Searching Algorithms

Linear Search Algorithm

- The purpose of the linear search algorithm is to find a target item within a list.
- Compares each list item one-by-one against the target until the match has been found and returns the position of the item in the list.
- If all items have been checked and the search item is not in the list then the program will run through to the end of the list and return a suitable message indicating that the item is not in the list.
- The algorithm runs in linear time. If *n* is the length of the list, then at worst the algorithm will make *n* comparisons. At best it will make 1 comparison and on average it will make (*n+1)*/2 comparisons.
- The performance of the algorithm will be improved if the target item is near the start of the list.

Example

Find the position of letter "Z" within the following list. Assume we do not have visibility of the list

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
position								
Value	V	A	S	Z	X	R	T	G

We compare it with the value in index position 0. We find that the value is "V" so we need to move on to the next index position. At index position 1 and 2 we still have not found Z. However, we get to index position 3 and we compare the target with the value and we find that they match, so the algorithm returns the index position and stops.

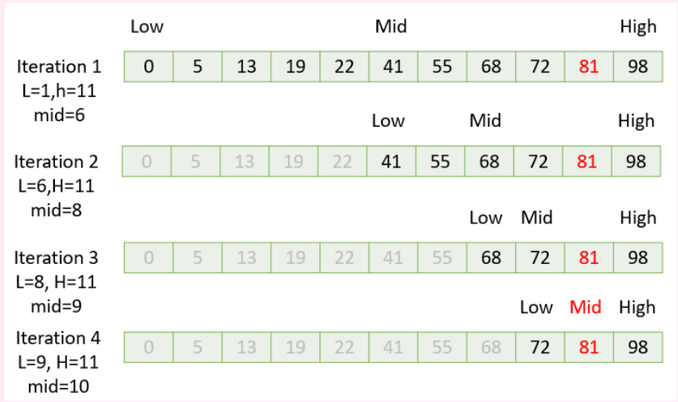
Pseudocode

```
i ← 0
x ← len(listOfItems)
pos ← -1
found ← False
WHILE i < x AND NOT found
 IF listOfItems[i] == itemSearch THEN
  found ← True
  pos ← i + 1
 ENDIF
 i=i+1
ENDWHILE
OUTPUT pos
```

Binary Search Algorithm

- The binary search algorithm works on a sorted list by identifying the middle value in the list and comparing it with the search item.
- If the search item is smaller the mid element becomes the new high value for the search area.
- If the search item is larger the mid element becomes the low value for the search area.
- The keeps repeating until the search item is found.
- When the search item is found the index position of the item is returned.
- At each iteration the search are halved in size consequently this is an efficient algorithm.

Example: Binary search in operation to find 81



Pseudocode

```
low ← 1
high ← LENGTH(arr)
mid ← (low + high) DIV 2
WHILE val ≠ arr[mid]
 IF  arr[mid] < val THEN
  low ← mid
 ELIF arr[mid] > val THEN
  high ← mid
 ENDIF
  mid ← (low + high) DIV 2
  ENDWHILE
OUTPUT mid
```

Linear search versus binary search

	Advantages	**Disadvantages**
Linear Search	• Very simple algorithm and easy to implement • No sorting required • Good for short lists	• slow because it searchers through the whole list • very inefficient for long lists
Binary Search	• much quicker than linear search, because it halves the search zone each step	• The list need to be ordered