Big-O-2

January 29, 2021

0.0.1 Logarithmic Time

[2]:
```
a = [1, 2, 3, 4, 5, 6 , 7, 8, 9, 10]
```

[3]:
```
def binarySearch(alist, item):
    first = 0
    last = len(alist)-1
    found = False

    while first <= last and not found:
        midpoint = (first + last)//2
        if alist[midpoint] == item:
            found = True
        else:
            if item < alist[midpoint]:
                last = midpoint-1
            else:
                first = midpoint+1

    return found
```

In English this is:

```
Go to the middle of the list
Check to see if that element is the answer
If it's not, check to see if that element is more than the item we want to find
If it is, ignore the right-hand side (all the numbers higher than the midpoint) of the list and
Start over again, by finding the midpoint in the new list.
```

0.0.2 Polynomial Time

O(n^2) algorithm.

[4]:
```
def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):
```

```
        # Last i elements are already in place
        for j in range(0, n-i-1):

            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]

# Driver code to test above
arr = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(arr)
```

[5]:
```
arr
```

[5]:
```
[11, 12, 22, 25, 34, 64, 90]
```

Drop the constants

Drop the non-dominant terms

[ ]:
```

```