CS170 Discussion Section 2: *1/25*

1. Recurrence Relations

Solve the following recurrences: in each case, find $f(n)$ so that $T(n) = \Theta(f(n))$.

1. $T(n) = 4T(n/2) + 42n$ $\qquad T(n) = \Theta(\qquad)$

2. $T(n) = 4T(n/3) + n^2$ $\qquad T(n) = \Theta(\qquad)$

3. $T(n) = 2T(2n/3) + T(n/3) + n^2$ $\qquad T(n) = \Theta(\qquad)$

4. $T(n) = 3T(n/4) + n \log n$ $\qquad T(n) = \Theta(\qquad)$

2. Counting inversions

This problem arises in the analysis of *rankings*. Consider comparing two rankings. One way is to label the elements (books, movies, etc.) from 1 to n according to one of the rankings, then order these labels according to the other ranking, and see how many pairs are "out of order".

We are given a sequence of n distinct numbers $a_1, \cdots, a_n$. We say that two indices $i < j$ form an inversion if $a_i > a_j$ that is if the two elements a_i and a_j are "out of order". Provide a divide and conquer algorithm to determine the number of inversions in the sequence $a_1, \cdots, a_n$ in time $O(n \log n)$ (*Hint:* Modify merge sort to count during merging).

3. Find the missing integer

An array A of length N contains all the integers from 0 to N except one (in some random order). In this problem, we cannot access an entire integer in A with a single operation. The elements of A are represented in binary, and the only operation we can use to access them is "fetch the jth bit of $A[i]$". Using only this operation to access A, give an algorithm that determines the missing integer by looking at only $O(N)$ bits. (Note that there are $O(N \log N)$ bits total in A, so we can't even look at all the bits). Assume the numbers are in bit representation with leading 0s.

4. K-Largest Elements

Give an efficient algorithm to determine the k-largest elements of an unsorted list of integers of length n, in any order. Your algorithm should run in $O(n)$ expected time for a fixed k. You may assume $n \geq k$.

Main idea:

Psuedocode:

Proof of correctness:

Running time analysis: