1. Solve the recurrence relation $T(n) = 3T(n/3) + n$

- by working through the recursion tree
  
  The important thing here is that you understand what the recursion tree is doing and why it works. Basically, you’re unrolling the recursion and writing down how much work you do at each step.

- by the guess and prove method, trying at least one wrong guess
  
  The “prove” part of this is an inductive proof. If you’re not comfortable setting up and working through the proof (where do you use your inductive hypothesis and where do you use the recurrence as given?), stop by office hours.
2. Insertion-sort works by sorting the first $i - 1$ elements and then inserting the $i$th element where it belongs among the first $i - 1$. Here is some code:

```c
void insertionsort(int a[], int n) {
    int i, j;

    for (i=1; i<n; i++) {
        for (j=i-1; j>=0; j--) {
            if (a[j+1] >= a[j]) break;
            swap(a, j, j+1); // swap these elements
        }
    }
}
```

Let’s look at running time in terms of the number of comparisons between elements.

(a) If the input consists of $n$ elements in sorted order, how many comparisons are made?

(b) What if the input is in descending order?

(c) What is the upper bound on the number of comparisons done by the algorithm?

(d) Call a pair of indices $(i, j)$ an inversion if $i < j$ but $a_i > a_j$. If $I$ is the number of inversions in the input, show that the number of comparisons is always between $I$ and $I + n - 1$.

In recitation, we looked at a recursive version of this procedure:

```c
insertion_sort(null)= null
| insertion_sort(hd::tl) =
    insert(hd, insertion_sort(tl))

insert(elt, null) = [elt]
| insert(elt, lst as (hd::tl)) =
    if (elt <= hd) then cons(elt, lst)
    else cons(hd, insert(elt, tl))
```
(a) Look at the insert procedure and notice that we’ll have to do a comparison (the “if” clause) for every element except the last one; we always enter the “then” clause, though, so that’s all: $n - 1$

(b) Here, we always enter the “then” clause and thus have to do the maximum possible number of comparisons (see below: $n(n - 1)/2$

(c) We can directly translate the procedure into a recurrence relation. Let $T_I(n)$ be the number of comparisons when inserting an element into a list of length $n$.

$T_I(0) = 0$, since there are no comparisons in the base case

$T_I(n) \leq 1 + T_I(n - 1)$

So $T_I(n) \leq n$.

Let $T_S(n)$ be the number of comparisons when sorting a list of length $n$.

$T_S(0) = 0$ because there are no comparisons in the base case of insertion sort

$T_S(n) = T_S(n - 1) + T_I(n - 1) \leq T_S(n - 1) + n - 1 = n(n - 1)/2$

since on every recursive call, we need to sort a list with one less element and then insert the first element into it.

(d) Intuition: When we check if the element we’re inserting is less than $hd$ in the insert procedure, what we’re really doing is checking whether that element is involved in any inversions with elements that came later in the input. If it is, we enter the “then” clause and do one comparison for each inversion it was involved in; if not, we did just that one extra comparison. Since we do $n - 1$ inserts, that’s at least $I$ and at most $I + n - 1$ comparisons.
3. Suppose you flip a fair coin \(n\) times. We say that a \(k\)-streak occurs in the sequence if, for some \(i\) between 1 and \(n\), flips \(i, i+1, \ldots, i+k-1\) all turn up heads. For example, in the sequence

\[
H, T, H, H, H
\]

there are four 1-streaks, two 2-streaks, and one 3-streak.

Give an expression (in terms of \(k\) and \(n\)) for the expected number of \(k\)-streaks.

*Hint:* what’s the probability that the set of \(k\) flips beginning with the first flip forms a \(k\)-streak? What about the set of \(k\) flips beginning with the second flip? The third flip? ...
4. Is $3^n \in O(2^n)$? Prove.

   No. *Try to prove it by induction and see where it fails.*