# Lecture 7 Scratchwork

COT 4400, Fall 2015

September 15, 2015

Identify and prove a tight upper bound on the worst-case time complexity of Insertion Sort (below).

**Input**: *data*: an array of integers to sort
**Input**: $n$: the number of values in data
**Output**: permutation of *data* such that $data[1] \leq \ldots \leq data[n]$
1 **Algorithm:** Selection Sort
2 **if** $n > 1$ **then**
3     Call Insertion Sort on $data[1..n-1]$
4     Let $ins = data[n]$
5     Let $j =$ last index of $data[1..n-1] \leq ins$
6     Shift $data[j+1..n-1]$ to the right one space
7     $data[j+1] = ins$
8 **end**
9 **return** *data*

**Goal:** Define $T(n)$ in terms of $T(n-1)$ and $n$.
Base case: $T(1) = O(1)$
Recursive case: Line 2 takes $T(n-1)$ time (calls Insertion Sort on array of length $n-1$).

$$\begin{aligned} T(n) &= T(n-1) + 2O(n) + 3O(1) \\ &= T(n-1) + O(n) \end{aligned}$$

Now we solve the recurrence $T(n) = T(n-1) + O(n)$:
Char. eqn: $c(x) = x - 1 = 0$
Roots: $x = 1$
General form of sol'n: $T(n) = O(1^n) + O(n^m f(n))$
$m = 1$, so $T(n) = O(1) + n^1 O(n)$
$T(n) = O(1) + O(n^2) = O(n^2)$