Question 1 (9)

BINARYSEARCH(sortedList, LowerNumber, UpperNumber, midpoint) (1)

‘’’takes a sorted list of integers. Two integers that specify the interval we are searching for. And an integer calculated by the starting index of the list + (the size of the list – the starting index of the list)/2. It will be used to index the midpoint. The function will return a Boolean value.‘’’

// while a number in the interval is not found and the list has not completely been checked

While valueFound = False and endOfList = False (log(n))

//if the midpoint number is in the interval return true

if sortedList[midpoint] < UpperNumber and sortedList[midpoint] > LowerNumber (1)

valueFound <- True (1)

//if the midpoint number is greater than the upper interval number

//then interval we're searching for is not in list AFTER the midpoint,

//so get rid of that part.

else if sortedList[midpoint] > UpperNumber (log(n))

//checks if list if the start and end have met or gone past eachother.

//if it is has then end the loop as we would have already checked whether

//it is a value in the interval.

if start = end (log(n))

endOfList <- True (1)

//calculates new end of searchable list and new midpoint.

end <- midpoint – 1 (log(n))

midpoint <- start + ( end – start)/2 (log(n))

round down midpoint (log(n))

Else (log(n))

if start = end (log(n))

endOfList <- True (1)

// calculates new beginning of searchable list and new midpoint.

start <- midpoint + 1 (log(n))

midpoint <- start + (end - start)/2 (log(n))

round down midpoint (log(n))

// returns whether a value in the interval was found or not.

return valueFound (1)

// to do this with recursion remove the part of the list that is no longer searched until interval found or no more list.

Runtime: 11log(n) + 6

* As the length of the list getting searched is shortened by moving to the midpoint, and then eliminating the list that is either above it or below it.
* The while loop thus is not run n times as it is not searched sequentially.
* Thus it is instead log(n) due to it being a divide and conquer algorithm

Big O: O(log(n))