Week 2 Assignments

After completing this week’s study activities, thoughtfully complete the assignments detailed below, Using the Mindtap Lab Experience resources when needed. Submit this completed document in Word format by the due date indicated in the syllabus.

Search Algorithms – part 1

Read through the background information for Lab Experience 2: Search for a Value

* 1. Open the lab simulator and select the Search Animator. Complete the activities in the Lab Experience 2 exercises #1 - #3 to gain an understanding of how the tool works and to run the tool using steps.
  2. Using the table below to complete the following:
     1. Choose the first element in the list as the target and step through the algorithm. As you step, keep a careful tally of exactly how many times each step is executed. Enter this data into the first row of the table. Compute the total number of steps executed and enter this into the table. Now reset the animator, use the second data element as the target, and enter the results into the second row of the table. Continue to do this until a distinct pattern becomes apparent. At this point, use the pattern you observe to complete the first eight rows of the table. In the last row of the table, assume that the target is the kth data element and express your entries in terms of k. For example, entries might be k or k + 1, etc. Do not fill in the entry in the Total column for the last row.
     2. Derive a formula that gives the number of steps required by the search algorithm in terms of the position of the target. To do this, simply add the entries in the bottom row of the table from and use algebra to simplify the expression obtained.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Target** | Step  1 | Step  2 | Step  3 | Step  4 | Step  5 | Step  6 | Step  7 | Step  8 | Step  9 | Step  10 | Step  11 | Total |
| Element 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element 3 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element 4 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element 5 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element 6 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element 8 |  |  |  |  |  |  |  |  |  |  |  |  |
| Element *k* |  |  |  |  |  |  |  |  |  |  |  |  |

* + 1. Formula for the search algorithm in terms of the position (k) of the target: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
    2. how many steps would you expect it to take to find the 100th element if there were that many in the list? \_\_\_\_\_\_\_\_\_

Search Algorithms – part 2

Read through the background information for Lab Experience 3: Search for the Largest Value

1. Stepping - open the Search Animator and select the Search for Largest algorithm in the Select Algorithm drop-down list. Use the Step button to execute the algorithm two or three times until you understand exactly how the algorithm works and how the animation corresponds to the steps in the pseudocode window. Remember to click Reset when you are ready to start over with a new set of data.
2. Observe that with the eight pieces of data, there are always seven passes through the loop (n - 1 if there were n pieces of data). However, the number of positions that the location marker takes on (Step 7) depends on the relative order of the data. You will now make use of the animator to explore this dependence in more depth. Using the step mode, gather data from 10 executions of the algorithm and enter the results in the table below. For each data set, copy down the entire data set and keep up with the position of the location marker and the data value at that position each time the location marker moves. For example, if the data set were 20, 15, 25, 30, 42, 18, 72, 11, then you would copy the entire list as the data set. The original position of the location marker would be position 1, and the value at that position is 20. Therefore, you would enter 1 for the location and 20 for the value. When the location marker moves, list the location number that the marker moves to and the data value there. Continue until the algorithm terminates. **Create new datasets and fill in the positions and values for each:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset 1: | 20, 15, 25, 30, 42, 18, 72, 11 |  | Dataset 6: |  |
| Positions: |  |  | Positions: |  |
| Values: |  |  | Values: |  |
|  |  |  |  |  |
| Dataset 2: |  |  | Dataset 7: |  |
| Positions: |  |  | Positions: |  |
| Values: |  |  | Values: |  |
|  |  |  |  |  |
| Dataset 3: |  |  | Dataset 8: |  |
| Positions: |  |  | Positions: |  |
| Values: |  |  | Values: |  |
|  |  |  |  |  |
| Dataset 4: |  |  | Dataset 9: |  |
| Positions: |  |  | Positions: |  |
| Values: |  |  | Values: |  |
|  |  |  |  |  |
| Dataset 5: |  |  | Dataset 10: |  |
| Positions: |  |  | Positions: |  |
| Values: |  |  | Values: |  |

1. Notice that the values stored at the positions taken on by the location marker form a subsequence of the original data set; that is, the values of the subset are taken from the original set in the same relative order as in the original set. **Explain in your own words how to scan the list from left to right and pick out the values that will be in this subsequence: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. Smallest number of positions for the location marker - **What would be the smallest number of positions that location marker could possibly point to during the execution of the algorithm?** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Describe the special conditions under which this would happen.
4. Largest number of positions for the location marker - **What would be the largest number of positions that the location marker could possibly point to during the execution of the algorithm?** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Describe the conditions under which this would happen. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pseudocode

Graphical user interface, text, application

Description automatically generated

1. What is the cost for each of the following driveways? Show how you calculated the price.
   1. Length: 50 feet, Width: 5 feet, Type: Gravel, Parking Spaces: 0
   2. Length: 100 feet, Width: 10 feet, Type: Asphalt, Parking Spaces: 2
   3. Length: 600 feet, Width: 10 feet, Type: Concrete, Parking Spaces: 8
   4. Length: 10 feet, Width: 5 feet, Type: Asphalt, Parking Spaces: 0
2. What would the pseudocode for this algorithm look like? **Write out a completed algorithm using pseudocode:**

Part 2 - Reading Pseudocode

Text

Description automatically generated

Answer the following questions.

1. What is output of this pseudocode for the following amounts?
   1. 14
   2. 25
   3. 99
2. What does this pseudocode do? Write a brief explanation in English of the algorithm.
3. This algorithm uses many loops and steps. Is there a less complicated way to write this algorithm? Create a different pseudocode for this algorithm that is shorter and more precise.

Flowcharts

Flowcharts are a graphical representation of a process or algorithm. A flowchart can be an excellent tool to complement pseudocode for understanding how an algorithm maps out steps to solve a problem. For a better grasp on why flowcharts are used and some of the basic flowchart symbols, check out the following short article on flowcharts: <http://www.mindtools.com/pages/article/newTMC_97.htm>

There are many different software programs that allow you to draw flowcharts. One freely available graph editor is yEd (<http://www.yworks.com/yed>). yEd is available for all major operating systems. If you are interested in using yEd for flowcharts, here is a brief tutorial showing you the basics of flowchart creation in yEd: <http://www.youtube.com/watch?v=ujMhxPJnJCw>

For this assignment, you will be a making a flowchart for an algorithm to determine the winner of a common two-player game: Rock, Paper, Scissors (RPS). The rules for RPS are listed below.

Diagram

Description automatically generated

Rules:

* On the count of 3, each player chooses Rock, Paper, or Scissors and shows their choice to their opponent.
* Rock blunts Scissors, so the player showing Rock would win in this case.
* Paper wraps Rock, so the player showing Paper would win in this case.
* Scissors cut Paper, so the player showing Scissors would win in this case.
* If both players choose the same item, the round is a draw and the game is played again.

Insert Flowchart here:

Answer the following questions.

1. What would an algorithm to check the outcome of a single round of RPS look like? Create a flowchart to get the move from each player and determine the appropriate outcome.
2. What additional logic would you need to add to the flowchart to modify the game to require the winner of the game to win three hands? Update your flowchart to include this new rule.

Sort Algorithms:

Using the Sort Animator and associated Unit 3 Mindtap lab exercises four and six to become familiar with the four types of presented sorting algorithms and their respective efficiencies, provide a comparison of the sort algorithms describing how they work and how they differ from each other:

|  |  |  |
| --- | --- | --- |
| Sorting Algorithm | How does it work? | How does it differ from the others/timing? |
| Insertion Sort |  |  |
| Selection Sort |  |  |
| Bubble Sort |  |  |
| Quick Sort |  |  |

Data Cleanup Algorithms:

Using the Data Cleanup Animator and associated lab exercise in Mindtap Unit 3, compare the three types of data CleanUp algorithms, stating how they are the same and how they are different:

Binary Search

Open the **Sort Animator**. Click **Reset** to obtain new data in the array. Then pick **Binary Search** from the **Select Algorithm** drop-down list.

Take the appropriate steps to make the array ready to search and step through the binary search algorithm. Notice that, each time through the loop, the number at the mid marker is compared with the target number. This happens each time mid takes on a new value. Therefore, we can count the comparisons by counting the number of positions that mid takes on. Using the step mode, first choose the first element in the list as the target and record the number of positions that mid takes on. Now repeat this, choosing the second element in the list as the target, and continue for each element in the list.

Complete the following table:

|  |  |
| --- | --- |
| What happens when you first try to choose Binary Search? | |
| Why should this happen? | |
|  |  |
| Number of comparisons with first number as target: |  |
|  |  |
| Number of comparisons with second number as target: |  |
|  |  |
| Number of comparisons with third number as target: |  |
|  |  |
| Number of comparisons with fourth number as target: |  |
|  |  |
| Number of comparisons with fifth number as target: |  |
|  |  |
| Number of comparisons with sixth number as target: |  |
|  |  |
| Number of comparisons with seventh number as target: |  |
|  |  |
| Number of comparisons with eighth number as target: |  |
|  |  |
| Largest number of comparisons: |  |
|  |  |
| Smallest number of comparisons: |  |
|  |  |
| Average number of comparisons: |  |
|  |  |
| Largest number of comparisons if *n* = 16: |  |
|  |  |
| Largest number of comparisons if *n* = 1024: |  |

This assignment will be graded using the following rubric:

|  |  |  |  |
| --- | --- | --- | --- |
| EXEMPLARY  93% - 100% | PROFICIENT  80% - 92% | PROGRESSING  70% - 79% | NOT YET  < 69% |
| Required components are correctly and substantively addressed. | Required components are generally addressed in a proficient manner. | Required components are addressed in a limited or imprecise manner. | Assignment is vague in addressing the specified task, is poor in quality, and/or is missing key components. |