CS 240: Data Structures and Data Management **Winter 2023**

Tutorial 03: January 30

This tutorial covers problems relating to priority queues, and average- and expected-case runtime analysis. There are 3 questions: 1 easy [E], 1 medium [M], and 1 hard [H].

1. [E] Given a family k sorted arrays $A_1, \ldots, A_k$, where the combination of the k arrays has n elements, give an $O(n \log k)$ time algorithm that produces a single sorted array containing all n elements. Hint: use a priority queue.

2. [M] Give the best-case and expected running time for the following function. You can assume that the Shuffle operation requires $\mathcal{O}(n)$ time and the array A contains no duplicates.
Note: the *Shuffle*() function produces each permutation equally likely.

Algorithm 1: MonkeySort($\mathcal{A}$)

Input: Array $\mathcal{A}$
Output: Array $\mathcal{B}$ which is in sorted order

```
1 B ← A
2 shuffle(B)
3 if B is sorted then
4 |   return B
5 end
6 else
7 |   return MonkeySort(A)
8 end
```

3. [H] Let A and B be two bitstrings of length n (modelled here as arrays where each entry is 0 or 1). A *string-compare* tests whether A is smaller, larger, or the same as B and works as follows:

Algorithm 2: *str-cmp*(A, B, n)

```
1 for (i = 0; i < n; i++) do
2 |   if (A[i] < B[i]) then return "A is smaller"
3 |   if (A[i] > B[i]) then return "A is bigger"
4 end
5 return "They are equal"
```

Show that the average-case run-time of *str-cmp* is in $O(1)$. You may use without proof that $\sum_{i \geq 0} \frac{i}{2^i} \in O(1)$.