**Week 7: Searching and Sorting Algorithms**

1. **selection\_sort**:

* **Guide**:
  + Selection sort works by repeatedly finding the smallest (or largest) element from the unsorted portion and swapping it with the first unsorted element.
* **Pseudocode**:
* FUNCTION selection\_sort(nums):  
   FOR i from 0 to length of nums - 1:  
   FIND the smallest item between index i and end of nums  
   SWAP nums[i] with the found smallest item  
   RETURN nums

1. **quick\_sort** (repeated from Week 3):

* **Guide**:
  + Quick sort involves picking a ‘pivot’ element and partitioning other elements into two subsets - those less than the pivot and those greater. This sorting mechanism is recursive, applying the same logic to each subset.
* **Pseudocode** (same as before):
* FUNCTION quick\_sort(arr):  
   IF length of arr is 1 or less:  
   RETURN arr  
   CHOOSE a pivot element from arr  
   PARTITION arr into two: elements less than pivot and elements greater  
   RETURN quick\_sort(lesser elements) concatenated with pivot concatenated with quick\_sort(greater elements)

1. **binary\_search\_first\_occurrence**:

* **Guide**:
  + This variant of binary search aims to find the first occurrence of an element. If you find an element matching the target, check its left to ensure it’s the first occurrence.
* **Pseudocode**:
* FUNCTION binary\_search\_first\_occurrence(arr, x):  
   INITIALIZE low to 0 and high to length of arr - 1  
   INITIALIZE result to -1  
   WHILE low <= high:  
   COMPUTE mid as average of low and high  
   IF arr[mid] is x:  
   UPDATE result to mid  
   MOVE high to mid - 1 (to find first occurrence)  
   ELSE IF arr[mid] is less than x:  
   MOVE low to mid + 1  
   ELSE:  
   MOVE high to mid - 1  
   RETURN result