Programming Club Meeting 11 Slides

Algorithms

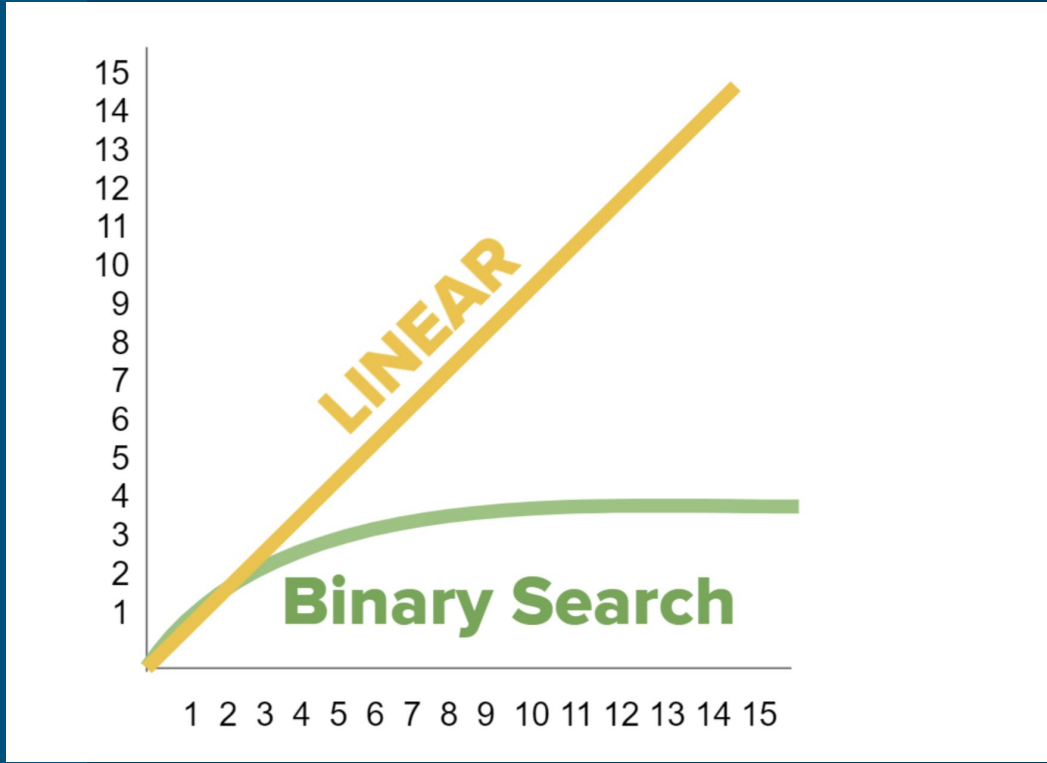
Need for Better Algorithms

- Many complex algorithms seek to solve certain problems faster
- The same way that a for loop is faster than a counting while loop, we can use better methods to solve problems
- Can organize the way that a solution's speed scales called Big O notation (won't be looking at this now)
- There are many places to write faster algorithms, we'll only look at binary search today

Binary Search

- To find an item in a list, the easiest (and only universally applicable) method is to look through the whole list one-by-one until you find it or have checked the whole list
  - Called linear search
  - Fairly slow with large lists
- If the list is sorted, we can use binary search to increase the speed
- Instead of going one-by-one, the computer will try to divide the list and minimize the number of items looked at
- There are three faster ways to find an item, but they have different requirements and are fairly difficult to implement

Linear vs. Binary Search



Implementing Binary Search

- Find the midpoint of the list and determine if that value is the target, too high, or too low
  - In searching, the “target” is the value that you are searching for
- If the item is too high, only look at the items before that value
- If it’s too low, only look at the items after it
- If it is the value, you’re done
- Repeat
- If the computer runs out of items to look at, the item isn’t in the list
- Ex: Number Guessing Game


left = 0
right = 9
Arr
0
1
2
3
4
5
6
7
8
9
item = 3
mid = 4
A[mid] = 4

Practice

Practice Problem 1:

Implement Binary Search

- Goal: Write a Python program that implements a binary search function.

Practice Problem 2: Prime Factorization

- Src: https://www.101computing.net/prime-factor-tree-algorithm/
- Goal: Write a Python function that will output the prime factorization of an inputted number. Also write a program that will test this functionality.
- Relevant Information:
  - You just need to find the prime factors and store them in a list
  - Ex:
    - 140 = [2, 2, 5, 7]
    - 150 = [2, 3, 5, 5]
  - The easiest method is dividing the current amount by the smallest prime number it's divisible by, adding that prime to the list, and repeating until the current number is prime
  - There is an outline file on the GitHub (in the folder 'primeFactorization') which includes a function to check if a number is prime

Practice Problem 3:

Largest Product

- Goal: Write a Python program that will determine and output the sub-list in a 2D array with the largest product. If multiple sub-lists have the same product, the first should be returned.
- Ex:
  - In - [[2,5], [-3,6], [4,4], [2,3], [83,1], [7,1]]
  - Out - [83,1]

Next Meeting: Files