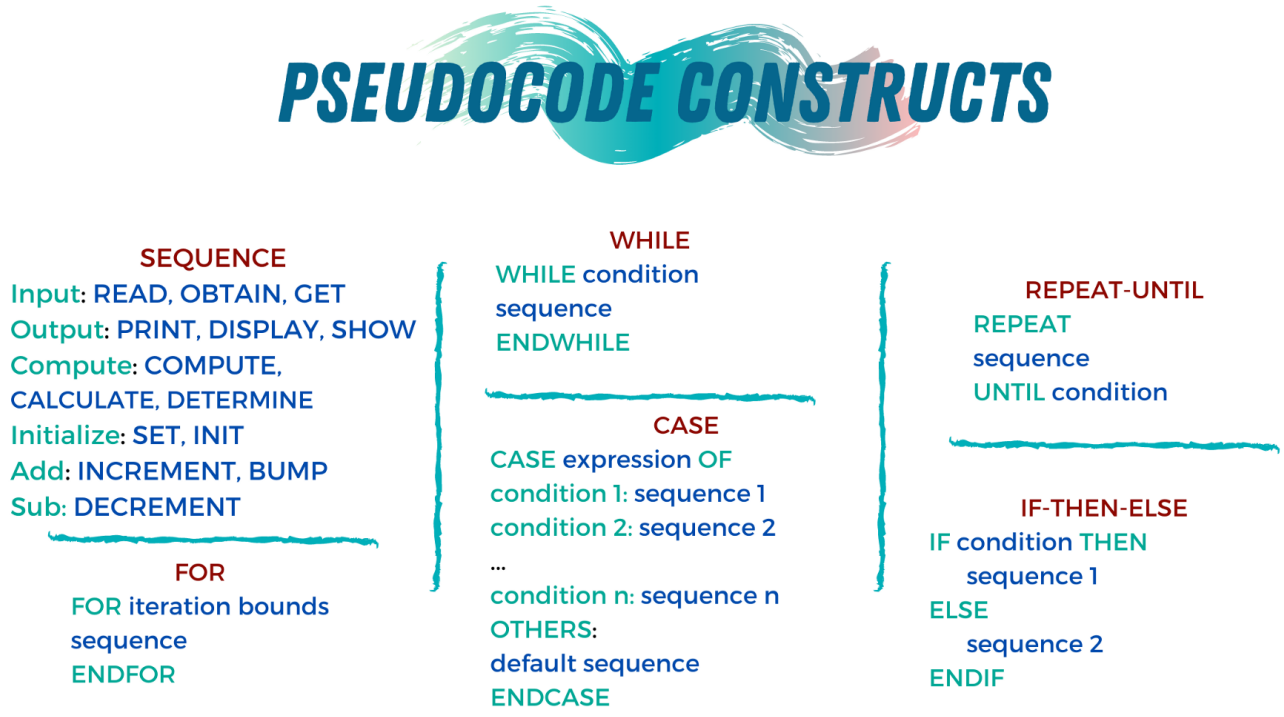
Tổng quan về Giải thuật cơ bản 🎉

AGENDA

1. Tổng quan về giải thuật cơ bản
2. Thuật toán tìm kiếm
3. Thuật toán sắp xếp

Pseudo code - Mã giả

- **What**: high level algorithm instruction
- **Why**: describe an algorithm in plain language, no need to specific in any programming language.
- **When**: research and propose solution
- **How**: learn some keywords to describe your algo



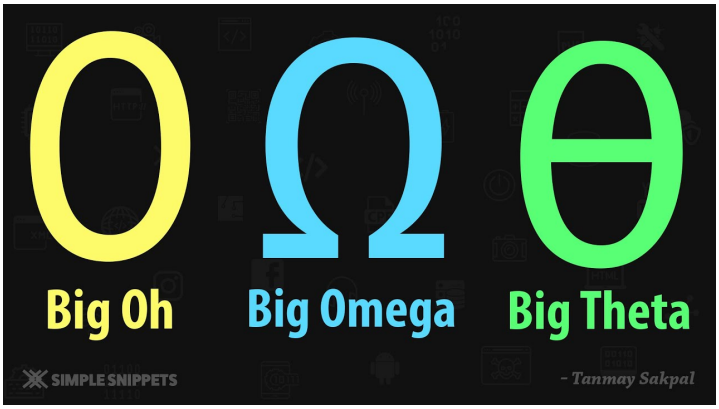
> Source: https://towardsdatascience.com/pseudocode-101-an-introduction-to-writing-good-pseudocode-1331cb855be7

```
FUNCTION findMax
INPUT:
  numberList - an array of numbers
OUTPUT: the maximum number
---
IF (numberList is not an array or array is empty)
  RETURN undefined;

SET max = first element
FOR each number in numberList
  IF current number > max
  THEN SET max = current number

RETURN max
```

Đánh giá mức độ hiệu quả của giải thuật

- **Time Complexity**: The time complexity of an algorithm is the amount of time taken by the algorithm to complete its process as a function of its input length, n source
- **Space Complexity**: The space complexity of an algorithm is the amount of space (or memory) taken by the algorithm to run as a function of its input length, n source

	Time Complexity	Space Complexity
What	Calc time needed	Calc memory space needed
How	Count time for all statements	Count memory for space for all vars (even input + output)
Depends on	Input data size	Mostly depends on auxiliary variables size
Important	More important for solution optimization	Less important with modern hardwares



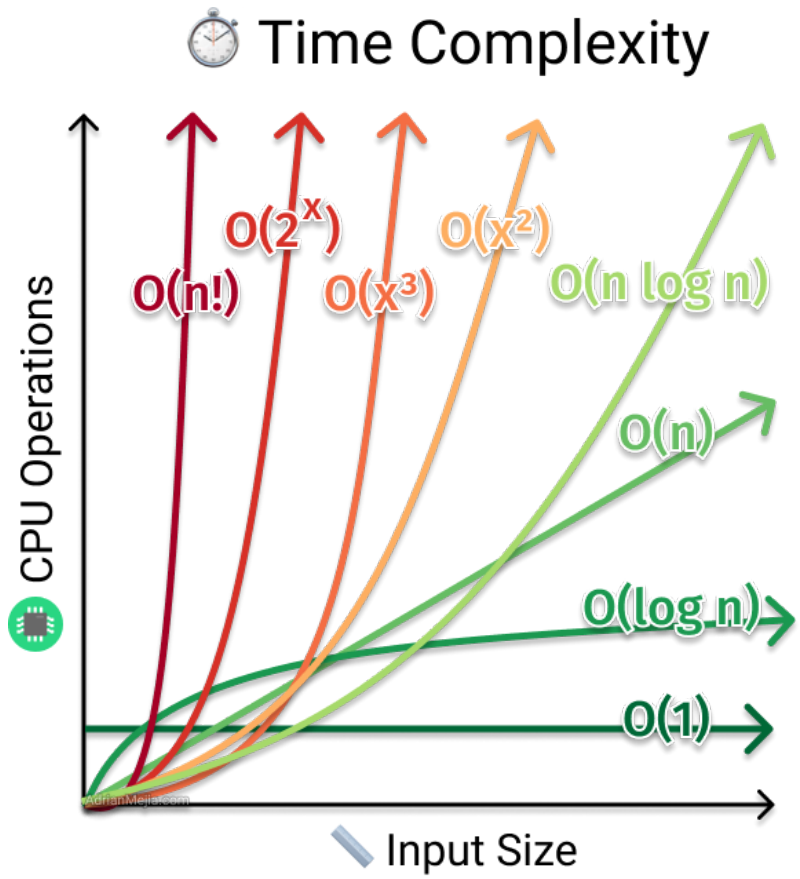
> Tham khảo: https://youtu.be/1tfdr1Iv6JA

```
function findIndex(numberList, target) {
  if (!Array.isArray(numberList) || numberList.length === 0) return -1;

  for (let i = 0; i < numberList.length; i++) {
    const number = numberList[i];
    if (number === target) return i;
  }

  return -1;
}
```

- Big O: **O(n)** (worst case, you need to loop to the end)
- Big Omega: **O(1)** (best case, you found it at the first position)

Một số Big O phổ biến

#	Name	Desc
1	O(1)	fixed number of times no matter how big the input is
2	O(log(n))	reduce half on every step
3	O(n)	one loop
4	O(n^2)	two nested loop
5	O(n^3)	three nested loop



> Tham khảo: https://adrianmejia.com/how-to-find-time-complexity-of-an-algorithm-code-big-o-notation/

Tham khảo

- https://www.educative.io/edpresso/time-complexity-vs-space-complexity
- https://adrianmejia.com/how-to-find-time-complexity-of-an-algorithm-code-big-o-notation/

Khoá học Javascript cho người mới bắt đầu 2021 🎉

- Tác giả: **Hậu Nguyễn** - Founder Easy Frontend
- Khoá học chỉ được published trên Udemy, không thông qua trung gian.
- Khoá học không bán dạng videos upload trên Google Drive hay bất cứ hình thức nào tương tự.
- Khoá học có nhóm discord để hỗ trợ trong quá trình học tập.

☎️ Liên hệ tác giả để được hỗ trợ:

- ✅ Facebook: https://www.facebook.com/nvhauesmn/
- ✅ Fanpage: https://www.facebook.com/learn.easyfrontend
- ✅ Youtube Channel: https://www.youtube.com/easyfrontend