**Week 3: Divide and Conquer Algorithms**

1. **merge\_sort**:

* **Guide**:
  + Merge sort divides the list into halves, sorts each half, and then merges them back. The actual ‘merging’ process is crucial as you combine two sorted halves into a single sorted list.
* **Pseudocode**:
* FUNCTION merge\_sort(arr):  
   IF length of arr is 1 or less:  
   RETURN arr  
   SPLIT arr into two halves  
   SORT each half using merge\_sort  
   MERGE the two sorted halves into one sorted list  
   RETURN merged list

1. **quick\_sort**:

* **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**:
* 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**:

* **Guide**:
  + Binary search works by repeatedly dividing in half the portion of the list that could contain the item, until you’ve narrowed the possibilities to just one. It’s essential for the list to be sorted for binary search to work.
* **Pseudocode**:
* FUNCTION binary\_search(arr, x):  
   INITIALIZE low to 0 and high to length of arr minus 1  
   WHILE low <= high:  
   COMPUTE mid as the average of low and high  
   IF arr[mid] is x:  
   RETURN mid  
   ELSE IF arr[mid] is less than x:  
   SET low to mid + 1  
   ELSE:  
   SET high to mid - 1  
   RETURN -1