CS 2050
Computer Science II

Thyago Mota

Metropolitan State University of Denver

Lives Transformed

Agenda

- Sorting Algorithms:
  - Merge Sort

Merge Sort

- Merge sort uses an algorithmic strategy called divide-and-conquer
- It splits the input collection into two halves and recursively calls itself on each of the splitted subcollections

Merge Sort

- The base case of the recursion is reached when the collection consists of only one element
- The important part of the algorithm happens when the recursions returns
- The algorithm merges the two sorted subcollections back to one again

Merge Sort

```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```

Merge Sort

```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```

Merge Sort

```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

 [13, 12, 84, 79, 10, 77]  [56, 1, 34, 27, 3]
```

Merge Sort

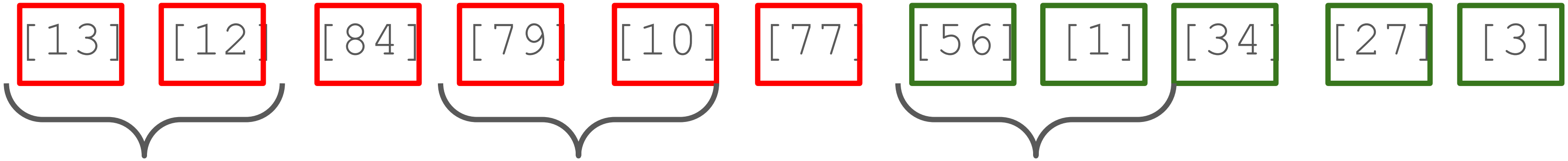
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

[13, 12, 84, 79, 10, 77] [56, 1, 34, 27, 3]

[13, 12, 84] [79, 10, 77] [56, 1, 34] [27, 3]
```

Merge Sort

```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

[13, 12, 84, 79, 10, 77] [56, 1, 34, 27, 3]

[13, 12, 84] [79, 10, 77] [56, 1, 34] [27, 3]

[13, 12] [84] [79, 10] [77] [56, 1] [34] [27] [3]
```

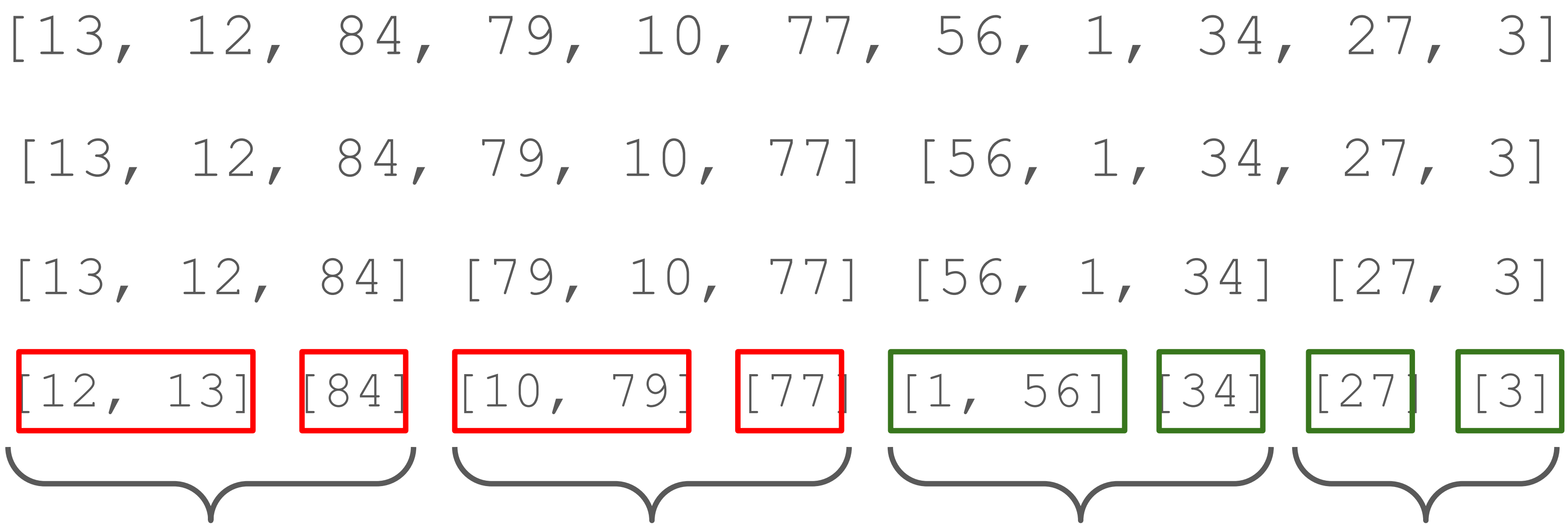
Merge Sort

```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

[13, 12, 84, 79, 10, 77] [56, 1, 34, 27, 3]

[13, 12, 84] [79, 10, 77] [56, 1, 34] [27, 3]

[13, 12] [84] [79, 10] [77] [56, 1] [34] [27] [3]
```



```
[13] [12] [84] [79] [10] [77] [56] [1] [34] [27] [3]
```

Merge Sort



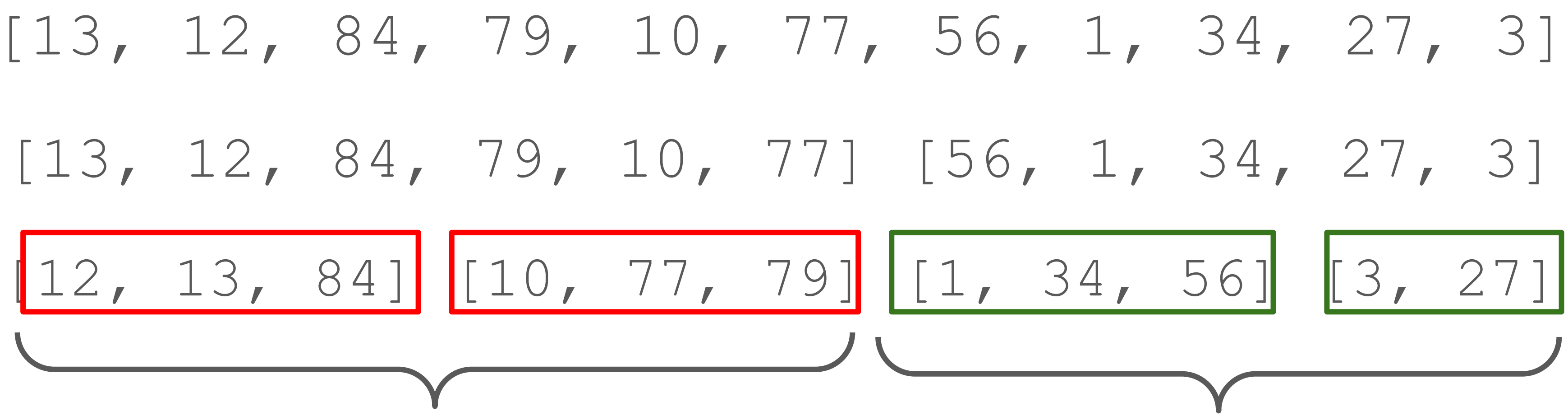
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

[13, 12, 84, 79, 10, 77] [56, 1, 34, 27, 3]

[13, 12, 84] [79, 10, 77] [56, 1, 34] [27, 3]

[12, 13] [84] [10, 79] [77] [1, 56] [34] [27] [3]

Merge Sort



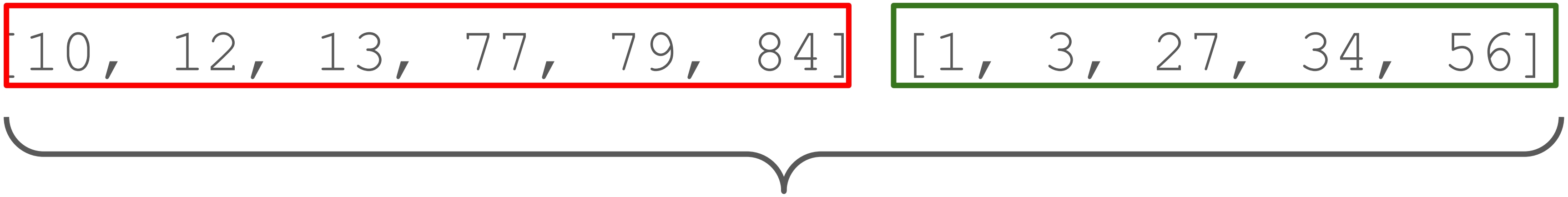
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

[13, 12, 84, 79, 10, 77] [56, 1, 34, 27, 3]

[12, 13, 84] [10, 77, 79] [1, 34, 56] [3, 27]

Merge Sort

[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[10, 12, 13, 77, 79, 84]

[1, 3, 27, 34, 56]

Merge Sort

```
[1, 3, 10, 12, 13, 27, 34, 56, 77, 79, 84]
```

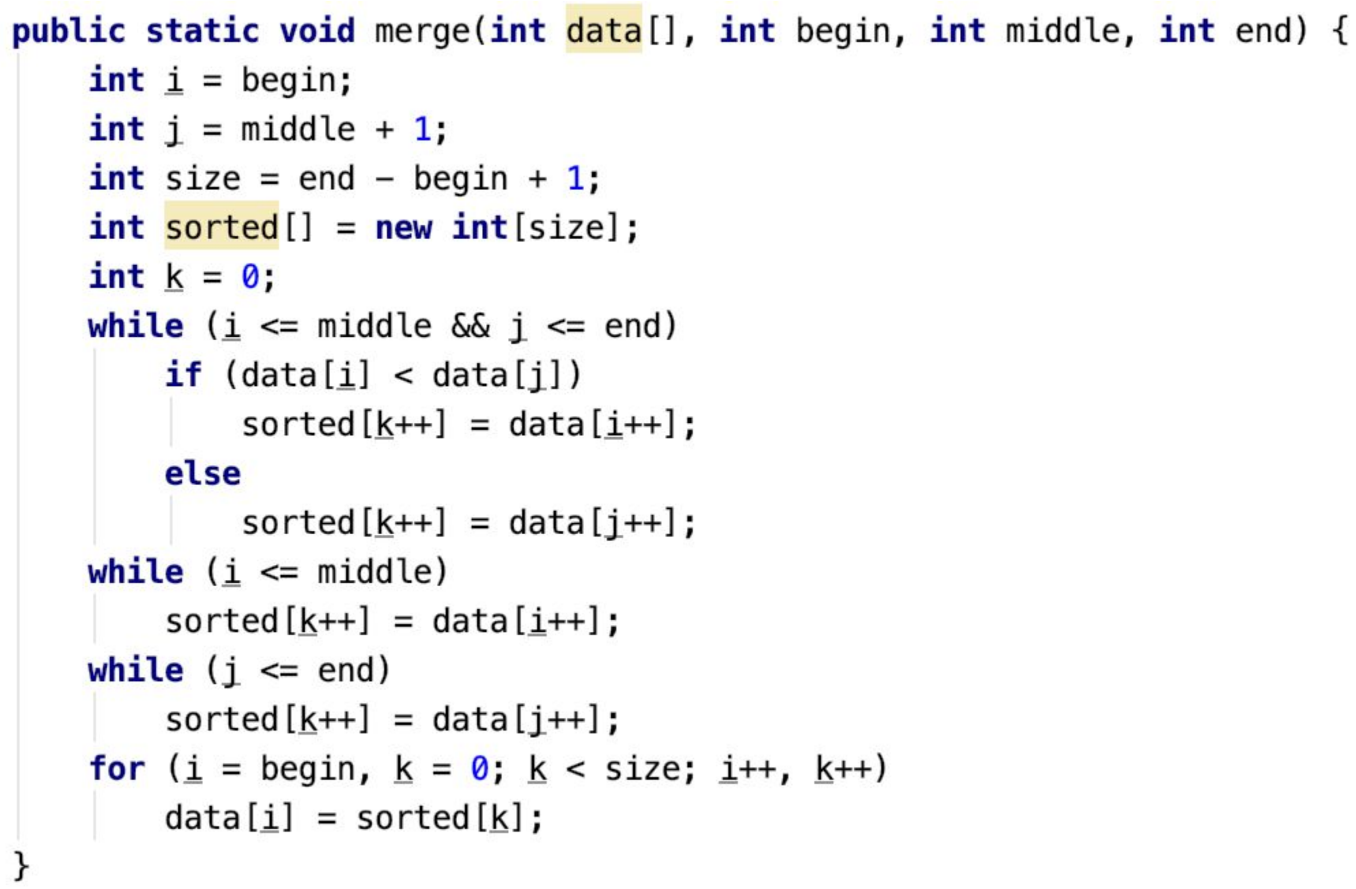
Merge Sort

Pause the video now and try to implement the merge sort algorithm!

GitHub

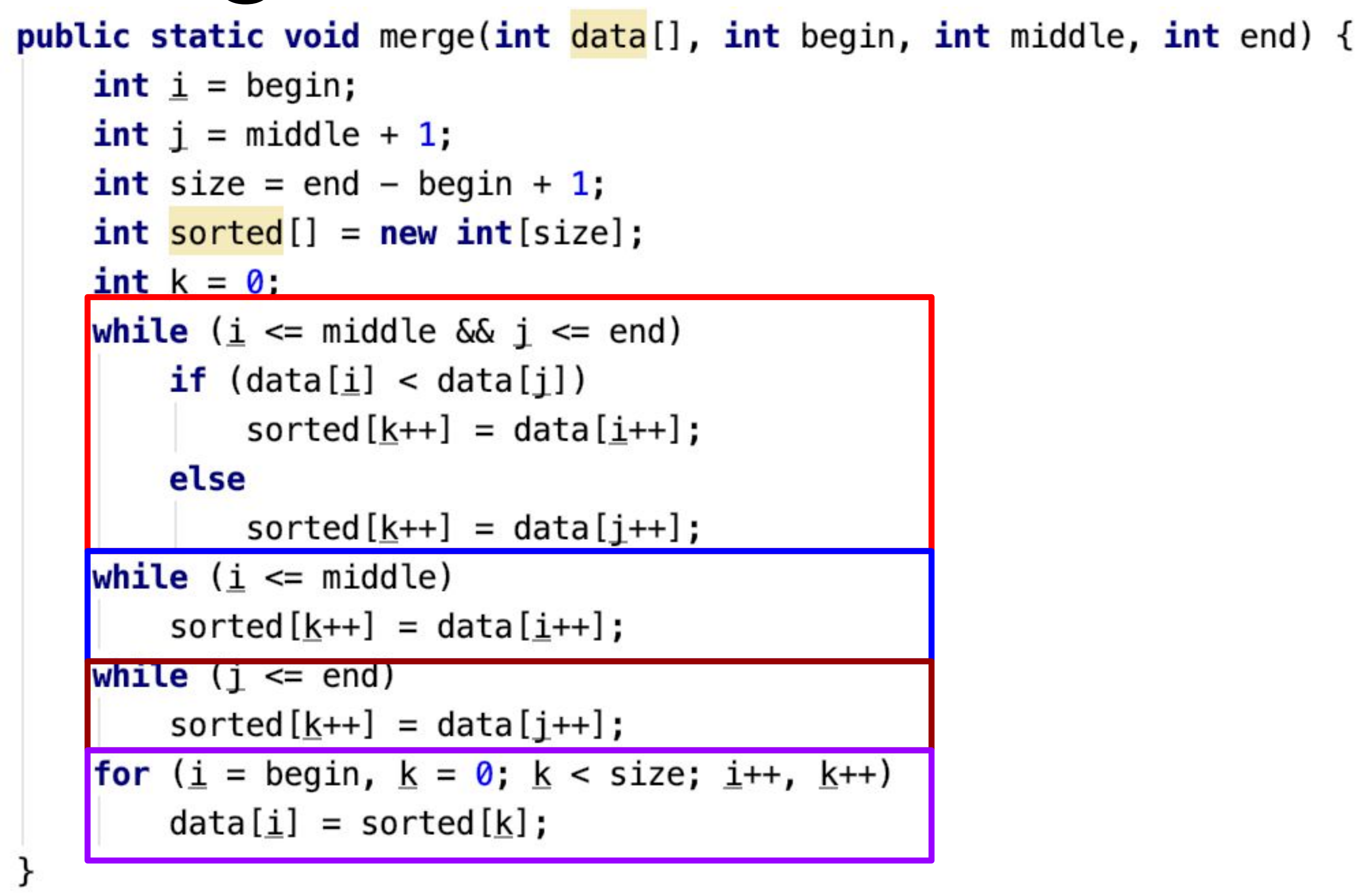
PAUSE

Merge Sort



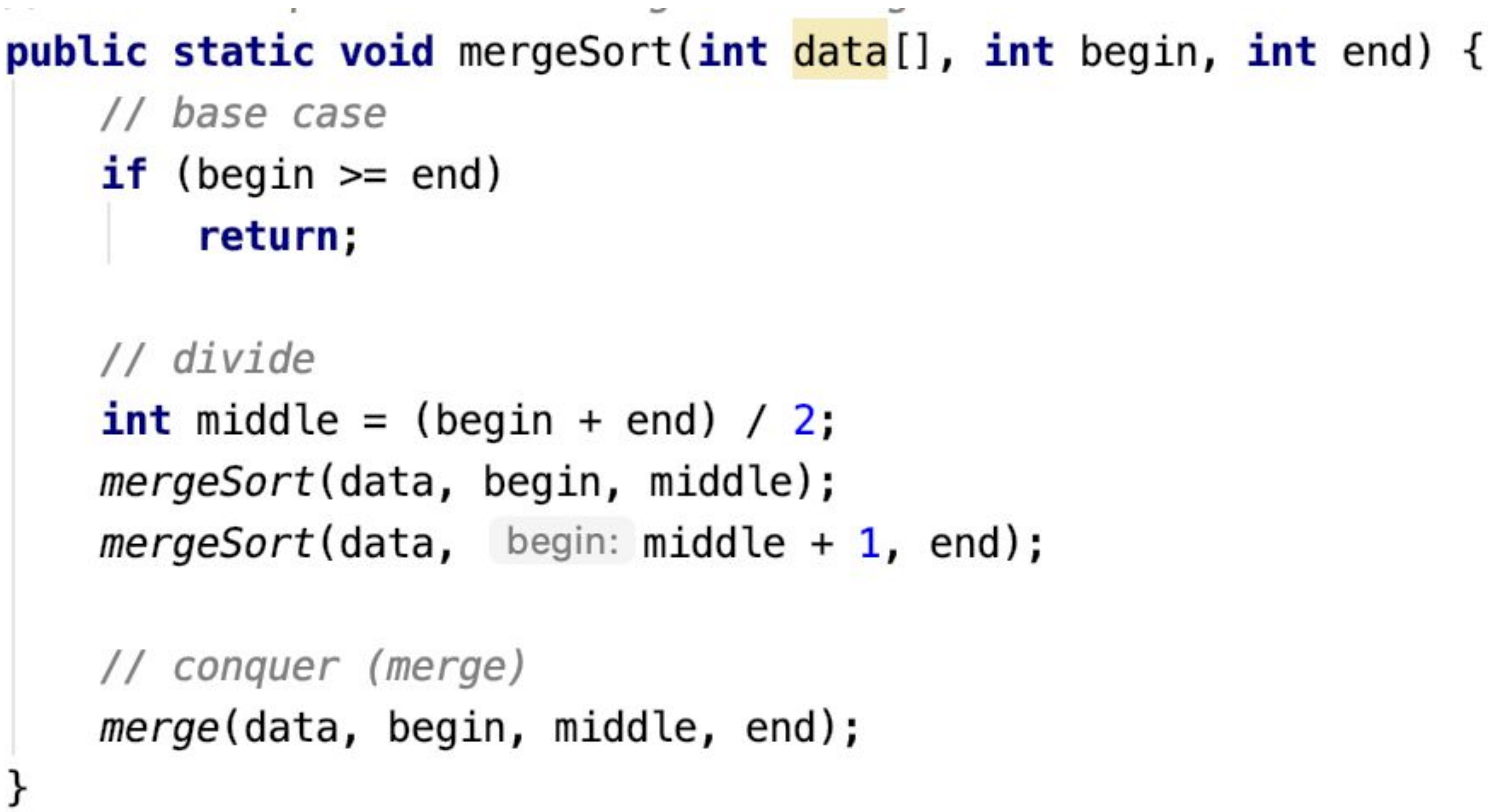
```java
public static void merge(int data[], int begin, int middle, int end) {
    int i = begin;
    int j = middle + 1;
    int size = end - begin + 1;
    int sorted[] = new int[size];
    int k = 0;
    while (i <= middle && j <= end)
        if (data[i] < data[j])
            sorted[k++] = data[i++];
        else
            sorted[k++] = data[j++];
    while (i <= middle)
        sorted[k++] = data[i++];
    while (j <= end)
        sorted[k++] = data[j++];
    for (i = begin, k = 0; k < size; i++, k++)
        data[i] = sorted[k];
}
```

Merge Sort



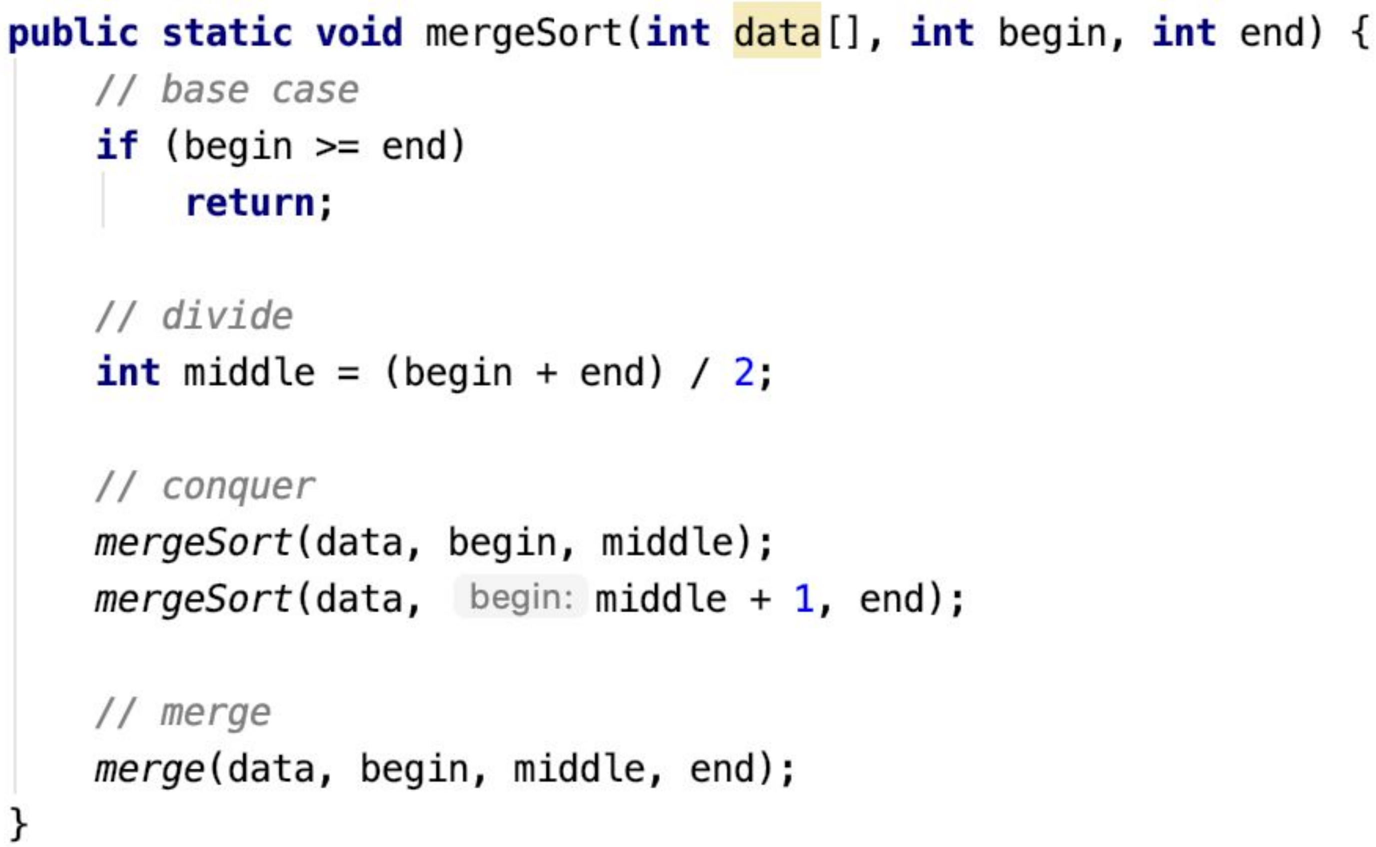
```
public static void merge(int data[], int begin, int middle, int end) {
    int i = begin;
    int j = middle + 1;
    int size = end - begin + 1;
    int sorted[] = new int[size];
    int k = 0;
    while (i <= middle && j <= end)
        if (data[i] < data[j])
            sorted[k++] = data[i++];
        else
            sorted[k++] = data[j++];
    while (i <= middle)
        sorted[k++] = data[i++];
    while (j <= end)
        sorted[k++] = data[j++];
    for (i = begin, k = 0; k < size; i++, k++)
        data[i] = sorted[k];
}
```

Merge Sort



```
public static void mergeSort(int data[], int begin, int end) {
    // base case
    if (begin >= end)
        return;

    // divide
    int middle = (begin + end) / 2;
    mergeSort(data, begin, middle);
    mergeSort(data, begin: middle + 1, end);

    // conquer (merge)
    merge(data, begin, middle, end);
}
```

Merge Sort



```java
public static void mergeSort(int data[], int begin, int end) {
    // base case
    if (begin >= end)
        return;

    // divide
    int middle = (begin + end) / 2;

    // conquer
    mergeSort(data, begin, middle);
    mergeSort(data,  begin: middle + 1, end);

    // merge
    merge(data, begin, middle, end);
}
```