, v = {2, 5, 8, 1, 7, 4}, q = 3

intercala(p, q, n, v)

v = {2, 5, 8, **1**, 7, 4}

mergeSort(q, n, v)

p = 4, n = 6, v = {2, 5, 8, 1, 7, 4}, q = 5

intercala(p, q, n, v)

v = {2, 5, 8, 1, **4**, **7**}

intercala(p, q, n, v)

v = {2, 5, 8, **1**, **4**, **7**}

intercala(p, q, n, v)

v = {**1**, **2**, **4**, **5**, **7**, **8**}

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

```
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);
        }
}
```