CS 473 ✧ Spring 2020
Homework 4

Due Wednesday, March 4, 2020 at 9pm

---

1. Consider a random walk on a path with vertices numbered $1, 2, \ldots, n$ from left to right. At each step, we flip a coin to decide which direction to walk, moving one step left or one step right with equal probability. The random walk ends when we fall off one end of the path, either by moving left from vertex 1 or by moving right from vertex n.

   (a) Prove that if we start at vertex 1, the probability that the walk ends by falling off the *right* end of the path is exactly $1/(n+1)$.

   (b) Prove that if we start at vertex k, the probability that the walk ends by falling off the *right* end of the path is exactly $k/(n+1)$.

   (c) Prove that if we start at vertex 1, the expected number of steps before the random walk ends is exactly n.

   (d) What is the *exact* expected length of the random walk if we start at vertex k, as a function of n and k? Prove your result is correct. (For partial credit, give a tight Θ-bound for the case $k = (n+1)/2$, assuming n is odd.)

   [Hint: Trust the recursion fairy. Yes, (b) implies (a) and (d) implies (c).]

2. The following randomized variant of "one-armed quicksort" selects the kth smallest element in an unsorted array $A[1\,..\,n]$. As usual, Partition$(A[1\,..\,n], p)$ partitions the array A into three parts by comparing the pivot element $A[p]$ to every other element, using $n-1$ comparisons, and returns the new index of the pivot element.

```
QuickSelect($A[1\,..\,n], k$):
  $r \leftarrow$ Partition($A[1\,..\,n]$, Random($n$))
  if $k < r$
      return QuickSelect($A[1\,..\,r-1], k$)
  else if $k > r$
      return QuickSelect($A[r+1\,..\,n], k-r$)
  else
      return $A[k]$
```

   (a) State a recurrence for the expected running time of QuickSelect, as a function of n and k.

   (b) What is the *exact* probability that QuickSelect compares the ith smallest and jth smallest elements in the input array? The correct answer is a simple function of i, j, and k (with a few cases). *[Hint: Check your answer by trying a few small examples.]*

   (c) What is the *exact* probability that in some recursive call to QuickSelect, the first argument is the subarray $A[i\,..\,j]$? The correct answer is a simple function of i, j, and k (with more cases). *[Hint: Check your answer by trying a few small examples.]*

   (d) Show that for any n and k, QuickSelect runs in $\Theta(n)$ expected time. You can use either the recurrence from part (a) or the probabilities from part (b) or (c).

3. A ***meldable priority queue*** stores a set of priorities from some totally-ordered universe (such as the integers) and supports the following operations:

   - MakeQueue: Return a new priority queue containing the empty set.
   - FindMin(Q): Return the smallest element of Q (if any).
   - DeleteMin(Q): Remove the smallest element in Q (if any).
   - Insert(Q, x): Insert element x into Q, if it is not already there.
   - DecreasePriority(Q, x, y): Replace an element $x \in Q$ with a new element $y < x$. (If $y \geq x$, the operation fails.) The input includes a pointer directly to the node in Q containing x.
   - Delete(Q, x): Delete the element $x \in Q$. The input is a pointer directly to the node in Q containing x.
   - Meld(Q_1, Q_2): Return a new priority queue containing all the elements of Q_1 and Q_2; this operation destroys Q_1 and Q_2. The elements of Q_1 and Q_2 could be arbitrarily intermixed; we do *not* assume, for example, that every element of Q_1 is smaller than every element of Q_2.

   A simple way to implement such a data structure is to use a heap-ordered binary tree, where each node stores a priority, along with pointers to its parent and two children. Meld can be implemented using the following randomized algorithm. The input consists of pointers to the roots of the two trees.

```
Meld($Q_1, Q_2$):
  if $Q_1$ = Null then return $Q_2$
  if $Q_2$ = Null then return $Q_1$
  if $priority(Q_1) > priority(Q_2)$
      swap $Q_1 \leftrightarrow Q_2$
  with probability 1/2
      $left(Q_1) \leftarrow$ Meld($left(Q_1), Q_2$)
  else
      $right(Q_1) \leftarrow$ Meld($right(Q_1), Q_2$)
  return $Q_1$
```

   (a) Prove that for *any* heap-ordered binary trees Q_1 and Q_2 (not just those constructed by the operations listed above), the expected running time of Meld(Q_1, Q_2) is $O(\log n)$, where $n = |Q_1| + |Q_2|$. *[Hint: What is the expected length of a random root-to-leaf path in an n-node binary tree, where each left/right choice is made with equal probability?]*

   (b) Prove that Meld(Q_1, Q_2) runs in $O(\log n)$ time with high probability. *[Hint: You can use Chernoff bounds, but the simpler identity $\binom{c}{k} \leq (ce)^k$ is actually sufficient.]*

   (c) Show that each of the other meldable priority queue operations can be implemented with at most one call to Meld and $O(1)$ additional time. (It follows that every operation takes $O(\log n)$ time with high probability.)