Search Problem

Searching in List

```
L = [1,2,3,4,5,6,7,8]

# NAIVE SOLUTION:

# We will scan the list and if element is preset return true

def naivesearch(elem, L):
  for c in L:
    if c == elem:
      return True
  return false
```

Worst Case Scenario: `O(n)`

Searching a Sorted List: BINARY SEARCH

What if **L** is sorted in ascending order?

```
def binarysearch(v,l):
  if l == []:
    return False

  m = len(l)/2

# if v = m then its a midpoint
  if v == l[m]:
    return True

# search first half as v < m
  if v<l[m]:
    return binarysearch(v, l[:m])

# search second half if v > m
```

```
  else:
    return binarysearch(v, l[m+1:])
```

How long does this take?

`log(n)`: number of times divide n by 2 to reach 1.

Another approach to this:

`T(n)` : time to search the list of length n:

- If n == 0: `T(0) = 1`
- If n > 0: `T(n//2) + 1`

If we Unwind this:

- `T(n)` = `T(n//2) + 1`
- `T(n)` = `(T(n//4)+1) + 1`
- `T(n)` = `(T(n//4)) + 1 + 1`
- `T(n)` = `(T(n//2^k)) + k`
- `T(n)` = `T(1) + k` for `k = log(n)`
- `T(n)` = `T(0) + 1 + log(n)` = `2 + log(n)`