This assignment covers material from the lectures on Chapters 7 and 8, in preparation for Quiz 4.

Due: October 14, 11:59pm

Not in textbook: Blondie has an array $A[1, \ldots, n + m]$, in which $n > 0$ of the elements are ‘0’ and the remaining $m > 0$ elements are ‘1’. He wants to find the index of one of the 0’s, and he uses the following randomized algorithm:

```
FindIndexOfZero(A)
while true do
  $i =$ random integer between 1 and $n + m$
  if $A[i] = 0$ then
    return $i$
  end if
end while
```

Find a tight asymptotic bound on the worst-case expected run time of this algorithm.

Page 180: Exercises 7.3-2 (Write a recurrence for the number of random numbers generated, then solve that recurrence via the substitution method.)

Page 184: Exercises 7.4-1, 7.4-2

Not in textbook: Consider this randomized version of MERGESORT:

```
RANDOMIZEDMERGESORT(A, $\ell$, $r$)
if $\ell < r$ then
  $m =$ random integer in the range $\{\ell, \ldots, r - 1\}$
  RANDOMIZEDMERGESORT(A, $\ell$, $m$)
  RANDOMIZEDMERGESORT(A, $m + 1$, $r$)
  MERGE(A, $\ell$, $m$, $r$)
end if
```

Recall that $\ell$ and $r$ are the lower and upper limits of the portion of the array to be sorted. The only change from standard MERGESORT is that $m$ is selected randomly, rather than dividing the array into two equal parts. Write and solve, using any appropriate method, a recurrence for the worst-case expected run time of this algorithm. Is this algorithm an improvement over the standard MERGESORT?

Not in textbook: In the search problem, the input is an array $A$ of size $n$ along with a search key $k$. The output is an integer $i$ such that $A[i] = k$, or $-1$ if $k$ is not in $A$. Prove, using the decision tree method, that any correct algorithm for this problem based on comparisons ($<$, $>$, $\leq$, $\geq$, and $=$) between elements takes $\Omega(\lg n)$ time.