

<image/nation>

Tech Training Series

Exclusive for



# Data Structure & Algorithm

2023/2024

How data are stored and how data are processed efficiently

*by*

Sunny NG

<image/nation>

# In this workshop (3 hours)

- Introduction to algorithm
- Introduction to data structure
- Algorithm Examples
- Data structure and algorithm visual simulation
- Common data operations
- Computation Complexity
- Time vs. Space (Speed vs. Storage)
- **Hands-on Practicing**
  - Variable, array, objects and JSON
  - Looping
  - Array iterating
  - Built-in methods for Array and object
  - Searching algorithm
  - Sorting algorithm
  - Map/Reduce

# Sunny Ng



- Founder / Master Trainer Image Nation
- Developer Web, Mobile, WeChat & IoT
- Content Creator Video producing / Live streaming
- AWS Solution Architect – Associate
- Alibaba Cloud Professional
- AWS Academy Educator
- Email: [sunny.ng@imagenation.com.hk](mailto:sunny.ng@imagenation.com.hk)
- 🐙 [github.com/ngsanluk](https://github.com/ngsanluk)

# Required Software Installation

1. Node JS
2. Visual Studio Code
3. Google Chrome Browser

# Node.js Download

- Node.js is a popular development tool and runtime for Web/JavaScript
- Download link
  - <https://nodejs.org/en/download/>




node

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

## Downloads

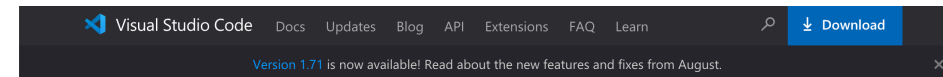
Latest Current Version: 18.8.0 (includes npm 8.18.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

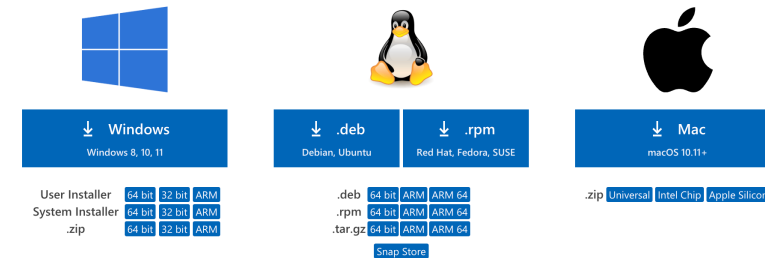
LTS Recommended For Most Users	Current Latest Features	
 Windows Installer node-v18.8.0-x86.msi	 macOS Installer node-v18.8.0.pkg	 Source Code node-v18.8.0.tar.gz
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v18.8.0.tar.gz	

# Visual Studio Code

- Visual Studio Code is one of the most popular modern code editors
- We will use VS Code for JavaScript coding/editing
- Download link
  - <https://code.visualstudio.com/download>

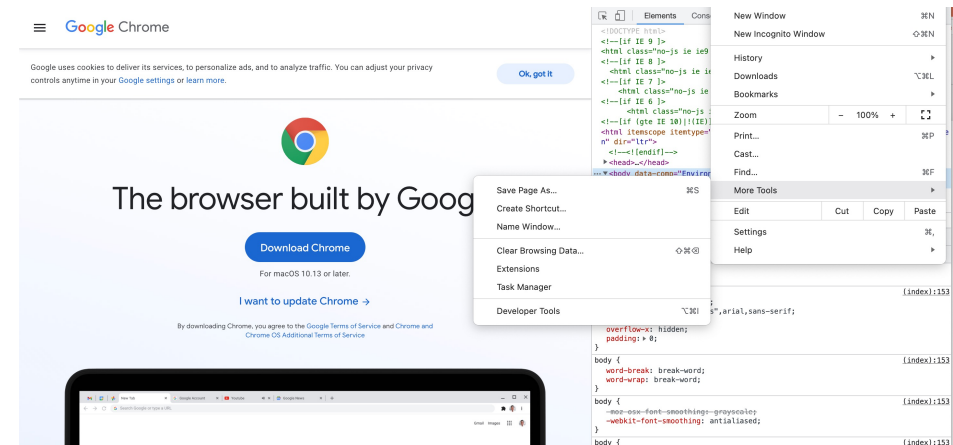


Download Visual Studio Code  
Free and built on open source. Integrated Git, debugging and extensions.



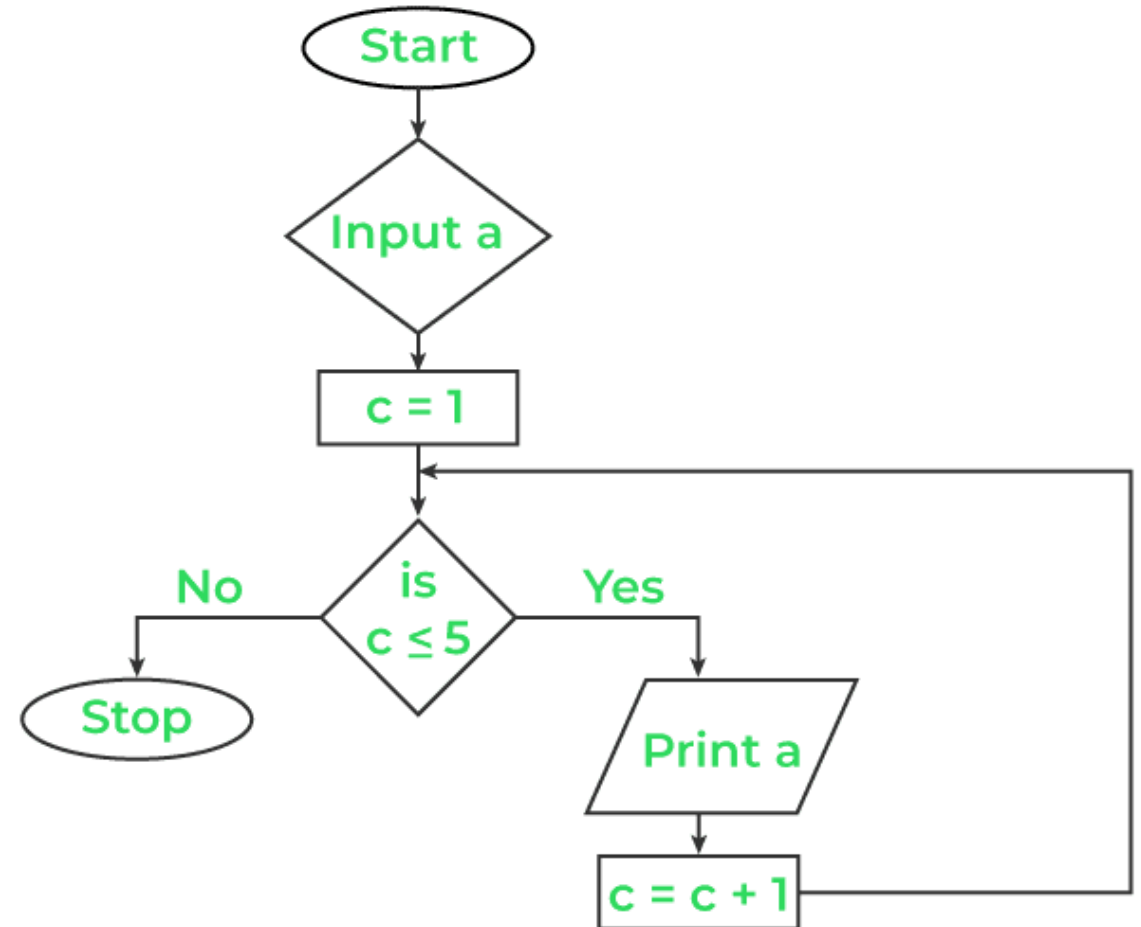
# Google Chrome Browser

- We will need to use the **Chrome Developer Tools**
- Google “**Chrome installer**” to download and install
- or
- Visit download link
- [https://www.google.com/intl/en\\_hk/chrome/](https://www.google.com/intl/en_hk/chrome/)



# What is Algorithm?

Algorithm is the step-by-step procedure that defines a set of instructions to be executed in a certain order to get the desired output.





# Searching has been a primary computation problem

- Algorithms are there to solve computation problems while computation problems usually roots to life problems
- One of classical computation problems is **searching**



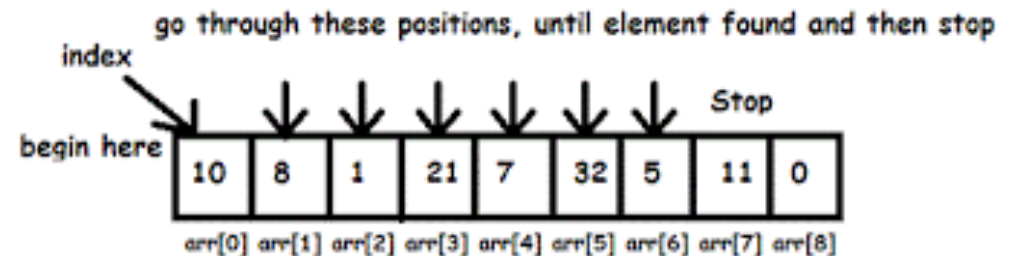
# Search is also real-life problem: large collection of books



# Searching is easy? Yes and No.

- Search is easy.
- But search **efficiently** is NOT easy at all.
- Consider the data volume that a nowadays business application is dealing with
  - No. of social media posts
  - No. of YouTube video uploads
  - No. of transactions on e-commerce platform

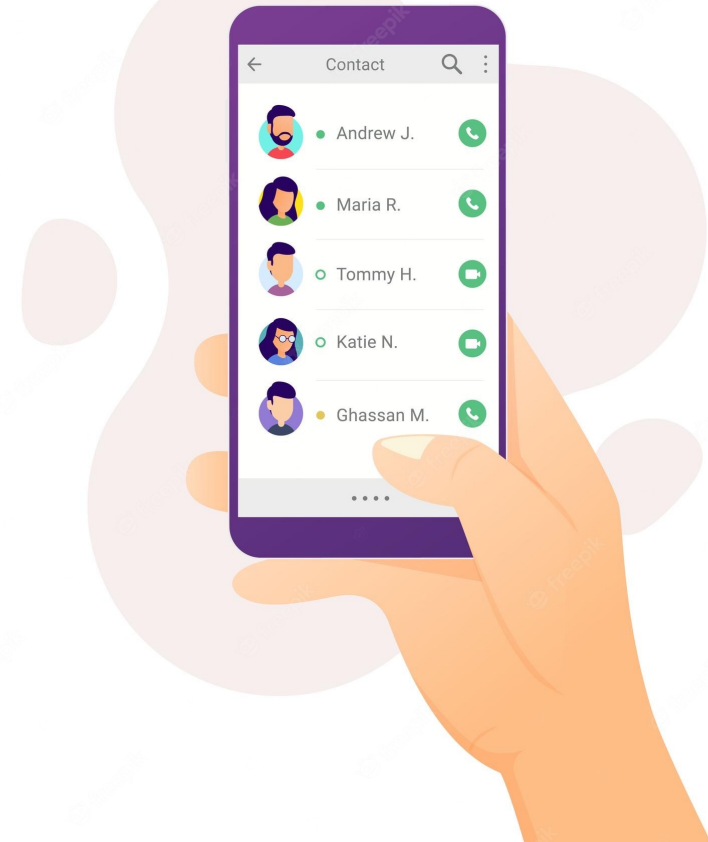
Linear search does give the answer, but way too slow



Element to search : 5

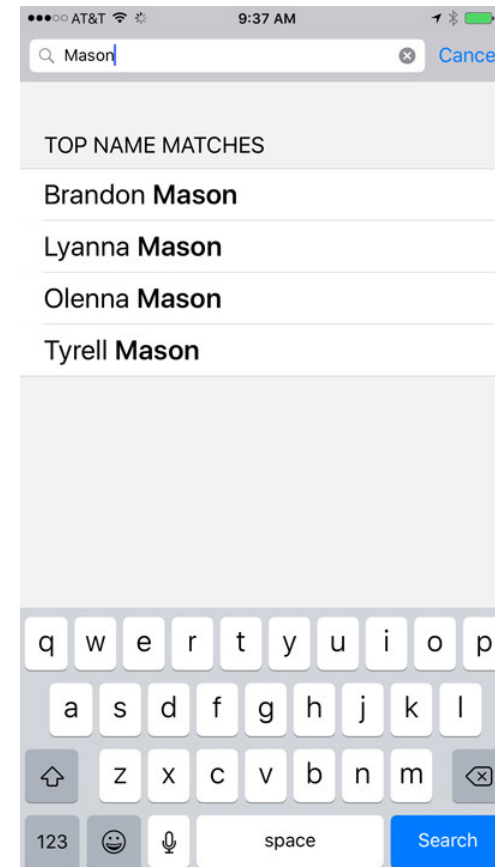
# Phonebook browsing

- If there are only dozens of names in your phone book, **browsing** the name list one-by-one might be okay.



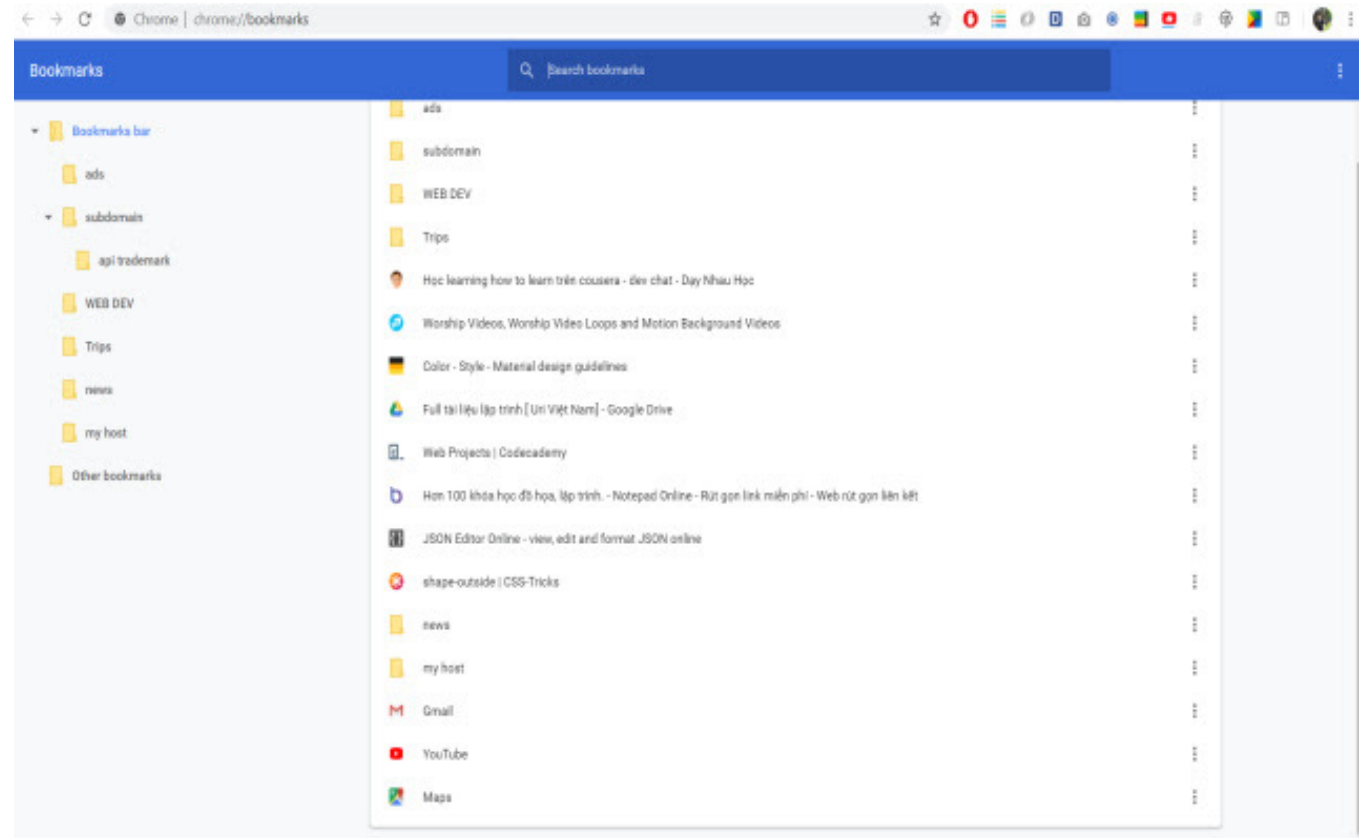
# Phonebook searching

- When phone book entries grow, **searching** (by keyword) is the only answer



# In the early days of the Web

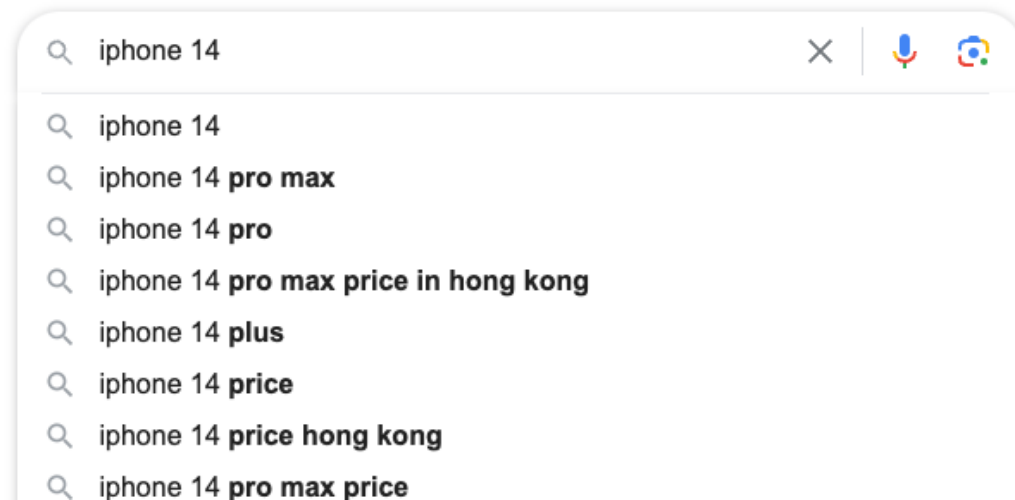
- Each web user would build their own bookmarks to include websites of interest



# Searching the web

- We search on Google and YouTube many many times each day
- Take a guess if I search “iPhone 14”, **how many** webpages will match?

# Let's give it a try





# Google Changes Life

- Do you realize how ridiculous **huge** is Google's database?
- Do you realize how **fast** Google are performing searching?

# What is Data Structure?

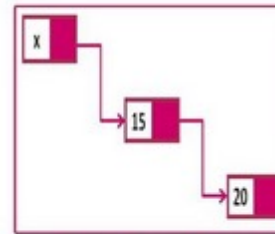
- Data Structures are the **special programmatic way** of storing data so that data can be used efficiently.
- Data Structure and Algorithm form an optimized combination to solve computation problem **efficiently**.
- Google gives their search result in such rapid speed because of their highly optimized data structure and algorithm.

# Common Data Structures

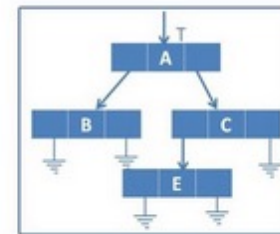
- Array
- Linked List
- Stack
- Queue
- Tree
- Hash-table
- Graph



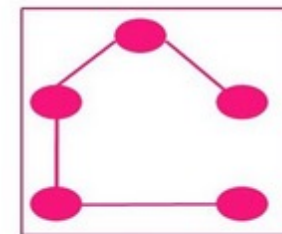
Sorting



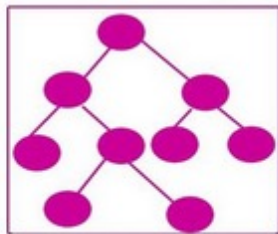
Link list



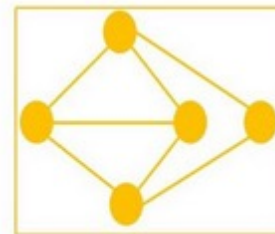
list



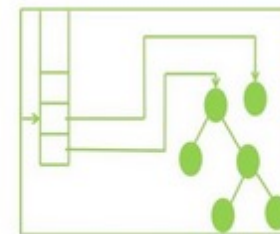
spanning tree



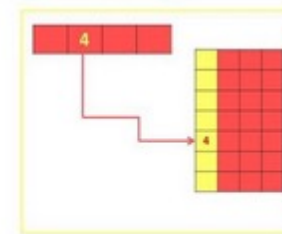
Tree



Graph



Stack



Hashing

Computer algorithms are written to be understood and executed by CPU.

Therefore, they could be difficult to understand by people without programming training

# Swapping two number is straight forward to human brain

Say -

We have two numbers

- $x = 100$

- $y = 200$

A human brain would

Just get to the answer

- $x = y$

- $y = x$

Let's try to code  
**swapping**  
in JavaScript

# Unfortunately, the codes below don't work as we expect

```
x = 100;
y = 200;
console.log("Before Swapping");
console.log(`x = ${x}`);
console.log(`y = ${y}`);

// swapping x and y
x = y;
y = x;

console.log("\nAfter Swapping");
console.log(`x = ${x}`);
console.log(`y = ${y}`);
```

# To Future Data Scientist

- **Bad news** is computer algorithms are difficult to understand, especially low-level programming like C programming
- **Good news** is modern high-level programming language like JavaScript and Python are succinct and much easier to read and code. And many useful algorithms are already built in.
  - Still, it's better that you understand what basic computation operations that a computer algorithms are performing



# After all, an algorithm is a **long set** of very simple operations

Operation/Instruction	It's processing	Code Examples
Conditional flow control	Do processing conditionally	<b>if</b> () {} <b>else</b> {}
Comparison Operators	Compare two value. It evaluates to TRUE or FALSE	<b>if</b> ( <b>a==b</b> ) {} <b>if</b> ( <b>a!=b</b> ) {}
Logical Operators	Combine multiple logical operations	<b>if</b> (x>=y <b>&amp;&amp;</b> x>=z) {}
Assignment Operator	Temporary save the value to memory for further processing	x = 100 i = i + 1
Looping	Respectively executing the processing	<b>for</b> (i=0;i<n;i++) {}
Arithmetic Operators	Mathematic operations	i = j * 2 - 3

# Simple Search Algorithm

# Linear Search

# Visualizing Linear Search

Searching for **33**

- Match number by number. Easy to understand.
- Easy to implemented by computer programming.
- Not very effective. Imagine the dataset size nowadays.

Linear Search



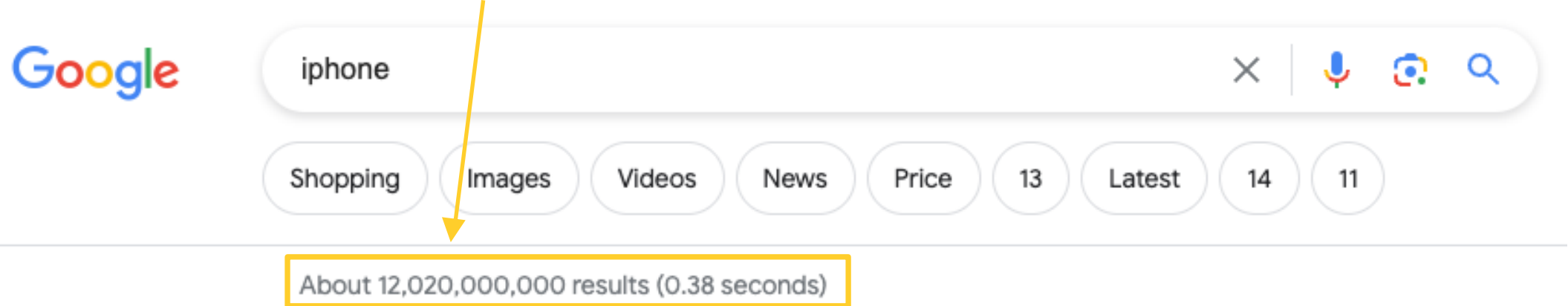
=  
33

# Linear Search in JavaScript Implementation

```
function search(arr, x) {  
    for (i = 0; i < arr.length; i++) {  
        if (arr[i] == x) return i; // found  
    }  
    return -1; // not found  
}
```

# Linear Search is useful only when the data size is small

Remember the **data size** of Google search is huge?



# We need **quicker** search algorithm ...

- Turns out if we use certain programming data structures (e.g. **array**) together with special algorithm (e.g. **binary sort**), the searching can be a lot faster.
- But there is a prerequisite
  - The array of data have to be pre-processed - **sorted**
  - This is the computation cost to faster sorting

# We are **halving** the data size in each round of binary search

Binary search

steps: 0



The required pre-processing

# Sorting the Array



# Algorithm Visualization

Help you understand algorithm better

# <https://visualgo.net/en>

## Help you understand algorithm better

**Do You Know?** [Next Random Tip](#)

Visualgo has two main components: The 24 visualization pages and their associated Online Quiz component (more questions are currently being added into the question bank). We do not script any of the questions in Online Quiz :O and all answers will be graded almost instantly :). You can this online quiz system by clicking the 'Training' button on the visualization module.

**VISUALGO**.NET/EN  
*visualising data structures and algorithms through animation*

Search...

**NUS** Computing  
National University of Singapore

Featured story: Visualizing Algorithms with a Click

**Optiver**

Breaking news [Fri, 09 Jun 23]: Visualgo project is funded by Optiver starting today. We now open Visualgo account registration to every Computer Science students/teachers worldwide.

**Sorting** [Training](#)  
array algorithm bubble

**Bitmask** [Training](#)  
bit manipulation set cs3233

**Linked List** [Training](#)  
stack queue doubly deque

**Binary Heap** [Training](#)  
priority queue recursive

**Hash Table** [Training](#)  
open addressing linear

# Computation Complexity

Measuring how efficient an algorithm is

# Ultimate Goals of Computation Complexity

## ■ Time Complexity

- Running time or the execution time of operations of data structure must be as small as possible.

## ■ Space Complexity

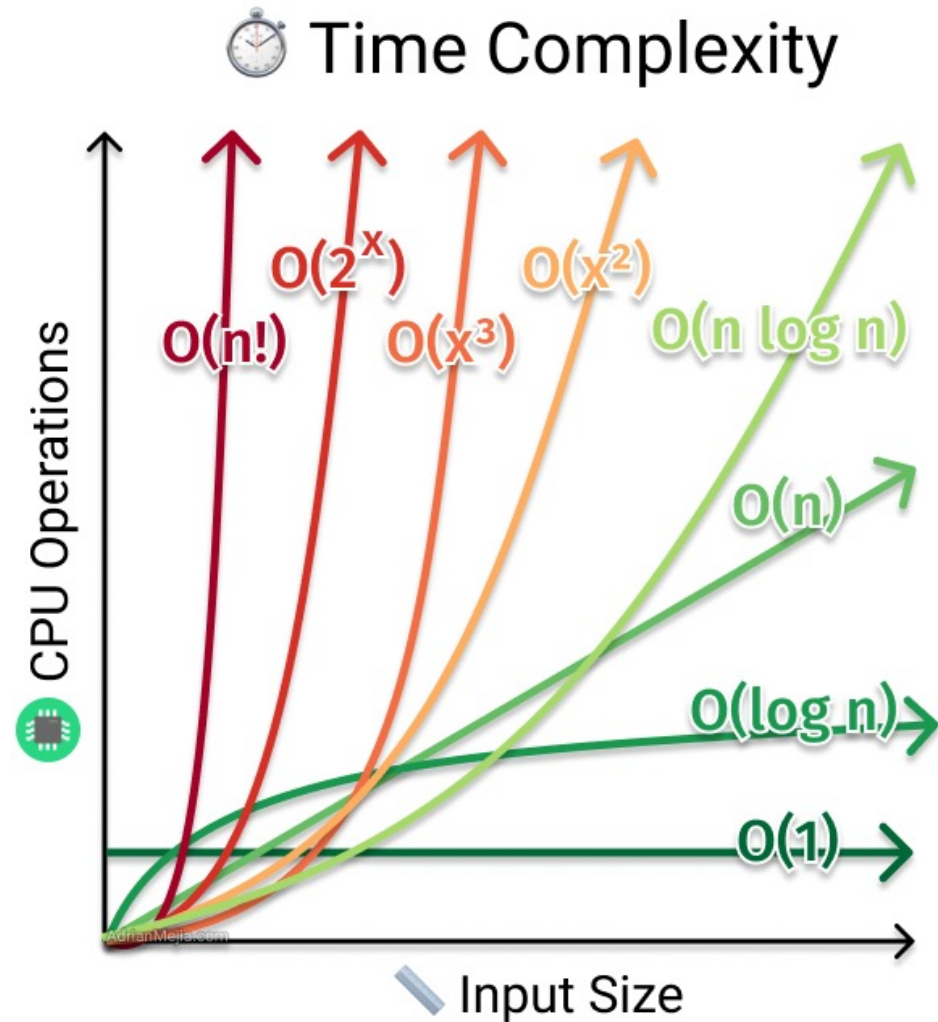
- Memory usage of a data structure operation should be as little as possible.

# Big-O

Common Time Complexity Measurement

Big-O judges an algorithm by its worst-case performance

E.g. In linear search, it must visit each element at least once and therefore the time complexity is  $O(n)$



# Time Complexity Examples

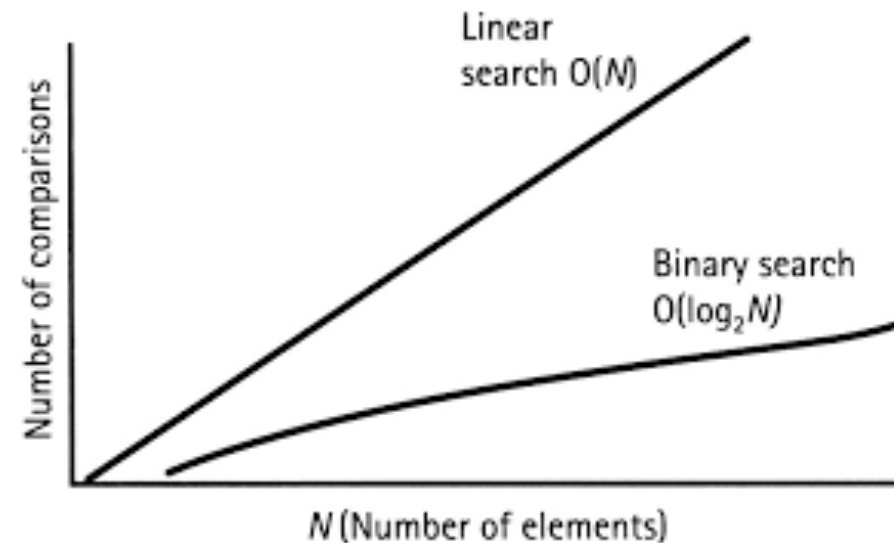
## Linear Search

- $O(n)$
- $n$  is the size of data to search

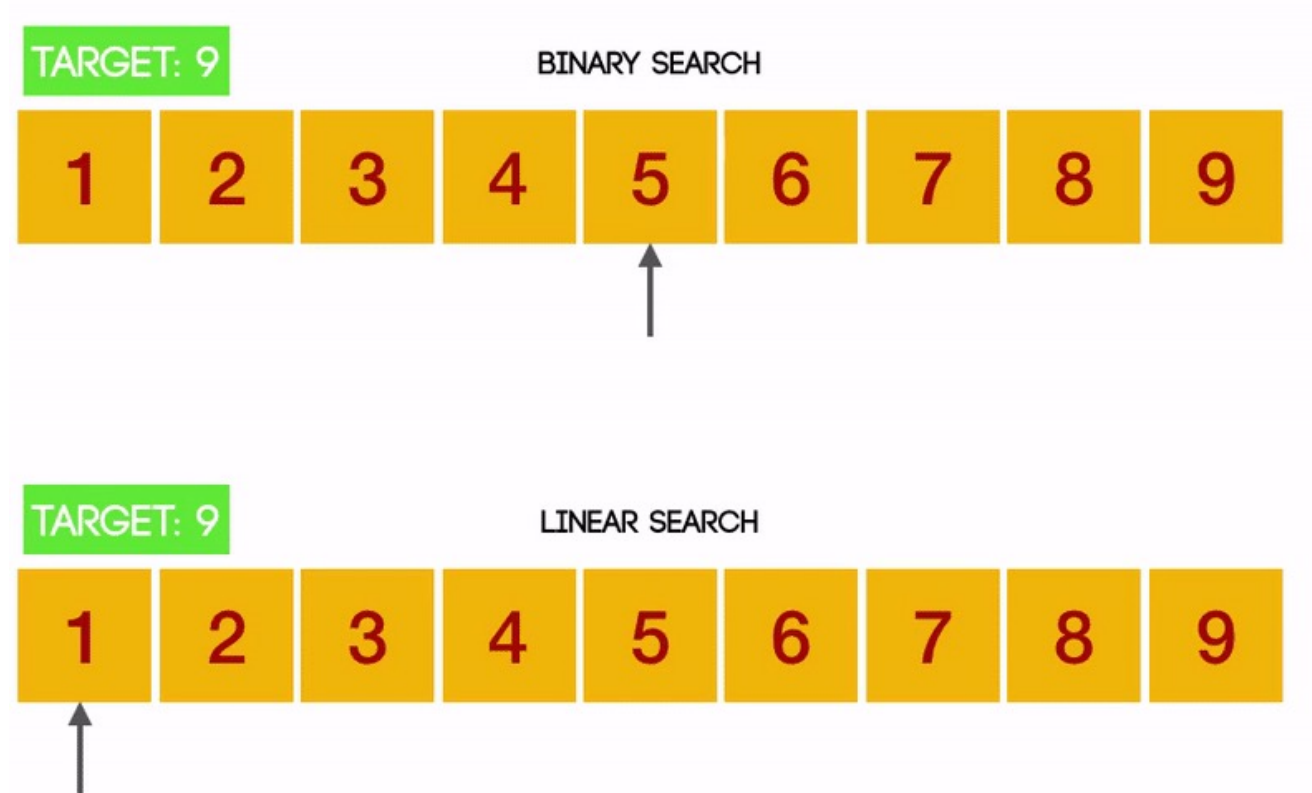
## Binary Search

- $O(\log n)$
- $n$  is the size of data to search

Binary Search is a better algorithm for searching



# Linear Search vs. Binary Search



# When the data size is huge ...

If you are searching a number towards **ONE MILLION** numbers

- for linear search, it has to perform **1 million** comparison before you get the answer
- for binary search, it takes at most **20 rounds** to find the target value.
  - $\log(1,000,000) \approx 20$
  - In calculator, you get the result by doing  $\ln(1,000,000) / \ln(2)$



# Common data processing algorithm

- **Search** - Algorithm to search an item in a data structure.
- **Sort** - Algorithm to sort items in a certain order.
- **Insert** - Algorithm to insert an item in a data structure.
- **Update** - Algorithm to update an existing item in a data structure.
- **Delete** - Algorithm to delete an existing item from a data structure.

# There is **NO** such thing as the **Best Algorithm**

- Some data-structure/algorithms are outstanding in time complexity, but NOT GOOD in space complexity
- Some data-structure/algorithms are the other way around
- Some data-structure/algorithms are good in search, but perform poorly in insert, delete and update while some data-structure/algorithms the other way around
- That's why you need to pick one that suits your use-case

# JavaScript Object

# JavaScript Object

- JavaScript is a compound data structure
- It uses a pair of braces `{ }` to denote object
- In an object, there are multiple pairs of Key-Value.
- Keys and values are separated by a colon `:` (NOT equals sign `=`)
  - The key is the attribute name (like a column name to a table)
  - The value could be type of number, string or other
- Each pair of key-value are separated by comas `,`

```
let person = {  
  name: 'John',  
  age: 20  
};
```

Keys ----- Values

# JavaScript Object can be nested

- JavaScript object is multiple level
- Meaning object can be inside object (and this is quite common)

```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
},
```

# Multiple objects with same structure form an array/list

- It use a pair of brackets `[]` to denote an array
- Inside the array, there are multiple objects
- Each object is wrapped inside a pair of braces `{ }`
- Objects are separated by a comas `,`

```
1  const list = [  
2    {  
3      name: 'Michael Scott',  
4      company: 'Dunder Mufflin',  
5      designation: 'Regional Manager',  
6      show: 'The Office'  
7    },  
8    {  
9      name: 'Barney Stinson',  
10     company: 'Golaith National Bank',  
11     designation: 'Please',  
12     show: 'How I met your mother'  
13   },  
14   {  
15     name: 'Jake Peralta',  
16     company: 'NYPD',  
17     designation: 'Detective',  
18     show: 'Brooklyn 99'  
19   },  
20 ]
```

Hands-on High-Level  
Language Practicing

2 hours

# Source Codes Download

`bit.ly/dsa-with-js`