

A decorative graphic on the left side of the slide. It consists of a green rounded square with a dashed border. A vertical red line with a dashed border passes through the center of the green square. A horizontal purple line with a dashed border extends from the right side of the green square across the slide.

## Stage 2. External Merge Sort

Younghoon Kim  
(nongaussian@hanyang.ac.kr)



# Sample Code

```
import org.apache.commons.lang3.tuple.MutableTriple;

public void sort(String infile, String outfile, String tmpdir, int blocksize, int nblocks) throws IOException {
    1) initial phase
    ArrayList<MutableTriple<Integer, Integer, Integer>> dataArr = new ArrayList<>(nElement);
    ...

    2) n-way merge
    _externalMergeSort(tmpdir, outfile, 0);
}

private void _externalMergeSort(String tmpDir, String outputFile, int step) throws IOException {
    File[] fileArr = (new File(tmpDir + File.separator + String.valueOf(prevStep))).listFiles();
    if (fileArr.length <= nblocks - 1) {
        for (File f : fileArr) {
            DataInputStream dos = new ... (f.getAbsolutePath(), blocksize);
            ...
        }
    } else {
        for (File f : fileArr) {
            ...
            cnt++;
            if (cnt == nblocks - 1) {
                n_way_merge(...);
            }
        }
        _externalMergeSort(tmpDir, outputFile, step+1);
    }
}
```



# Sample Code

```
public void n_way_merge(List<DataInputStream> files, String outputFile) throws IOException {
    PriorityQueue<DataManager> queue = new PriorityQueue<>
        (files.size(), new Comparator<DataManager>() {
            public int compare(DataManager o1, DataManager o2) {
                return o1.tuple.compareTo(o2.tuple);
            }
        });
    while (queue.size() != 0) {
        DataManager dm = queue.poll();
        MutableTriple<Integer, Integer, Integer> tmp = dm.getTuple();
        ...
    }
}
```