



## **Searching Algorithms**

#### **Linear Search**

- 1. Check the first value
- 2. If it is desired value
  - Stop
- 3. Otherwise check the second value
- 4. Keep Going until all elements have been checked or the value is found

## **Binary Search**

- 1) Put the list in order.
- 2) Take the middle value.
- Compare it to the desired value.
  - a) If it is the desired value.
    - i) Stop.
  - b) If it is larger than the desired value.
    - i) Take the list to the left of the middle value.
  - If it is smaller than the desired value.
    - i) Take the list to the right of the middle value.
- 4) Repeat step 3 with the new list.

#### **Trace Tables**

- Tests algorithms for logic errors which occur when the algorithm is executed.
- Simulates the steps of algorithm.

• Each stage is executed one at a time allowing inputs, outputs, variables, and processes to be checked for the correct value at each stage.

	Stage	X	Υ	Output
<pre>X = 3 Y = 1 while X &gt; 0</pre>	1	3	1	
	2		2	
	3	2		
	4		3	
	5	1		
	6		4	
	7	0		
	8			4

# Unit 1 – Fundamentals of Algorithms

### **Key Terms**

- Abstraction
- Using symbols and variables, to represent a realworld problem with a computer program.
- Removing unnecessary elements
- Example a program is to be created to let users play chess against the computer.
  - Board is created as an array(s).
  - Pieces are objects that have positions on the
  - The shape and style of the pieces may not be required.
- Decomposition
- Breaking down large problems into a set of smaller
  - Smaller problems are easier to solve
  - Each part can be solved independently
  - Each part can be tested independently
  - The parts are combined to produce the full problem.
- There are usually several different approaches, and not one single right way to do this.

#### What is an Algorithm

- An algorithm is a series of steps which can be followed to complete a task.
- A computer program may use an algorithm.
- A computer program and an algorithm are not the same thing.
- Algorithms help to work out the steps needed to solve a given problem.
- This helps us plan how to write a computer program.
- An algorithm will always finish and return an answer or perform a series of tasks that it was supposed to.

## **Sorting Algorithms**

#### **Bubble Sort**

- 1) Take the first element and second element
- 2) Compare the two
  - a) If element 1 > element 2
    - i) Swap then
  - b) Otherwise
    - i) Do nothing
  - c) Move to the next pair in the list
  - d) If there are no more elements return to step (1)
  - e) Otherwise, return to step (2)
- Repeat until you have worked through the whole list without making any changes

#### Merge Sort

- 1) Split the list into individual elements.
- 2) Merge the elements together in pairs, putting the smallest element first.
- 3) Merge two pairs together, putting the smallest first.
- 4) Keep merging until all pairs are in order.

Inefficient and slow

#### **Comparing Algorithms Linear Search Binary Search** Pros Works with unsorted lists More efficient Not affected by changes to the list Efficient for large lists · Works well for small lists Cons Slower Does not work with unsorted • Inefficient for large lists lists **Bubble Sort** Merge Sort • Far more efficient and faster Simplest and easiest to code Uses less memory Consistent running time Cons Slower with larger lists • Uses more memory

More complexed to program

## **Algorithm Efficiency**

- Several different algorithms can solve the same problem
- Efficiency allows us to compare two different algorithms that solve the same problem.
- A more efficient algorithm is a better
- The guicker the algorithm can complete its task, the more efficient it is.
- For example, an algorithm that can be executed in 10 instructions, is more efficient than one which takes 25 instructions.

## **Determining The Purpose of Algorithms**

- There are several ways to determine the purpose of an algorithm.
- We can dry run the algorithm, by assigning values to its inputs, and working through to see what happens.
- Trace Tables allow us to record these values as the algorithm is run.
- Visual Inspection involves simply looking at the algorithm to determine its purpose.
- Sometimes the algorithm may follow a standard pattern which we can recognise.
- With shorter or simpler algorithms, the purpose may be obvious by simply looking at it.

## **Pseudocode**

- Uses short English words and statements to describe an algorithm.
- Generally looks a little more structured than normal English sentences.
- Flexible.
- Less precise than a programming language.

IF Age is equal to 14 THEN Stand up ELSE Age is equal to 15 THEN Clap ElSE Age is equal to 16 THEN Sing a song ELSE Sit on the floor END

## An Example Algorithm

This algorithm, written in pseudocode, follows a simple pattern for working through each letter of an input to determine if it matches a predefined word. This might form part of a hangman game

guess ← USERINPUT FOR  $i \leftarrow 0$  TO LEN(word) IF word[i] = guess THEN **OUTPUT** "found" **ENDIF ENDFOR**