

```
quickSort(v, 0, v.length - 1)
```

```
p = 0, r = 5, pivo = 3
```

```
quickSort(v, p, pivo - 1)
```

```
p = 0, r = 2, pivo = 0
```

```
quickSort(v, p, pivo - 1)
```

```
p = 0, r = -1
```

```
return
```

```
quickSort(v, pivo + 1, r)
```

```
p = 1, r = 2, pivo = 2
```

```
quickSort(v, p, pivo - 1)
```

```
p = 1, r = 1
```

```
return
```

```
quickSort(v, pivo + 1, r)
```

```
p = 3, r = 2
```

```
return
```

```
return
```

```
return
```

```
quickSort(v, pivo + 1, r)
```

```
p = 4, r = 5, pivo = 4
```

```
quickSort(v, p, pivo - 1)
```

```
p = 4, r = 3
```

```
return
```

```
quickSort(v, pivo + 1, r)
```

```
p = 5, r = 5
```

```
return
```

```
return
```

```
return
```

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

```
public static void quickSort(int[] v, int p, int r) {
    if (p < r) {
        int pivo = particao(v, p, r);
        quickSort(v, p, pivo - 1);
        quickSort(v, pivo + 1, r);
    }
}
```