TSSM™
Creating VCE Success

Software Development Teach Yourself Series

Topic 7: Sorting and Searching Algorithms Units 3&4

A: Level 14, 474 Flinders Street Melbourne VIC 3000
T: 1300 134 518 **W:** tssm.com.au **E:** info@tssm.com.au

Contents

CASE STUDY

All Teach Yourself Series in this package will refer to the following case study.

Tariq Mulner is the manager of a school canteen. He manages how many lunches are going to be prepared each day. It is difficult to tell how many lunches will be sold, so he would like a software solution that students can use to order lunches. This application would provide him with a complete list of orders.
Most of the students have smart phones, so Tariq is suggesting the solution is a phone app that can read in the lunch order, and send it to his device so he can print out the order list.

Photo by Tirachard Kumtanom from Pexels used with permission

Algorithms

An algorithm is a solution to a logic problem. In the Study Design we use Pseudocode to write algorithms.

The Features of Pseudocode

- Every algorithm must begin and end with START and STOP or BEGIN and END.
- When assigning data to a variable use ←. Example: IntNumber ← 5
- Show each calculation required. Example: IntAnswer ← IntNumber1 + IntNumber2
- Displaying data as output Example: lblAnswer.Text ← IntAnswer or Display IntAnswer
- Decision Control Structures: Example: IF condition THEN Action1 ELSE Action2 ENDIF
- Case Control Structures:
  Example: Case where condition, CASE1 Action1 CASE2 Action2 END CASE
- Repetition Control Structures: Example: FOR counter ← first value to last value END FOR
- Repetition Control Structures: Example: REPEAT actions UNTIL condition
- Repetition Control Structures: Example: WHILE condition DO actions END WHILE
- Use indentation and spaces to illustrate how control structures are incorporated

Here is an example of a pre-test loop allowing three attempts at entering a username and password.

```
START
        strUsername ← BillBurr123
        strPassword  ← Summer1965
        Read in strUsernameEntered
        Read in strPasswordEntered
        Counter ← 0
        WHILE strUsername <> strUsernameEntered AND strPassword <> strPasswordEntered AND Counter < 3
                Display Prompt "Your Username and Password are incorrect please try again!"
                Read in strUsernameEntered
                Read in strPasswordEntered
                Counter ← Counter +1
        END WHILE
        IF Counter < 3 THEN
                IF strUsername == strUsernameEntered AND strPassword == strPasswordEntered
                        Display Prompt "Your Username and Password are correct!"
                ELSE
                        Display Prompt “You have exhausted your attempts. Please contact SysAdmin!”
                END IF
        END IF
END
```

Arrays

In Software Development we use Arrays to clearly demonstrate how data can be manipulated with algorithms. Array data has an index that allows us to access the data and relocate it if necessary.

Array Example: Names(6) holds 6 names.

0	1	2	3	4	5
David	Daphne	Doug	Delphi	Daniel	Dobby

Searching

In Software Development we investigate two search algorithms.

1. Linear Search
2. Binary Search

Linear Search

The simplest search algorithm is Linear. It basically starts at the very first item, check if it is the data we are looking for, if it isn't, it checks the next one. This continues until the item is found or until end of file (EOF).

Advantages

- Easy to program
- Good for use with small files

Disadvantages

- Inefficient
- Takes too long to find data in large files

We are going to write a linear search for Delphi in Names(6).

0	1	2	3	4	5
David	Daphne	Doug	Delphi	Daniel	Dobby

```
START
        Counter ← 0
        Found ← False
        WHILE Found = False AND Not EOF DO
                If Names (Counter) <> Delphi THEN
                        Counter ← Counter + 1
                ELSE
                        Found ← TRUE
                END IF
        END WHILE
END
```

Trace Table		
Counter	Found	Names(Counter)
0	False	David
1	False	Daphne
2	False	Doug
3	True	Delphi

The number of items in the array is 6 (n). The worst case for finding any item with linear is if the item is the last in the array. This means if Delphi was in position 5 it would take 6(n) operations or repeats through the loops to check each item.
This is written $O_{(n)}$.

Binary Search

Binary is more complex yet, more efficient search algorithm. There is one limitation to this algorithm, it must be sorted or order in ascending or descending order. Binary search breaks up the list of data into halves, tests each half for the item searched and discards the half that does not contain the searched item.

Below we have ordered Names(6) into ascending order so it can be searched with a binary search.

0	1	2	3	4	5
Daniel	Daphne	David	Delphi	Dobby	Doug

The process of the binary search is identifying the highest and lowest of the list, then finding the middle item. The searched for item will be compared with the middle item to see if it is higher or lower than the middle item. Let's look for Daphne!

```
START
        Lowest← 0
        Highest← 5
        Found ← False
        WHILE Found← False AND Not EOF DO
                Middle ←( Highest – Lowest ) /2

                IF Names(Middle) = Daphne THEN
                        Found ← True
                ELSE IF Names(Highest) = Daphne THEN
                        Found ← True
                ELSE IF Names(Lowest) = Daphne THEN
                        Found ← True
                ELSE IF Daphne > Names(Middle)THEN
                        Lowest ← Middle + 1
                ELSE   ** Daphne < Names(Middle)
                        Highest ← Middle - 1
                END IF
        END WHILE
END
```

Integer division ignores the fractionalamount

Trace Table						
Found	**Lowest**	**Highest**	**Middle**	**Names(Lowest)**	**Names(Highest)**	**Names(Middle)**
False	0	5	2	Daniel	Doug	David
	0	1	0	Daniel	Daphne	Daphne
True						

The advantages of using a binary search is that it uses less operations than linear search ($O_{(logn)}$) and can be used for larger lists of items. However, it is more complex to code.

Sorting

Many programming languages have a sort function that uses built in functioning to sort your data for you. In this study we need to demonstrate to be able to use control structures to sort our data. We focus on two algorithms:

1. Selection Sort
2. Quick Sort

Selection Sort

This sorting algorithm is the easier of the two to program, but the least efficient. The key information you need to know about Selection Sort is that it PASSES through the list looking for the lowest item. Once found, it SWAPS the lowest item with the first item of the unsorted list.

Below is an algorithm for sorting an Array of unknown length.

```
BEGIN
    FOR PassCounter: 1 to (ArrayLength - 2) DO
        Smallest ← ArrayList(PassCounter - 1)
        SmallestIndex ← PassCounter
        FOR ItemCounter: PassCounter to ArrayLength - 2 DO
            IF ArrayList(ItemCounter) < Smallest THEN
                Smallest ← ArrayList(ItemCounter)
                SmallestIndex ← ItemCounter
            END IF
        END FOR
        Temp ← ArrayList(PassCounter - 1)
        ArrayList(PassCounter - 1) ← ArrayList(SmallestIndex)
        ArrayList(SmallestIndex) ←Temp
    END FOR
    Display ArrayList()
END
```

Quick Sort

This algorithm is complex, but more efficient. The key things you need to know about Quick Sort algorithms are the following features:

- Divide and Conquer approach
- Uses a PIVOT
- Uses Recursion

```
START FUNCTION: QuickSort
BEGIN
    IF ArrayLength >1 THEN
        Pivot ← ArrayList(0)
        StartIndex ← 0
        EndIndex ← ArrayLength
        WHILE StartIndex <= EndIndex DO
            WHILE ArrayList(StartIndex) < Pivot DO
                StartIndex ← StartIndex + 1
            END WHILE
            WHILE ArrayList(EndIndex) > Pivot DO
                EndIndex ←EndList - 1
            END WHILE
            IF StartIndex < = EndIndex THEN
                Temp ← ArryList(StartIndex)
                ArrayList(StartIndex) ← ArrayList(EndIndex)
                ArrayList(EndList) ← Temp
                StartIndex ← StartIndex + 1
                EndIndex ←  EndIndex - 1
            End IF
        End While
        Function QuickSort (0 to EndIndex)
        Function QuickSort(StartIndex to ArrayLength)
    END IF
END
```

Quick sort selects an item (for this instance we will identify the first item) as the PIVOT. All the items lower than the pivot are moved to the left of the pivot. This is called a PASS. At the end of a PASS, the PIVOT is in the correct location. Everything on the left of the PIVOT is now a new list with all values less than the PIVOT. Everything on the right of the PIVOT is now a new list with all values greater than the PIVOT. Both of these lists are now quicksorted with new PIVOTs until all the items are in their correct place.

Review Questions

Applied Questions

Tariq has asked you to produce an app. The Analysis is available in TOPIC 1. You have 3 weeks to develop the app to hand over. Ideally Tariq should have the solution for 2 weeks before evaluation begins.

Tariq has some data that needs searching and sorting

Students(9)

0	1	2	3	4	5	6	7	8	9
Julian	Michael	Suzie	Veronica	Macie	Aaron	Christian	Lachlan	Betty	Archie

Products(6,1) – the first row shows the names of the products, the second row the average sales per day.

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	Beef Pies	Egg Sandwiches	Hamburgers	Hot Chips	Hot Dogs	Sausage Rolls	Tacos
1	34	23	7	21	8	19	24

1. Students(9) is an array that holds 10 student names. Which algorithm is best used to search for Betty?

A. Quick
B. Linear
C. Selection
D. Binary

2. If we used Binary Search to find Sausage Rolls, how many operations would it take?

A. 4
B. 5
C. 1
D. 2

3. How many operations would it take to find Macie if we used a Linear Search?

A. 9
B. 4
C. 1
D. 5

4. Products(6,1) is sorted by product name. Which algorithm is best used to search this list?

A. Selection
B. Linear
C. Binary
D. Quick

5. Create a TRACE TABLE to find out how many Hamburgers are sold each day using a Binary Search.

Trace Table						
Found	**Lowest**	**Highest**	**Middle**	**Products(Lowest,0)** **Products(Lowest,1)**	**Products(Highest,0)** **Products(Highest,1)**	**Products(Middle,0)** **Products(Middle,1)**

6. Students(9) is unsorted. We need to use Quick Sort so we can prepare for a larger list of students that will need to be imported when the application is implemented.

Students(9)

0	1	2	3	4	5	6	7	8	9
Julian	Michael	Suzie	Veronica	Macie	Aaron	Christian	Lachlan	Betty	Archie

We have completed TWO passes using the FIRST item as the PIVOT.

PIVOT

0	1	2	3	4	5	6	7	8	9
Julian	Michael	Suzie	Veronica	Macie	Archie	Christian	Lachlan	Betty	Aaaron

0	1	2	3	4	5	6	7	8	9
Archie	Christian	Betty	Aaron	Julian	Michael	Suzie	Veronica	Macie	Lachlan

The Pivot item moves to the correct location with all the items below it moved to the left, leaving the rest on the right. They remain in the order they were in.

Now we have TWO Separate lists.

PIVOT

0	1	2	3
Archie	Christian	Betty	Aaron

Fill in the PASS below with the PIVOT in the correct location.

0	1	2	3

TSSM
Creating VCE Success

Solutions to Review Questions

1. B. Linear

2. D. 2

3. D. 5

4. C. Binary

5. Create a TRACE TABLE to find out how many Hamburgers are sold each day using Binary Search.

Trace Table						
Found	**Lowest**	**Highest**	**Middle**	**Products(Lowest,0)** **Products(Lowest,1)**	**Products(Highest,0)** **Products(Highest,1)**	**Products(Middle,0)** **Products(Middle,1)**
False	0	6	3	Beef Pies 23	Tacos 24	Hot Dogs 8
False	0	2	1	Beef Pies 23	Egg Sandwiches 23	Hamburgers 7
True						

6. Second PASS.

0	1	2	3
Aaron	Archie	Christian	Betty