Started on	Monday, 13 November 2023, 9:38 AM
State	Finished
Completed on	Friday, 24 November 2023, 1:37 PM
Time taken	11 days 3 hours
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Information

Consider the following algorithm:

```
procedure Algorithm1(list)
    let n be the length of the list
    for i = 0 to n-1 do
        for j = 0 to n-1 do
            if i != j and list[i] == list[j] then
                return true
            end if
        end for
    end for
    return false
end procedure
```

Question 1

Correct

Mark 1.00 out of 1.00

What task does the algorithm perform?

Select one:

- a. Returns true if the list contains any duplicate elements, false otherwise ✔
- b. Returns true if two lists are equal, false otherwise
- c. Returns an element that appears more than once in the list
- d. Returns an element that appears in both of two lists

Your answer is correct.

The correct answer is: Returns true if the list contains any duplicate elements, false otherwise

Question 2

Complete

Not graded

Explain why the worst-case running time of the algorithm is quadratic, i.e. O(n^2).

There are two for loops that are run. the outer loop and inner loop both have a runtime of O(n), because for each iteration of the outer loop it will fully loop through the inner for loop the runtime becomes O(n^2). As in order to loop through the outer loop (n) completely each iteration has to go through an entire other list of size n effectively making it n x n or n^2

Information

Now suppose the algorithm is changed as follows:

```
procedure Algorithm2(list)
    let n be the length of the list
    for i = 0 to n-1 do
        for j = 0 to i-1 do  // n-1 has been changed to i-1 here
            if i != j and list[i] == list[j] then
                return true
            end if
        end for
    end for
    return false
end procedure
```

Question 3

Complete

Not graded

Explain why the algorithm is still correct (i.e. still carries out the task you identified in question 1).

This algorithm will check every value in the list of length n to see if it is equal to the value of J up to the value of I in each iteration. So it still checks the entire list to see if it contains any duplicate values as it iterates, but does it by checking if J is equal to any of the previous values of i in the list. Because both i and j are using the same list, every combination of values will be tested using this method.

Question 4

Complete

Not graded

Explain why this algorithm runs approximately twice as fast (in the worst case) as the previous algorithm.

Even though both have the same runtime 0(n^2) the difference is that in the previous algorithm for each outer loop iteration the inner loop would iterate n times, whereas in this algorithm for each iteration of the outer loop the inner loop iterates i times, with i being the number of iterations of the outer loop. As such this algorithm has far less calculations to do and is much faster.

Question 5

Correct

Mark 1.00 out of 1.00

Is the time complexity of this algorithm still quadratic?

Select one:

- a. Yes ✔
- b. No

Your answer is correct.

The correct answer is: Yes

Question 6

Complete

Not graded

Explain why the time complexity is / isn't quadratic.

It's quadratic as for each iteration of the outer loop it does iterate i times, so in the end it ends up iterating the full n times for the entire length of the list. The worst case scenario is that this algorithm has to run all of the iterations for i n times and also run all the iterations for j n times making the time complexity quadratic

Information

Now let's assume the list contains items that can be sorted in order (e.g. numbers), and consider the following algorithm which performs the same task again:

```
procedure Algorithm3(list)
    let n be the length of the list
    let sortedList = Sort(list)
    for i = 1 to n-1 do
        if sortedList[i-1] == sortedList[i] then
            return true
        end if
    end for
    return false
end procedure
```

Question 7

Complete

Not graded

Explain why this algorithm only needs to check consecutive elements of the list, rather than checking every possible pair.

If the code is looking for duplicate numbers and the list is ordered from either lowest to highest or highest to lowest then duplicate values will always be together in terms of placement in the list as they the same numbers.

The duplicates will be smaller than the same amount of numbers and bigger than the same amount of numbers, so they will be consecutive/side by side in the list.

Question 8

Correct

Mark 1.00 out of 1.00

What is the time complexity of Python's built-in **sort** function?

Hint: you will need to do some online research to answer this question.

Select one:

- a. $O(n \log n)$ ✔
- b. $O(1)$
- c. $O(n)$
- d. $O(\log n)$
- e. $O(n^2)$

Your answer is correct.

The correct answer is: $O(n \log n)$

Question 9

Correct

Mark 1.00 out of 1.00

Therefore what is the overall time complexity of the algorithm above, if it is implemented in Python using the built-in **sort** function?

Select one:

- a. O(log n)
- b. O(1)
- c. O(n^2)
- d. O(n log n) ✔
- e. O(n)

Your answer is correct.

The correct answer is: O(n log n)

Question 10

Complete

Not graded

Explain your answer to question 9.

You only consider the worst runtime possible for an algorithm when determining it's time complexity, as the if statement in the code has a complexity of O(n) but instead we consider the time complexity to be O(n log n) as that is the worst runtime for anything within that algorithm,

Information

Recall the two algorithms we have seen so far:

```
procedure Algorithm2(list)
    let n be the length of the list
    for i = 0 to n-1 do
        for j = 0 to i-1 do  // n-1 has been changed to i-1 here
            if i != j and list[i] == list[j] then
                return true
            end if
        end for
    end for
    return false
end procedure


procedure Algorithm3(list)
    let n be the length of the list
    let sortedList = Sort(list)
    for i = 1 to n-1 do
        if sortedList[i-1] == sortedList[i] then
            return true
        end if
    end for
    return false
end procedure
```

Question 11

Correct

Mark 1.00 out of 1.00

If the size of the input list is very large, which algorithm is likely to run faster?

Select one:

- a. Algorithm2
- b. Algorithm3 ✔

Your answer is correct.

The correct answer is: Algorithm3

Question 12

Complete

Not graded

Explain your answer to question 11.

Algorithm 2 has a quadratic runtime, meaning the larger the input list the larger the runtime exponentially. Whereas in algorithm 3 the runtime is (n log n) this is a significantly smaller increase each time as n log n is very small in comparison to n^2.

Question 13

Complete

Not graded

Suggest **one** reason why a programmer might choose the "slower" algorithm (i.e. the algorithm you **did not** choose in question 11) over the "faster" one.

The slower algorithm could be used if the programmer wanted to use strings rather than integers as elements in the list, as the python built in sort method cannot sort lists, whereas the slower algorithm (algorithm 2) is able to check for duplicates because there is no sorting in that algorithm.

◄ Worksheet 3 Brief

Jump to...

Formative Queue for Factorio ►