Unit 4, Week 3

Answer these questions on a separate piece of paper.

Binary Search Algorithm

1. Draw a binary search tree showing how to search for a given number in the following array, using the pseudocode specified.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Item | 2 | 4 | 5 | 8 | 10 | 13 | 15 | 22 | 29 | 31 |

def binary\_search(array, target):

    L = len(array)

    if L == 0:

        return -1

    pivot = L // 2

    if array[pivot] == target:

        return pivot

    elif array[pivot] < target:

        result = binary\_search(array[pivot+1:L], target)

        if result == -1:

            return -1

        else:

            return pivot + 1 + result

    else:

        return binary\_search(array[0:pivot], target)

2. What does a return value of -1 signify?

3. State which items could take the longest to find using this tree.

4. What must be true about the array for all items to take the same amount of time to find?

5. For each of the arrays below, explain why binary search does not work:

(a)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 |
| Item | 1 | 2 | 2 | 3 | 5 |

(b)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 |
| Item | 3 | 2 | 4 | 9 | 1 |

**Sorting Algorithms**

6. Considering MergeSort and QuickSort:

a. Which algorithm requires the most storage space?

b. Which algorithm is the most efficient for **any** given data set? Why?

c. Which algorithm chooses a random pivot to divide the data? Why does it do this?

d. Which algorithm best describes the following pseudocode:

function sort(array)

    if length(array) ≤ 1

        return array

    pivot = choose pivot element from array

    left = empty array

    right = empty array

    for each element in array

        if element < pivot

            append element to left

        else if element > pivot

            append element to right

        else

            // do nothing, element equal to pivot

    left = sort(left)

    right = sort(right)

    return concatenate(left, pivot, right)

e. Write a one line description for each sorting algorithm.