QUIZ

Started on	Monday, 15 July 2024, 2:47 PM
State	Finished
Completed on	Monday, 15 July 2024, 3:12 PM
Time taken	24 mins 39 secs

Information

This is a compulsory assessment. You have maximum of three (3) attempts to complete the assessment. The system will automatically upload your attempts when the due date and time is reached. The due date is **15 July 2024, 23:00.**

Remember, the due date is the last date for submission and not the day on which you should start with the assessment.

There are 15 quiz questions and a question (16) which is an open question. The mark for this question will be captured from a peer-evaluated project that you will complete in **the Etutor class**.

This peer-evaluation project must be submitted on the latest by **17 July 2024 23:00.** The project will be distributed between the peers - who will have until the 22th of July to do the peer evaluation and return the mark. This mark will then be copied and added as the mark to the last question. Only then will the final mark be calculated for Assessment 2.

Question 1

Complete

Marked out of 1.00

In the worst-case scenario, when is the target element not found in an array during sequential search?

- a. When the target element is at the first position
- b. When the target element is at the middle position
- c. When the target element is at the last position
- [✓ d.] When the array is empty

In the worst-case scenario for a sequential search, the target element is not found in the array when the algorithm has iterated through the entire array without finding the target. This means the element is either not present in the array or it is located at the last position.

Question 2

Complete

Marked out of 1.00

Which of the following best describes the time complexity of binary search compared to sequential search?

[✓ a. Binary search has a time complexity of O(log n), while sequential search has a time complexity of O(n).]

b. Binary search has a time complexity of O(n), while sequential search has a time complexity of O(log n).

c. Both binary search and sequential search have a time complexity of O(n).

d. Both binary search and sequential search have a time complexity of O(log n).

Binary search has a time complexity of O(log n) because it divides the search interval in half with each iteration, resulting in a logarithmic time complexity. On the other hand, sequential search has a time complexity of O(n) because it checks each element of the array sequentially until it finds the target element or reaches the end of the array, resulting in a linear time complexity.

Question 3

Complete

Marked out of 1.00

What is the key difference between sequential search and binary search?

[✓ a.] Binary search is much more efficient than sequential search for large lists due to its logarithmic time complexity, but it requires the list to be sorted

b. Binary search scans through each element sequentially, while sequential search divides the search interval in half.

c. Sequential search works only on sorted arrays, while binary search works on unsorted arrays.

d. Binary search has a time complexity of O👎 , while sequential search has a time complexity of O(log n).

Efficiency:

Binary search is much more efficient than sequential search for large lists due to its logarithmic time complexity, but it requires the list to be sorted. Sequential search is simpler and works on unsorted lists but is less efficient for large lists due to its linear time complexity.

List Requirement:

Sequential search works on any list, while binary search requires a sorted list.

Question 4

Complete

Marked out of 1.00

Which of the following is a disadvantage of the sequential search algorithm?

- a. Requires the array to be sorted
- [✓ b. Inefficient for large arrays]
- c. Time complexity of O(log n)
- d. More complex to implement

The disadvantage of the sequential search algorithm is that it is inefficient for large arrays. Since it has to iterate through each element of the array sequentially, the time it takes to find an element grows linearly with the size of the array.

Question 5

Complete

Marked out of 1.00

Consider the following array: `int[] numbers = { 5, 12, 34, 8, 20, 17 };`.

Perform a sequential search to find the index of the element `8`.

Which of the following describes the time complexity of the sequential search the best:

- a. O(log n)
- [✓ b. O(n)]
- c. O(n^2)
- d. O(1)

The time complexity of a sequential search algorithm is linear, meaning it grows in proportion to the size of the input array. In this case, it iterates through each element of the array until it finds the target element or reaches the end of the array

Question 6

Complete

Marked out of 1.00

What is the primary advantage of binary search over sequential search?

- a. Binary search is simpler to implement
- b. Binary search works on unsorted arrays
- c. Binary search has a time complexity of O(n).
- [✓ d. Binary search has a time complexity of O(log n).]

Binary search has a logarithmic time complexity, meaning it can efficiently search large, sorted arrays by repeatedly dividing the search space in half. This leads to faster search times compared to sequential search, which has a linear time complexity of O👎 , where the time to search increases linearly with the size of the array.

Question 7

Complete

Marked out of 1.00

```
Consider the following array:

int[] numbers = {12, 25, 23, 38, 8, 62,
6, 46}

Following the binary search algorithm
to determine the index for '23', what
will be the result?

The index for element 23 is :
```

[✓ a. The element 23 is not present in the array.]

b. The element 23 is in index 4.

c. The element 23 is in index 3.

d. The element 23 is in index 5.

Start:

1. Define the target element:
   $\text{target} = 23$.
2. Define the indices:
   - Start index (startstart): 0
   - End index (endend): length of array - 1 (end=7)

Iteration 1:

3. Calculate the middle index:
   $\text{mid} = \lfloor \frac{\text{start} + \text{end}}{2} \rfloor = \lfloor \frac{0+7}{2} \rfloor = 3$
4. Compare nullnumbers[mid] with the target:
   - $\text{numbers}[3] = 38$
   - null23<38, so search in the left half.

Iteration 2:

5. Update the end index:
   $\text{end} = \text{mid} - 1 = 3 - 1 = 2$
6. Recalculate the middle index:
   $\text{mid} = \lfloor \frac{\text{start} + \text{end}}{2} \rfloor = \lfloor \frac{0+2}{2} \rfloor = 1$.
7. Compare nullnumbers[mid] with the target:
   - $\text{numbers}[1] = 25$
   - null23<25, so search in the left half.

Iteration 3:

8. Update the end index:

nullend=mid−1=1−1=0.

9. Recalculate the middle index:
   $\mathrm{mid} = \lfloor \frac{\mathrm{start}+\mathrm{end}}{2} \rfloor = \lfloor \frac{0+0}{2} \rfloor = 0$.
10. Compare nullnumbers[mid] with the target:

- nullnumbers[0]=12
- null23>12, so search in the right half.

Iteration 4:

11. Update the start index:
    $\mathrm{start} = \mathrm{mid} + 1 = 0 + 1 = 1$
12. Recalculate the middle index:
    $\mathrm{mid} = \lfloor \frac{\mathrm{start}+\mathrm{end}}{2} \rfloor = \lfloor \frac{1+0}{2} \rfloor = 0$.
13. Compare $\mathrm{numbers[mid]}$ with the target:

- $\mathrm{numbers[1]} = 25$
- null23<25, so search in the left half.

Iteration 5 (Final Iteration):

14. Update the end index:
    $\mathrm{end} = \mathrm{mid} - 1 = 0 - 1 = -1$
15. Since $\mathrm{start} > \mathrm{end}$, the search interval is empty. The target element 23 is not found in the array.

Conclusion:

The element 23 is not present in the array `int[] numbers = {12, 25, 23, 38, 8, 62, 6, 46}`.

Do you know why this is the case?

Question **8**

Complete

Marked out of 1.00

```
Consider the following array: int[]
numbers = { 2, 5, 8, 12, 16, 23, 38,
56, 72, 91 }.
```

Perform a BINARY search to find the index of the element 23.

What is the time complexity of the binary search algorithm?

[✓ a. O(log n)]
b. O(n)
c. O(n^2)
d. O(1)

Binary search operates by repeatedly dividing the search interval in half, effectively reducing the size of the search space with each iteration.

Question **9**

Complete

Marked out of 1.00

In binary search, how does the search interval change in each iteration?

[✓ a. It is halved.]
b. It is doubled.
c. It is decremented by one.
d. It is incremented by one.

Each iteration of binary search reduces the size of the search interval by half, focusing on the subinterval where the target element is likely to be found based on comparisons. This halving process continues until the target element is found or the search interval is reduced to zero.

Question 10

Complete

Marked out of 1.00

Which of the following best describes the behaviour of binary search as the size of the input array increases?

- ○ a. Time complexity increases linearly.
- [✓ b. Time complexity increases logarithmically]
- ○ c. Time complexity remains constant
- ○ d. Time complexity increases exponentially.

As the size of the input array increases, binary search's logarithmic time complexity ensures that the time taken to search for an element grows at a much slower rate compared to the increase in array size.

Question 11

Complete

Marked out of 1.00

In binary search, what is the key requirement for the input array?

- ○ a. It must be sorted in descending order.
- [✓ b. It must be sorted in ascending order.]
- ○ c. It must contain only unique elements.
- ○ d. It must be of fixed length.

Binary search relies on the ability to efficiently divide the search space in half, which is only possible if the array is sorted

Question 12

Complete

Marked out of 1.00

In bubble sort, after completing a pass through the array, which of the following statements is true?

- [✓ a. The largest element is guaranteed to be at its correct final position.]
- b. The smallest element is guaranteed to be at its correct final position.
- c. The array is guaranteed to be sorted.
- d. The array is guaranteed to be partially sorted.

After completing a pass through the array in bubble sort, the largest element "bubbles up" to its correct final position at the end of the array. However, the entire array may not be sorted after each pass; only the largest unsorted element is guaranteed to be in its correct position.

Question 13

Complete

Marked out of 1.00

Consider an array of length n containing distinct integer values. Which of the following statements regarding the number of comparisons performed by the bubble sort algorithm is true?

- a. The number of comparisons is always exactly n.
- b. The number of comparisons is at most n-1.
- c. The number of comparisons is at least n.
- [✓ d. The number of comparisons is at most $\frac{n(n-1)}{2}$]

In bubble sort, in the worst-case scenario, the algorithm performs comparisons for every pair of elements in the array. For an array of length n, there are n elements, and each element needs to be compared with all subsequent elements. The total number of comparisons can be calculated using the formula for the sum of the first $n-1$ integers, which is $\frac{n(n-1)}{2}$.

Question 14

Complete

Marked out of 1.00

Consider the following array: array = [38, 27, 43, 3, 9, 82, 10].

Using merge sort – the first round of the merge sort algorithm will be

[38] [27] [43] [3] [9] [82] [10].

What will the second round of the merge sort algorithm be:

[✓ a. [27, 38], [3, 43], [9, 82], [10]]

b. [38, 27], [43, 3], [82, 9], [10]

c. [38,27,43],[3,9,82] , [10]

d. [3,27,38], [9,43,82,10]

In the second round of the merge sort algorithm, the pairs of elements will be merged into sorted subarrays. Starting with the initial division into individual elements, the process is as follows:

1. **Initial split**: [38], [27], [43], [3], [9], [82], [10][38], [27],[43],[3],[9],[82],[10]
2. **Second round merge**: Merge each pair of adjacent elements into sorted subarrays.

This results in:

[27, 38], [3, 43], [9, 82], [10]

Explanation:

- [38][38] and [27][27] merge to form [27, 38][27,38]
- [43][43] and [3][3] merge to form [3, 43][3,43]
- [9][9] and [82][82] merge to form [9, 82][9,82]
- [10][10] remains as a single element since it has no adjacent element to merge with at this stage

Question **15**

Complete

Marked out of 1.00

Consider an array of length n containing integer values.

Which of the following statements regarding the time complexity of the merge sort algorithm is true?

- a. Merge sort has a time complexity of O(n) in all cases.
- [✓ b.] Merge sort has a time complexity of O(n log n) in the worst-case scenario.
- c. Merge sort has a time complexity of O(n^2) in the best-case scenario.
- d. Merge sort has a time complexity of O(log n) in the average-case scenario.

Merge sort has a time complexity of O(n log n) regardless of the input distribution or content. This makes it efficient for large datasets, as it consistently performs well. In the worst-case scenario, merge sort splits the array into halves recursively until each subarray contains only one element, then merges them back together in sorted order, resulting in a time complexity of O(n log n).

Question **16**

Not answered

Marked out of 5.00

The purpose of this project is to allow you, as a 2nd year CS student the opportunity to not only implement a solution but to reflect on it.

Also, it will allow you to peer-review other solutions and critically evaluate alternative solutions.

The narrative for the question, and how and where you will upload your project, will be shared and explained to you the eTutor class.

Do NOT submit anything here - it will not be marked - you have to follow the instructions that will be explained to you by the eTutor. Just leave this space empty. The mark will be carried over from the peer-review project to this space.