Homework 3

Dillon VanBuskirk

September 18, 2016

1 Introduction, Requirements, and Results

For this assigment, I coded a solution to the textbook exercise 6.5-9 that sorts and merges k sorted lists into a single sorted list in $O(n \log k)$ time, where n is the total number of elements. This has been completed using a min-heap.

2 Pseudocode

The algorithm I've used utitlizes the algorithm package from C++ that has push heap and pop heap functions which coding the solution very simple. The algorithm is presented below.

Algorithm 1: k-way sorted list merge

```
for (k lists)
        heap.push(k[0])
while (heap.size > 0)
        heap.pop
        if ((array which the minimum was extracted).size > 0)
                heap.push(k[next])
```

With the use of the algorithm package's functions, which implement the heap functions using a vector, it made it very easy. Typically, after extracting the minimum, it is required that the heap properties must be restored. However, since it used vectors, extracting the minimum from the heap automatically adjusts the front pointer after that first element is erased.

3 Testing

Testing that was performed for this assignment began with expected input which was k files which contained the same quantity of unique elements. From there, I tested non-unique elements, different number of elements per file, as well as negatives and massive outliers. All of these tests passed. I've included the input files which I used and included the final output file which read all 10 files.

4 Efficiency

This algorithm is within the required time, $O(n \log k)$. This is accomplished with the heap because the heap will only ever have k amount of elements so each insert (after extraction) will always take lgk time. Since it will do this for every element in every file, it will perform the insert n times, which is the total number of inputs. Therefore, the algorithm runs sorts and merges k sorted lists into a single sorted list in $O(n \log k)$ time.