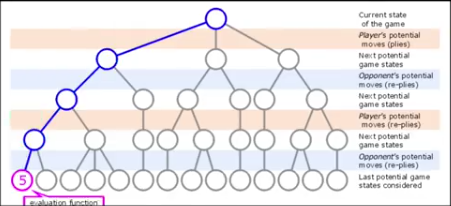
*Thom cormen – Introduction to algorithm*

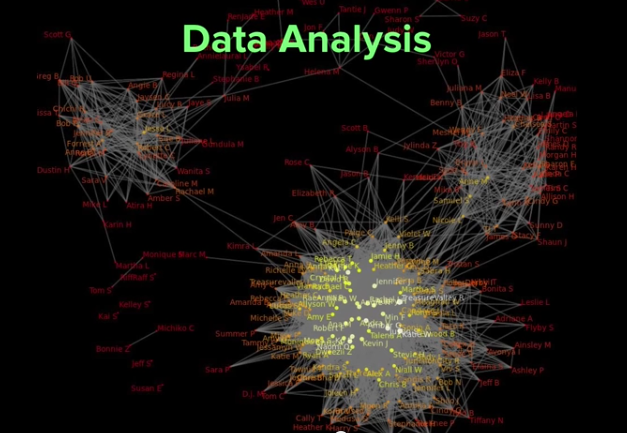
*When finished course: 1 algorithm a day https://en.wikipedia.org/wiki/List\_of\_algorithms*

Algorithms are used in every corner of the data industry. Audio, video compression algorithm, rendering algorithm to color Pixar animated characters.

EX1: Minimax algorithm used for programming chess playing computers:

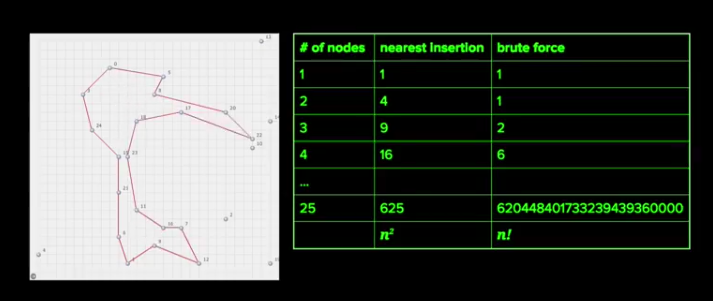


EX2: websites use algorithm to analyze and structure vast databases:



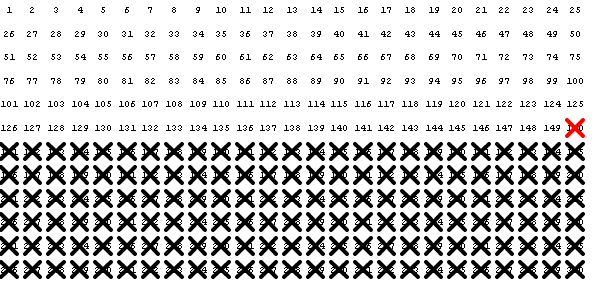
**Asymptotic Analysis:**

Used to determine whether an algorithm is efficient or not.



Binary Search

Guess the number the computer thinks between 1 and 300. We start at the middle, tells us whether bigger or smaller, then we take the mean of that new interval and so on:

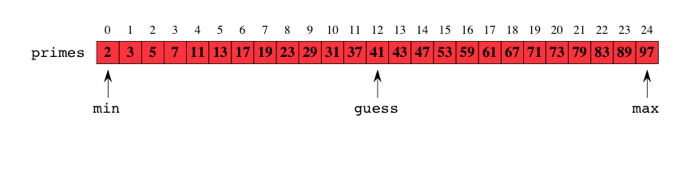


**Implementing binary search of an array**

var primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97];

Suppose we want to know whether the number 67 is prime. If 67 is in the array, then it's prime.

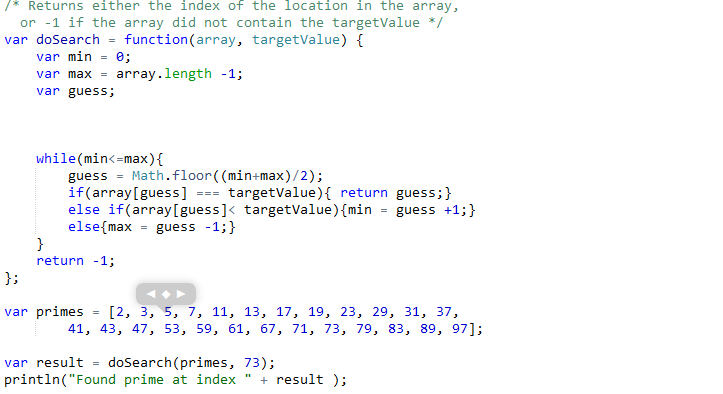
We might also want to know how many primes are smaller than 67. If we find the position of the number 67 in the array, we can use that to figure out how many smaller primes exist



Since we know 67>41, we take only the right half of the array.

* find mid array
* 67>mid?
* min = mid + 1 , max = max or min = min, max = mid - 1
* find mid new array
* 67>mid?

JAVASCRIPT:



IMPORTANT: don’t forget the equal. Why? Here’s Why:

Let’s say you are looking for primes[1]=3

min=0

max = array.length -1 = 24

guess = 12

=>min = 0, max = 11

etc…

Until we arrive to min = 0 , max = 1, guess = 0. primes[0] = 2 < 3 --> min = guess + 1 = 1, max = 1. The function returns -1, we are screwed.

*Now, with <=*

*min = 1, max = 1. We do one more loop, because min <= max is fulfilled. guess = (min+max) /2 = 1*

*and guess[1] = 3. WE FOUND 3!* Jesus that was tedious.

I don’t know why, but the problem I mentioned only appears for even indexes, so writing while min<max will actually work if we are looking for a prime number whose index in the array is an odd number.

* What’s good about binary search is that it has a complexity of lg(n) +1 (log 2 of n).

In other words, to find a value in an array of 2,097,152 elements, we need only 22 steps at most instead of.. well, 2,097,152 steps. So we can say that the utility of binary search grows exponentially, and for once, we will have used that word correctly. ALSO, if an element is not the array, we will need the max amount of steps to find that out (hence lg(n)+1). But that’s Ok. (Round up the log if necessary.)