These lecture stubs are incomplete notes, that you can complete yourself. You have the option to:

- Complete them in class as I explain things on the board
- Complete them later when you study the material again
- Bring them with you in any of the CS231 exams

Part 1 - Complete the table from lecture 1

In each entry of this table, draw or describe how each corresponding operation is performed on each data structure.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Array</th>
<th>Linked List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find by index</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find by value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete by index</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete by value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add to the end</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add to the front</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part 2 - Writing pseudocode

Sorting is one the most common operations performed on lists and arrays, and it’s one of the first things that are typically taught in an algorithms class. In this lecture, we’ll discuss insertion sort in detail.

In insertion sort, the idea is that you scan the array from left to right, and take elements that are out of place, and “insert” them into the correct spot on the left hand side of the array. The code of that algorithm can be written as follows:

**Algorithm 1** InsertionSort(A)

1: for i in range(2,len(A)) do
2:   current = A[i]
3:   j = i-1
4:   while j ≥ 1 and A[j] > current do
6:     j–
7:   end while
8:   A[j+1] = current
9: end for

Part 3 - Is it correct?

How can you prove that this algorithm is correct?
Part 4 - Is it efficient?

1. What is the complexity of algorithm above?

2. How did you figure this out?

Let’s analyze it together.