Lecture 32: Sorting Big Data: Sorting on the Disk

splitting the input file

```
def split(file, size):
    open_files = []
    with open(file) as fin:
        tmp = tempfile.TemporaryFile('w+t')
        for line in fin:
            if os.fstat(tmp.fileno()).st_size < size:
                print(line,file=tmp,end="")
            else:
                tmp.flush()
                tmp.seek(0)
                open_files.append(tmp)
                tmp = tempfile.TemporaryFile('w+t')
                print(line,file=tmp,end="")

        tmp.flush()
        tmp.seek(0)
        open_files.append(tmp)

    return open_files
```

sorting the smaller files in memory

```
def sort_files(files):

    sorted_files = []

    for file in files:
        contents = [line for line in file]
        contents.sort()
        tmp = tempfile.TemporaryFile('w+t')
        for line in contents:
            print(line, file=tmp, end="")
        tmp.flush()
        tmp.seek(0)
        sorted_files.append(tmp)
        file.close()

    return sorted_files
```

merging

```
def merge_files(files, final):

    tmp = files[0]
    for file in files[1:]:
        tmp2 = tempfile.TemporaryFile('w+t')
        for line in merge_iter(tmp, file):
            print(line,file=tmp2,end="")
        tmp.close()
        tmp = tmp2
        tmp.flush()
        tmp.seek(0)

    with open(final, 'w+t') as fout:
        for line in tmp:
            print(line,end='',file=fout)
```

streaming merge

```
def merge_iter(iter1, iter2):
    try:
        val1 = next(iter1)
        val2 = next(iter2)
        while True:
            if val1 < val2:
                yield val1
                val1 = next(iter1)
            else:
                yield val2
                val2 = next(iter2)

    except StopIteration:
        # one of the two iterators is empty, but we don't know which, so
        # just yield all the remaining values in both (the one without
        # any remaining values won't yield anything
        for val in iter1:
            yield val
        for val in iter2:
            yield val
```

sorting big data on the disk

```
def bigsort(input, output, size=2*20):
    merge_files(sort_files(split(input, size)), output)
```