CS 240: Data Structures and Data Management **Spring 2026**

Tutorial 03: June 5

1. **Average Runtime Analysis** Suppose A is an array containing n distinct elements. In addition, assume each element is in between 1 and n, inclusive. Analyze this pseudo-code to determine a tight bound on the average number of question mark (?) that are printed, rather than a runtime. You may assume n is divisible by 2.

```
mystery(A, n)
    count = 1
    for i = 1 to n-1
        if A[i] is divisible by A[0]
            count++
    for i = 1 to count
        print("?")
```

2. **Expected Runtime** Give a tight bound on the expected number of iterations of the while loop in the following program.

```
def silly(n):
    while n != 0:
        n = randint(0, n) # random int in {0, ..., n}
    print("done")
```

3. **Heap review** A complete binary tree containing n nodes is stored level-by-level.
   (a) Derive a formula for the left child, right child, and parent of the i^{th} node.
   (b) Prove that if $i \geq \lfloor \frac{n}{2} \rfloor$, then i is a leaf node (hint: use your answer to part (a)).
   (c) Explain why a node at height h participates in at most h swaps during `fix-down`.
   (d) Combine (b) and (c) to explain why `fix-down` only processes nodes with index $\leq \lfloor \frac{n}{2} \rfloor - 1$.
   (e) Give a formula for the number of nodes of height h.
   (f) Prove that heapify can be done in time $\Theta(n)$ (hint: $\sum_{k=1}^{\infty} \frac{k}{2^k} = 2$).
   Bonus: Can you find a exact formula for the number of swaps required by `heapify` in the worst case?

4. **Bonus (no solution provided, feel free to ask about this during consulting hours)** Consider the following algorithm, which takes as input a (1-indexed) array A of length n, each entry of which is an integer between 1 and m (inclusive).

```
def maximum(A):
    max = 0
    for k = 1 to len(A):
        if A[k] > max:
            max = A[k]
    return max
```

How many times is `max` (re)assigned

- in the worst case?
- in the best case?
- in the average case?