Why Algorithms Matter ?

An `algorithm` is a set of instructions for completing a specific task.

An algorithm is the set of instructions for the computer to achieve a particular task. When we write any code, then, we're creating algorithms for the computer to follow and execute.

Ordered Arrays

Every time a value is added, it gets placed in the right ***order*** such that array remains sorted.

```
array = [21, 34, 44, 52]
```

Now if we wish to insert 38 in the array.

- Check the value at index 0, here it is 21, and `21 < 38`. So it needs to be inserted at the right side.
- Check the value at index 1, here it is 34, and `34 < 38`. So again it need to be inserted at the right side.
- Check the value at index 2, here is is 44, and `44 > 38`. So 38 needs to be inserted at the current position, that is index `2`.
- Now we shift all the elements to the right and make space for 38.
- At last we insert 38 at index 2. Thus maintaining the order.

we need to ***always conduct a search before the actual insertion to determine the correct spot for the insertion***. For N elements in an ordered array, the insertion took `N + 2` steps in total.

If our value ends up toward the beginning of the ordered array, we have fewer comparisons and more shifts. If our value ends up toward the end, we get more comparisons but fewer shifts.

Searching An Ordered Array

```
array = [34, 45, 55, 100]
```

Here in case of `Linear Search` if the search value is 46, even though it is not present in the array, we need to search for it until the end.

But in case of ordered array, we can stop early, at 55 we can be sure that 46 doesn't contain in the array, since it's impossible for the 46 to be anywhere to the right of it.

`Linear Search`

```
int linearSearch(int l[], int n, int target){
  for(int i = 0; i < n; i++)
    if(l[i] == target)
      return i;
  return -1;
}
```

This is the `C Implementation` of the linear search, if `target` is found it returns it's index, or else it returns `-1`. For `N` elements it takes `N` steps.

Binary Search

Binary Search is another search algorithm that works on ordered array.

`array = [34, 45, 55, 100, 122, 144, 159]`

Now let's say we want to find element 45.

- Calculate the middle index, here it is 3.
- Check the value at index 3, it is 100, target is 45 (`45 < 100`). So we know that 45 lies to the left of the 100. Thus we eliminate half of the array elements which are at the right side of 100.
- Now again find the middle index here it is 1.
- Check the value at index 1, it is 45, target is 45 (`45 == 45`). Hence element is found and we return the index.

If the target was 46, and once we reach at the index 1, `45 < 46` so 46 should be at the right side of the 45, but then we have only one element that is `55`, so we conclude that 46 is not present in the ordered array.

```c
int binarySearch(int l[], int n, int target)
{
  int lower = 0;
  int upper = n - 1;

  int middle = 0;
  int value = 0;

  while (lower <= upper)
  {
    middle = lower + (upper - lower) / 2;
    value = l[middle];

    if (value == target)
      return middle;
    else if (value < target)
      lower = middle + 1;
    else if (value > target)
      upper = middle - 1;
  }

  return -1;
}
```

This is the `C Implementation` of the binary search, if `target` is found it returns it's index, or else it returns -1. For `N` elements it takes `N` steps.

Binary Search Vs Linear Search

With order arrays of small size, the binary search doesn't has much advantage. But for larges sizes it is the most efficient one.

For an ordered array containing 100 elements, linear search takes `100 Steps`, and binary search takes `7 steps`.

When we use binary search, each guess we make eliminates half of the possible cells we'd have to search.

Each time we double the data, the binary search algorithm adds just one more step..

For an array of size `3` it takes `2` steps, after we double the size it takes `3` steps and so on.

For `linear search`, ***each time we double the size of the array, we double the number of steps of our search***.

With an array of `10000` elements, linear search can take up to `10000` steps, while binary search takes up to a maximum of just `13` steps. For an array of size `one million`, linear search would take up to `one million` steps, while binary search would take up to just `20` steps.