Linear Search

The linear search is the simplest, but also the slowest searching method. It does not depend on any particular data organization

Consider how to find a phone number in an unalphabetized phone book!

```
int a[MAX];

i = 0;
while(i < MAX && a[i] != targetval)
    i++;
```

Performance of Linear Search

A linear search could take as few as one iteration, or as many as n iterations.

On average, $\frac{n}{2}$ iterations are required to find an item in the list. If the item is not in the list, it will require n iterations to find out.

We say that a linear search is of order n, O(n)

"Big Oh"

Binary Search

If the list is ordered (ie., sorted), we can improve the search.

Procedure:

1. Look at element n/2. Item will either be in the lower half or upper half. Throw away irrelevant half

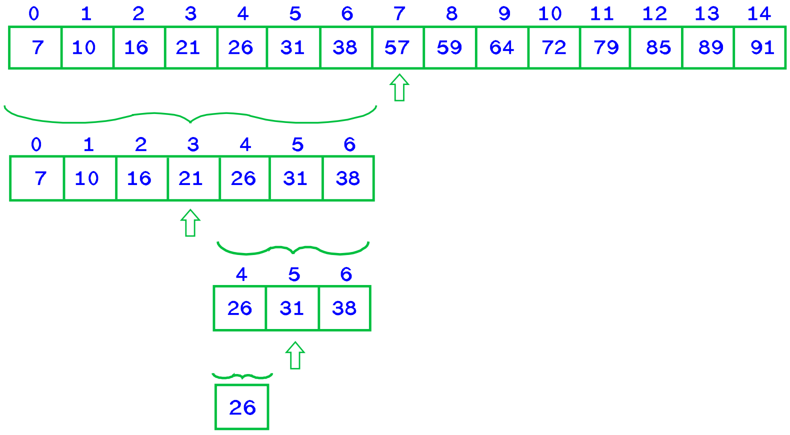
2. Now look at the middle element of the remaining half; throw away the irrelevant quarter.

3. Repeat the process until the remaining section consists of only one element. If this element == item, then item is in list, else item is not in list.

This process is similar to how we use a phone book!

Binary Search Example

Is the number 27 in the list?



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
7	10	16	21	26	31	38	57	59	64	72	79	85	89	91

0	1	2	3	4	5	6
7	10	16	21	26	31	38

4	5	6
26	31	38

26

Conclusion: Not in list

Performance of Binary Search

The size of list that needs to be searched decreases by half with each iteration. Each iteration requires one comparison. So, at most $log_2 n$ comparision are required to find whether a value is in the list.

We say that a binary search is of order $log_2 n$, O(log n)