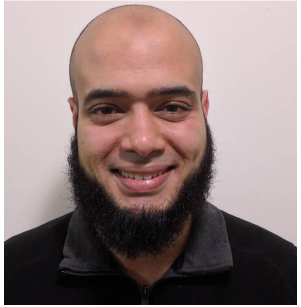
Data Structures

Asymptotic Complexity 1

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



How much time? memory?

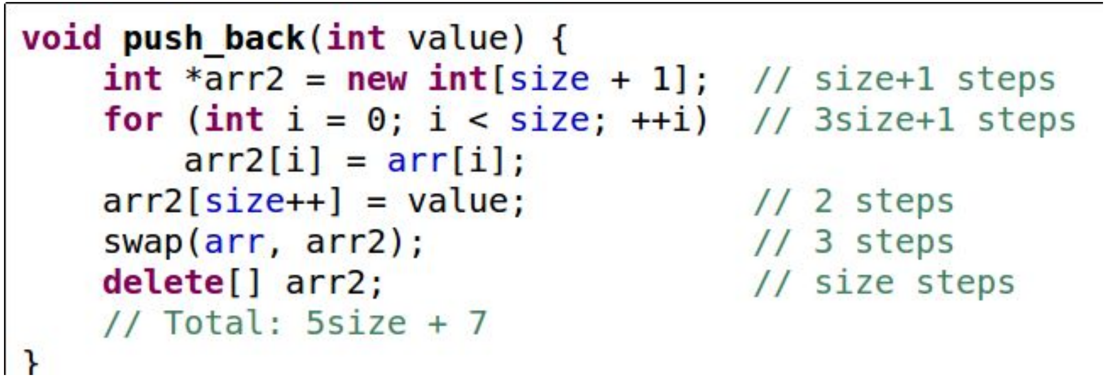
- Some services (e.g. from Google) are very fast, but many are slow
- Some of your computer programs consumes your memory
- Our code consumes time and memory!
- When we develop our code and run it, it will take some time
- But how to **estimate** them so that we know good our code?
- We might think in different ways to do so
    - E.g. run the function and compute total time, but this is hardware dependent?!
- **Asymptotic Complexity**: is a field that answers this question

My educational approach

- The formal introduction to this field involves many abstract concepts and mathematics
- Students find it inconvenient in the begin
- I prefer an informal treatment
    - Focusing only on a small portion that we actually use in the industry
    - Gaining **incremental** experience in computing the complexity
- Later, when you deal with the math/logic behind it ⇒ Much easier

Approximately, How many steps

- Below code is doing **push_back** in Vector (of currently size elements)
    - It creates an array. It copies the old data. It adds the new element. Swap pointers. Remove old
- If we tried to estimate the number of steps ⇒ ~ 5size + 7
- It takes linear number of steps!



```
void push_back(int value) {
    int *arr2 = new int[size + 1];  // size+1 steps
    for (int i = 0; i < size; ++i)  // 3size+1 steps
        arr2[i] = arr[i];
    arr2[size++] = value;           // 2 steps
    swap(arr, arr2);                // 3 steps
    delete[] arr2;                  // size steps
    // Total: 5size + 7
}
```

With large N

- Let’s say we have function f(n)
- We computed the exact number of steps: n + 19
- Think about **large** n = 10^9 (million)
    - Does it matter if it is 1000,000,000 steps or 1000,000,019 steps? Clearly no
- It is more **intuitive** to just think, it takes n steps
- What if it is 5n. Again 1 billion steps vs 5 billion steps is not that far
    - Both extremely slow
- Thinking about **large** N, we actually don’t care about these factors
    - Time wise all following are close: n,   10n+19,   13n+20,   n+1000
    - In all of them, the code takes **linear number** of steps

Big O notation

- If a code takes **9n+17 steps**, we say it is O(n) code
  - Order of N
- It means the code runs in **linear time** relative to n
  - Little more mathematically,

Big O notation: Guidelines

- Remember, we alway think with very **large** N
- Practically, the **largest** term in the **equation** is the one that dominates
  - All others are neglectable with large N
- Assume your code takes the following number of steps:
  - 5n^2 + 10n ⇒ O(n^2). So we selected the **largest term** (n^2) as it will dominate
  - n * (n+1) / 2 ⇒ O(n^2), Again, expand the expression to find the **biggest**
  - 2n^4 + 5n^3 + n + 9 ⇒ O(n^4): Again the **largest**
  - 17 ⇒ O(1): We say, this is constant time (**largest** is n^0).
    - It always confuses students.
    - If an algorithm is doing 10^6 fixed steps, it is again O(1). We don't have this in practice
    - Observe: what matters is n, as it affects total steps for a LARGE n

How skilled programmers find the order
VERY fast?

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."