Algorithms

Keyword	Definition
Algorithm	A series of instructions to solve a problem
Decomposition	Breaking down a large problem into smaller more manageable ones
Abstraction	Removing unimportant parts of a problem in order to concentrate on those that are important
Sequence	A series of steps which are completed one after the other
Selection	The ability to chose different paths through a program
Iteration	Repeating a part of a program

Arithmetic Operators

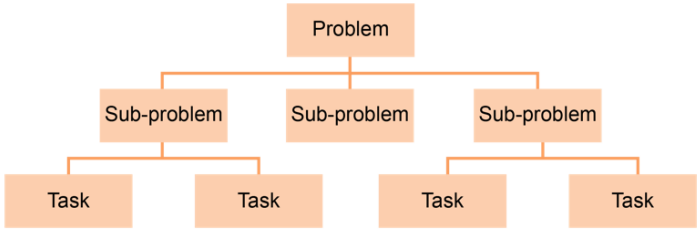
Symbol	Description	Example
+	Add	5+7 = 12
-	Subtract	5-7 = -2
/	Divide	15/10 = 1.5
*	Multiply	5*7 = 35
^	Exponent	5^2 = 25
MOD	Modulo (Remainder)	17 MOD 3 = 2
DIV	Integer division	17 DIV 3 = 5

Abstraction

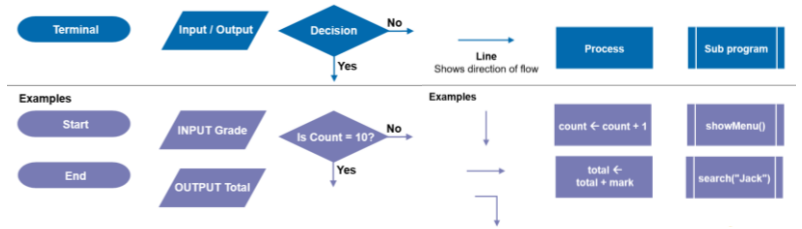
The London underground map is a good example of abstraction

Unnecessary detail such as the distance between the stops, the curvature of the track and the depth of the track have all been removed

Decomposition



Flowchart Symbols



Data Types

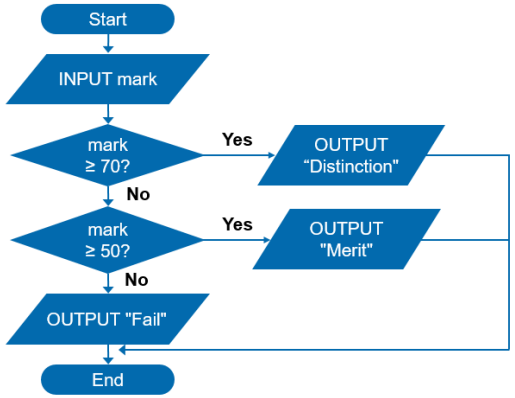
Data type	Description	Example
INTEGER	a whole number	1475, 0, -5
REAL	a number with a decimal point	56.75, 6.0, -2.456, 0.0
BOOLEAN	Can only be TRUE or FALSE	TRUE, FALSE
CHARACTER	A single alphabetic or numeric character	'a, 'K', '4', '@', '%'
STRING	One or more characters enclosed in quote marks	'Jo Hobson', '123'

Boolean Operators

AQA	Meaning	Python/C#	VB
>	greater than	>	>
≥	greater than or equal to	>=	>=
<	less than	<	<
≤	less than or equal to	<=	<=
=	equal to	==	=
≠	not equal to	!=	<>

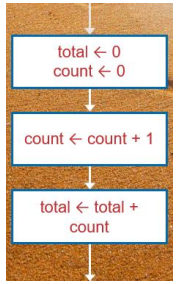
Algorithms

Flowchart or Pseudocode




```
Mark ← USERINPUT
IF mark >= 70 THEN
   OUTPUT "Distinction"
ELSE IF mark > 50 THEN
   OUTPUT "Merit"
ELSE
   OUTPUT "Fail"
ENDIF
```

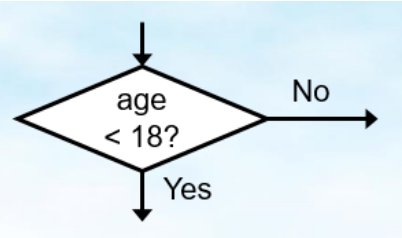
Sequence



Order of execution

```
mealCost ← 4.00
drinkCost ← 2.00
total ← mealCost + drinkcost
```

Selection

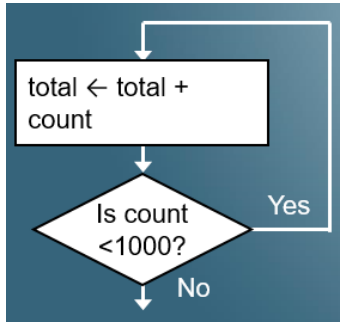



```
OUTPUT "How many hours a night do you sleep?"
hoursPerNight ← STRING_TO_INT(USERINPUT)
IF hoursPerNight < 8 THEN
      OUTPUT "That's not enough!"
ELSE
      OUTPUT "That's plenty!"
ENDIF
```

Execute this if TRUE

Execute this if FALSE

Iteration – WHILE loop



Most common condition controlled loop

```
password ← ""
WHILE password ≠ "rE5Bh9dP"
   password ← input("Enter password")
ENDWHILE
OUTPUT "Correct password"
```

Repeat while password is not equal to "rE5Bh9dP"

Runs when password equals "rE5Bh9dP"

Iteration – FOR loop

```
total ← 0
FOR counter ← 1 TO 7
   maxTemperature ← USERINPUT
   total ← total + maxTemperature
ENDFOR
averageWeeksTemp ← total / 7
OUTPUT INT_TO_STRING(averageWeeksTemp)
```

Repeat the loop 7 times

Counter controlled loop, use when you know how many times you need to loop round

Iteration – REPEAT UNTIL loop

```
password ← ""
REPEAT
   OUTPUT "Enter password"
   password ← USERINPUT
UNTIL password = "rE5Bh9dP"
OUTPUT "Correct password"
```

Condition controlled loop. This loop is executed at least once

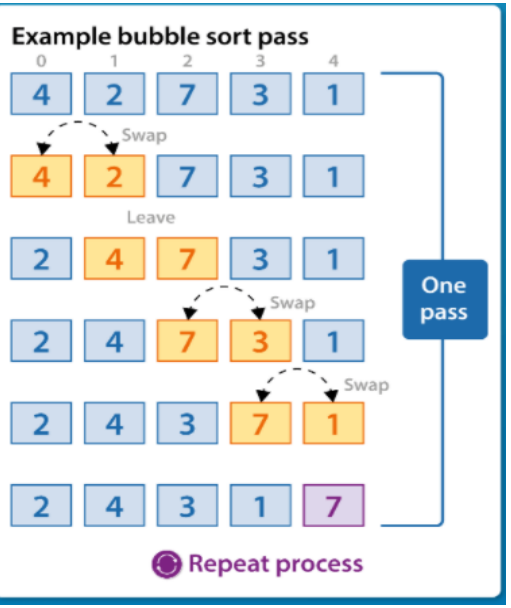
Algorithms

Bubble Sort

One pass

1. Compare the first two items in the list. If they are in order, leave them as they are. If items are not in order, swap them. In this example 4 is larger than 2 so they need to be swapped.
2. Repeat step 1 for items 2 and 3 in the list. In this example 4 is smaller than 7 so they do not need to be swapped.
3. Continue applying step 1 to the rest of the items in the list.

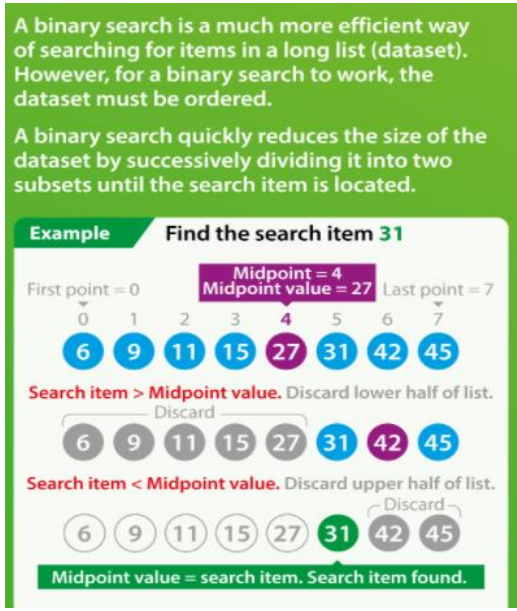
Once one pass is completed, repeat the process again and again until no swaps are required and all the numbers are in order.



Binary Search

A binary search is a much more efficient way of searching for items in a long list (dataset). However, for a binary search to work, the dataset must be ordered.

A binary search quickly reduces the size of the dataset by successively dividing it into two subsets until the search item is located.

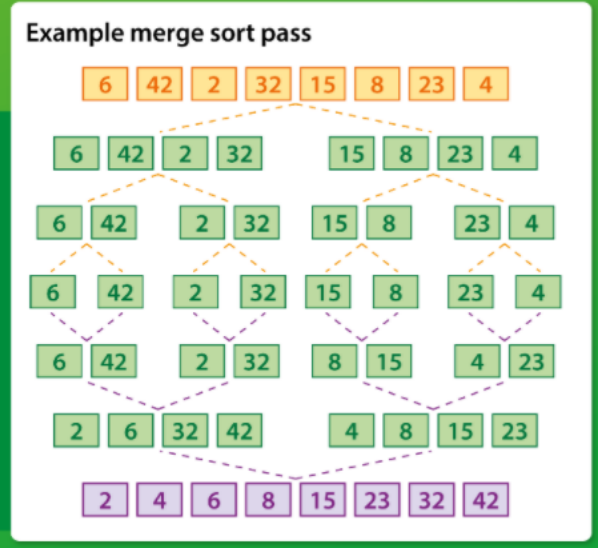


1. Find the midpoint of the dataset:
   Midpoint = (First + Last) ÷ 2
   = (0 + 7) ÷ 2
   = 3.5 ← round up to 4
2. If the search item is the midpoint value then the search item has been found.
   31 > 27 so the search item has not been found.
3. If the search item is more than the midpoint value, discard the lower half of data and search for the item in the upper half of the data.
   31 > 27 so discard the lower half of the data.
4. If the search item is less than the midpoint value, discard the upper half of data and search for the item in the lower half of the data.
   31 < 42 so discard the upper half of the data.
5. Repeat steps 1–4 until the search item is located.
   Midpoint value = 31 so search item has been found.

Merge Sort

A merge sort is a 'divide and conquer' algorithm that splits a list into discrete elements and then merges the elements together in order.

1. Split the list in half to create two subsets.
2. Continue to split the subsets in half, until only individual items remain.
3. Merge individual items together in pairs, putting them back together in order.
4. Continue to merge the pairs together, with each subset being sorted in order.
5. Once all subsets have been combined, the list should be in order.



Linear Search

A linear search looks at each item in a list sequentially (one item at a time) until the desired search item is located or until all the items have been searched.

Linear searches are relatively simple, but they are inefficient when searching through long lists.