# Exercises

## 2.3-5 Referring back to the searching problem (see Exercise 2.1-3), observe that if the sequence A is sorted, we can check the midpoint of the sequence against V and eliminate half of the sequence from further consideration. The *binary search* repeats this procedure, halving the size of the remaining portion of the sequence each time. Write pseudocode, either iterative or recursive, for binary search. Argue that the worst-case running time of binary search is Ѳ(lg n).

The Binary search is an algorithm based on divide-and-conquer design technique. Which means that on each iteration/division, the number of elements becomes **n / number of iteration \* 2**. Which means that the height of the recursive tree is lg n => **Ѳ(lg n).**

## 2.3-6 Observe that the while loop of lines 5-7 of the INSERTION-SORT procedude in Section 2.1 uses a linear search to scan (backward) through the sorted subarray A[1..j – 1]. Can we use a binary search instead to improve the overall worst-case running time of insertion sort to Ѳ(n lg n)?

It’s practically impossible to combine these two algorithms because Insertion sort is a sorting algorithm while Binary search is working over **already sorted data.**