Algorithms - Dominance relations

Alejandro Camus

Assignment: https://classroom.google.com/u/0/c/Mzc0NTg1MDE2NTFa/a/Mzc3NzU3NjQwMDBa/details

Big O notation groups function into categories of equivalent functions concerning their Big O, for example, f(x) = 5x^2+x = O(x^2)

1. Which are these functions? Name at least 8 eight functions ordered by dominance.
Hint: Start with f(n) = 1 << ... << f(n) = n!, which are 6 functions in between?
3. For each of these functions investigate an example of an algorithm having such big O and explain your reasoning.

O(1)
-Odd or even number
-Look up in a dictionary or hash map

O(log n) - Logarithmic
-Finding element on sorted array with binary search

O(n) - Linear
-Find max or min element in unsorted array

O(n log n) - Linearithmic
-Sorting elements in array with merge sort

O(n^2) - Quadratic
-Duplicate elements in array comparing each element of itself to each element of itself.

-Sorting array with bubble sort
Because: (n-1) + (n-2) + (n-3) + ... + 3 + 2 + 1
Sum = n(n-1)/2
So: O(n^2)

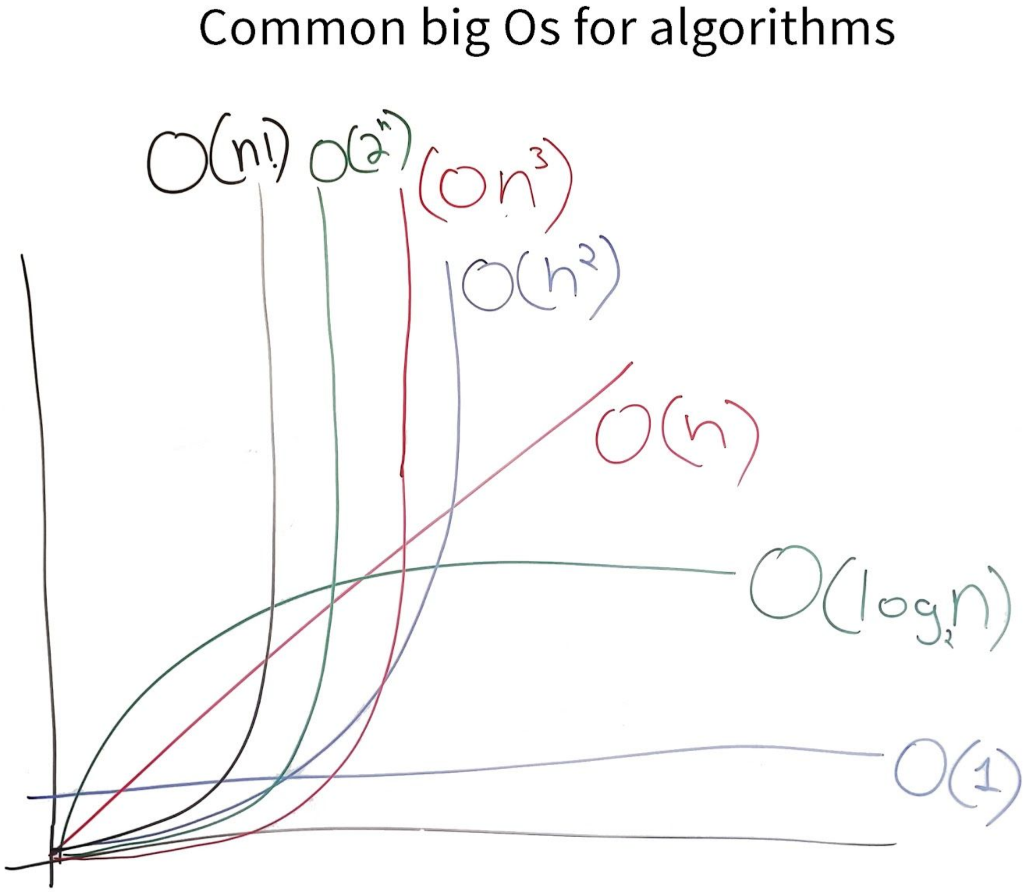
O(n^3) - Cubic
-3 variables equation solver (triple nested loops)

O(2^n)
-Fibonacci:

```
def Fibonacci(n):
    if n<0:
        print("Incorrect input")
    # First Fibonacci number is 0
    elif n==0:
        return 0
    # Second Fibonacci number is 1
    elif n==1:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)
print(Fibonacci(9))
```

O(n!)
-Permutations

2. Create a figure plotting all these functions together



Common big Os for algorithms