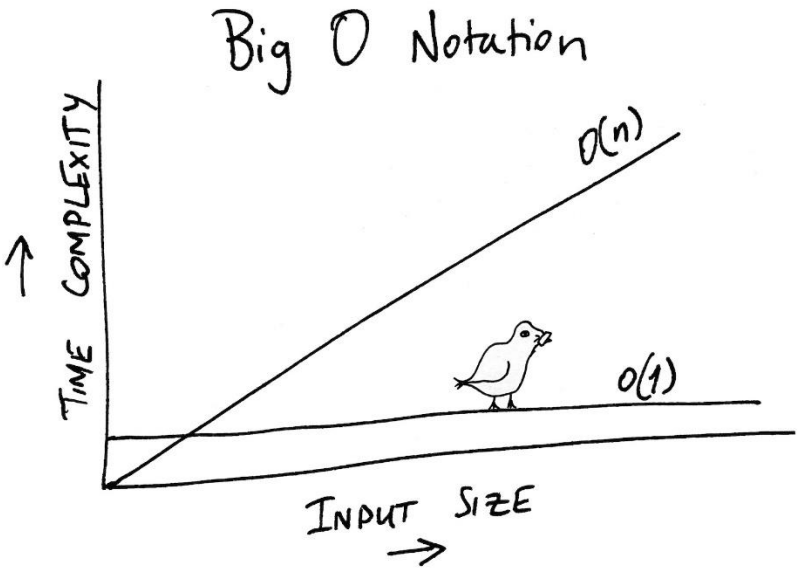
Time vs space complexity

Time complexity: calculates the time required

Space complexity: calculates the space required

Asymptotic notations:

1. Big-Oh (O) notation (denotes "fewer than or the same as" <expression> iterations)
2. Big Omega (Ω) notation (denotes "more than or the same as" <expression> iterations)
3. Big Theta (Θ) notation (denotes "the same as" <expression> iterations)



Big O notation is a mathematical way of describing how well algorithms scale and perform given a certain input.

Space complexity = Auxiliary Space + Space used for input values; O(1) – constant time

https://lucasfcosta.com/2017/06/12/Big-O-Explained-And-Why-You-Will-Never-Beat-a-Pigeon.html

Calculate the complexity of the following code snippets:

```
int sum(int num1, int num2) {
   int s;
   s = num1 + num2;
   return s;
}
```

Response: time complexity=

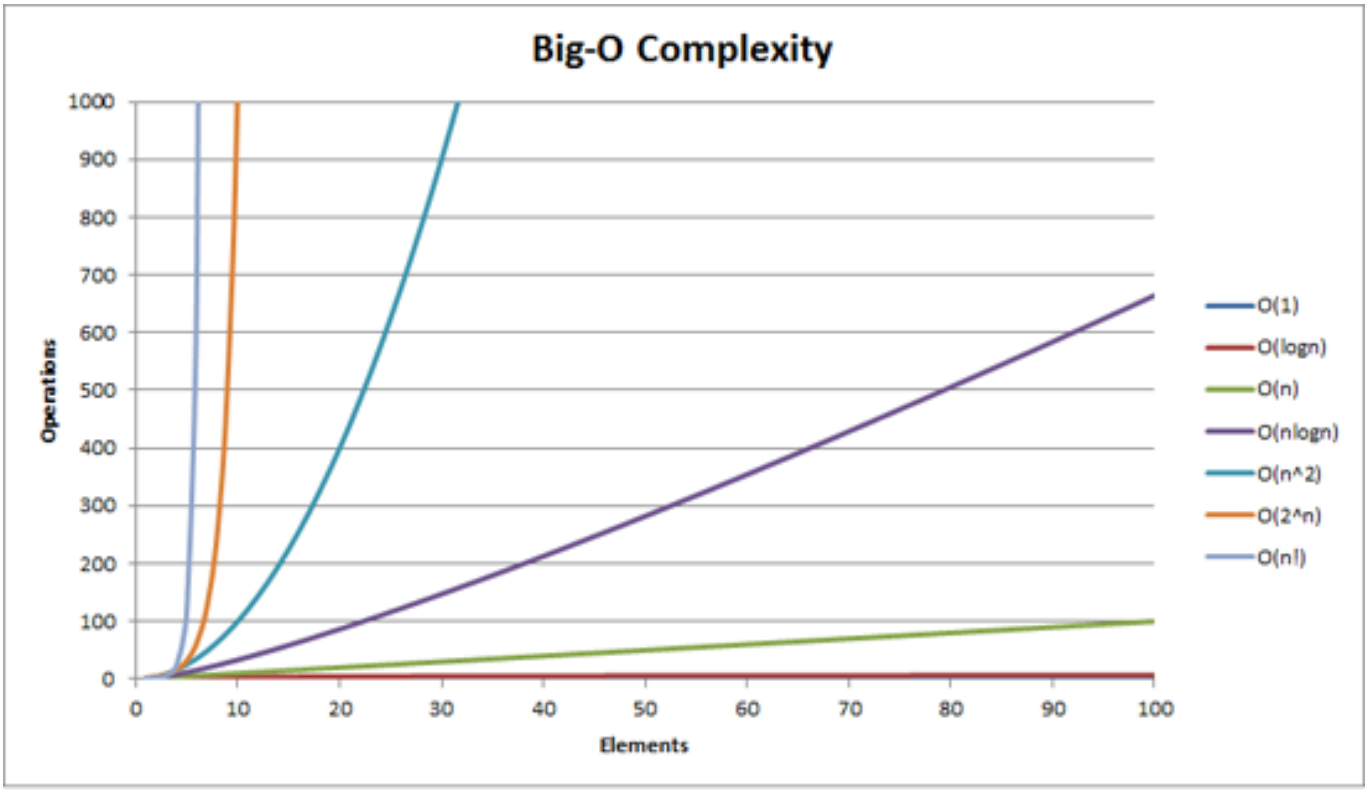
```
int sumofarray(int a[],int n)
 {
    int i,sum=0;

    for(i=0; i<n; i++)
    {
         sum+=a[i];
    }
   return sum;
 }
```

Response: time complexity=

```
int countduplicates(int arr[], int n)
{
   int i, j, count=0;
    for(i=0; i<n; i++)
    {for(j = i + 1; j < n; j++)
      {
        if(arr[i] == arr[j])
        {
          count++;
        }}}
    return count;
}
```

Response: time complexity=



O(logn) – binary trees; O(nlogn) – merge sort;

Different steps get added

```
Something(){
        doStep1()  // O(a)
        doStep2()  // O(b)
}
```

O(a+b)

Drop constants

```
O(2n) ⇔ O(n)
```

Different inputs = different variables

```
foreach x in arrayA
        foreach y in arrayB
                …
```

O(a*b); where a = len of arrayA and b = len of arrayB

About Git and GitHub: https://devmountain.com/blog/git-vs-github-whats-the-difference/

GitHub link: https://github.com/