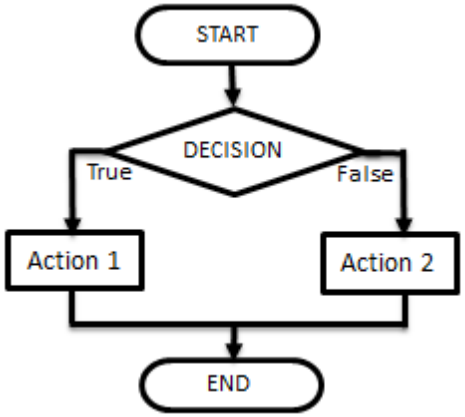
2.1 ALGORITHMS

COMPUTATIONAL THINKING

Abstraction	•Focussing on just the important details of a problem
Decomposition	•Breaking a problem down into smaller parts so that it is easier to solve
Algorithmic thinking	•creating a step by step solution to a problem

FLOWCHART



PSEUDOCODE

```
START
IF the Decision = TRUE THEN:
   Perform Action 1
ELSE
   Perform Action 2
ENDIF
END
```

- Universal language, planning phase before actual coding in for e.g. python
- Work out where you need inputs, outputs, decisions, loops and variables.

SEARCHING ALGORITHMS

To find an item in a list, computers need to use a searching algorithm. A linear search and binary search are both examples of sorting algorithms.

Linear Search: Checks each item in the list one by one until it finds what it is looking for
+ Simple, list doesn't need to be ordered
- Not efficient, takes time with lots of data

Binary Search: Finds the middle item in an ordered list by doing (n+1)/2. IF the middle item is what it is searching for it stops. If not, it compares the item you are searching for to the middle item so that it knows whether to look in the first half or second half of the list. Then it repeats these steps until the item is found
+ More efficient than a linear search
- Only works on an ordered list, complex to program

SORTING ALGORITHMS

Sorting algorithms sort items into an ordered list.

Bubble Sort: Checks the first two items in a list, swaps them if they are in the wrong order and then moves onto the next two items and repeats the process. Once it has passed through the list once it goes through again until none of the items need swapping. + Simple. – Takes a long time

Merge Sort: Finds the middle item (n+1)/2 and splits the list in half. Repeats this step until the list is split into individual items (sub-lists). It them merges (joins) the sublists in pairs. Each time the sublists are paired they are sorted into the correct order. + Efficient – Slow

Insertion Sort: Looks at the second item in a list and compares it to the items that are in front of it, then inserts it into the right place. It then moves to the next item in the list and repeats these steps. + Quick for sorting small lists – slow with long lists