

$$O(1)$$

$$O(\log n)$$

$$O(n)$$

$$O(n \log n)$$

$$O(n^2)$$

⋮

$$O(\log n) + O(n)$$



$$O(n)$$

---

Prove  $n^2$  is not  $O(1)$

Try it. If  $n^2$  was  $O(1)$ ,  
that means I could find a  $C$   
so that

$$n^2 \leq C \cdot 1 \quad (\text{at least for } n \geq N)$$

Can I find a  $C$  where

$$n^2 \leq C \quad ?$$

No, because any  $C$  I pick will eventually be smaller than  $n^2$  for some  $n$  big enough

---

Show that

$$10n^3 + 6n + 9 \text{ is } O(\underline{n^3})$$

Find  $C$  so that

$$10n^3 + 6n + 9 \leq C \underline{n^3} / n^3$$

$$10 + \frac{6}{n^2} + \frac{9}{n^3} \leq C$$

When  $n=1$ , the left side is

$$10 + \frac{6}{1^2} + \frac{9}{1^3} = 25$$

If  $N=3$ , the try  $n=3$

~~$$10 + \frac{6}{1^2}$$~~

---

```

fun printTriangle(int n) {
    for (i in 1..n) {
        for (j in 1..i) {
            print("*")
        }
        println()
    }
}

```

1 \*  
 2 \*\*  
 3 \*\*\*  
 ⋮  
 n \*\*\*\*\* \*  
 ↖ stars

How many \*s  
get printed?

How many times  
does the  
inner print  
happen?

$$1 + 2 + 3 + \dots + (n-2) + (n-1) + n$$

Diagram illustrating the pairing of terms in the sum  $1 + 2 + 3 + \dots + (n-2) + (n-1) + n$ . Brackets above and below the sequence show pairs of terms that sum to  $n+1$ . The first bracket connects 1 and  $n$ , the second connects 2 and  $n-1$ , and so on, with the last bracket connecting  $n/2$  and  $n/2 + 1$ .

$$= \underbrace{(n+1)}_{\text{common sum}} \underbrace{\frac{n}{2}}_{\# \text{ of pairs}}$$

$$= \frac{n^2}{2} + \frac{n}{2} \rightarrow O(n^2)$$

---


$$1 + 2 + 3 + \dots + 98 + 99 + 100$$

Diagram illustrating the pairing of terms in the sum  $1 + 2 + 3 + \dots + 98 + 99 + 100$ . Brackets above and below the sequence show pairs of terms that sum to 101. The first bracket connects 1 and 100, the second connects 2 and 99, and so on, with the last bracket connecting 50 and 51.

$$= 101(50)$$

$$\begin{array}{r} 50 \\ 5050 \end{array}$$

Stacks

9

3

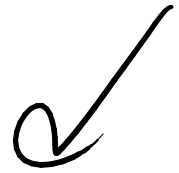
1

stack

remove

9

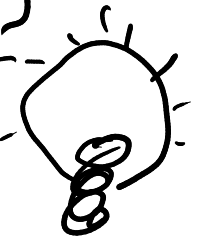
ADT



push - add onto stack Stack  
pop - remove from stack  
peek - look at what is on top  
isEmpty - true/false

A stack is an example of an  
Abstract Data Type (ADT)

↳ a description of a way of  
storing and manipulating data,  
but w/o the details of



how you implement it

There are lots of ways to implement a stack.

① List

② Linked List (coming)

③ Array

Two most common approaches  
I would try in Kotlin



less helpful but it's a nice example

Today, I am going to code a Stack w/ an array

In Kotlin, an array is a lot like a list, but its size is fixed. Can't be changed.

(So if array fills, you need to make a new bigger array and copy everything from the old array in.)

- inside, all lists really use arrays (Python, Kotlin, ...)