6.5-9.

Give an $O(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. (*Hint:* Use a min-heap for k-way merging.)

Answer.

The K-Way-Merge procedure first build a min-heap T from head elements of all the k input lists. Now the top element in T is the smallest element to append to the final list U. The procedure then successively transplants this top element to U with its position occupied by its successor in the input list it comes from. A Min-Heapify operation is performed at the top of T to maintain the min-heap property. To retrieve the original successor of elements that has been removed from the input list efficiently, we augment each element with a pointer *src-list* which points to the list it came from.

```
K-Way-Merge(A_1, A_2, ..., A_k)
 1  n = 0
 2  for i = 1 to k
 3       n = n + A_i.length
 4  for i = 1 to k
 5       T[i] = List-Extract-Head(A_i)
 6  Build-Min-Heap(T)
 7  for j = 1 to n
 8       U[j] = T[1]
 9       U[j].src-list = U
10       T[1] = List-Extract-Head(T[1].src-list)
11       Min-Heapify(T, 1)
```

The List-Extract-Head procedure in lines 5 and 9 above extracts the head element out of a list.

```
List-Extract-Head(L)
1  head = L[1]
2  for i = 2 to L.length
3       L[i-1] = L[i]
4  L.length = L.length - 1
5  return head
```

It takes List-Extract-Head $O(n/k)$ time to accomplish its work. Observe that the amount of basic operations K-Way-Merge perform is dorminated by the **for** loop in lines 7–11, in which either List-Extract-Head or Min-Heapify takes more time. A little math shows that

$$\lg k > n/k$$

when $k > 2$. Therefore, the running time of K-Way-Merge is $O(n \lg k)$.

Email address: informlarry@gmail.com