ConFoo.CA
DEVELOPER CONFERENCE
2024

I Completed All 9 Advents of Code: Lessons Learned

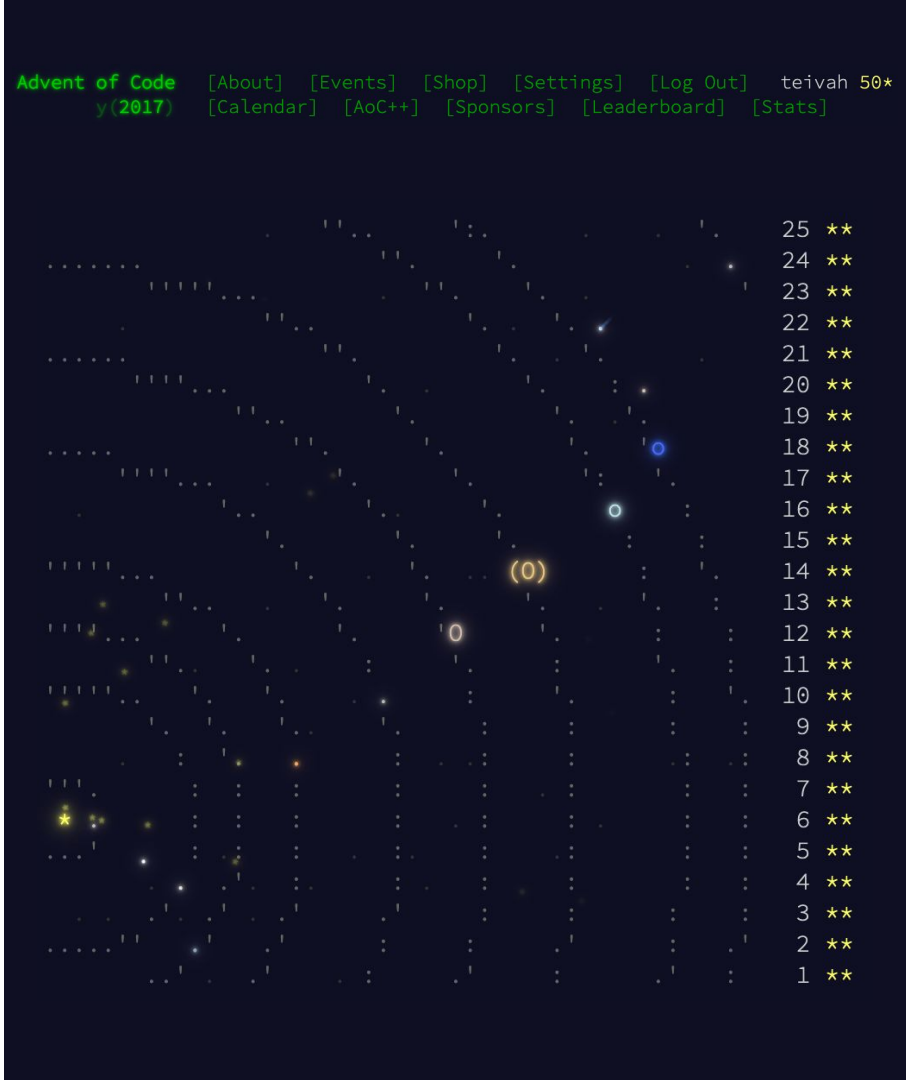
Teiva Harsanyi

Software Engineer at Google

100 Go Mistakes author


LeetCode

Advent of Code



Advent of Code

2018 day 18 (Game of Life like)

Part 1: Calculate after **10** steps

Part 2: Calculate after **1,000,000,000** steps

Advent of Code

55,000+

Lines of code written

Agenda

- Algorithms & Data Structures
- Coding
- Beyond Coding

Agenda

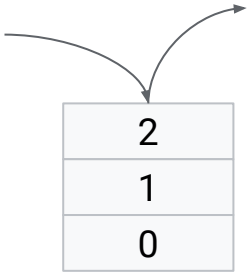
- Algorithms & Data Structures
- Coding
- Beyond Coding

Common Data Structures to Complete an AoC

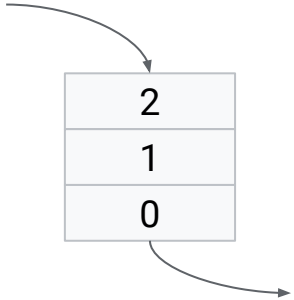
Array

Linked list

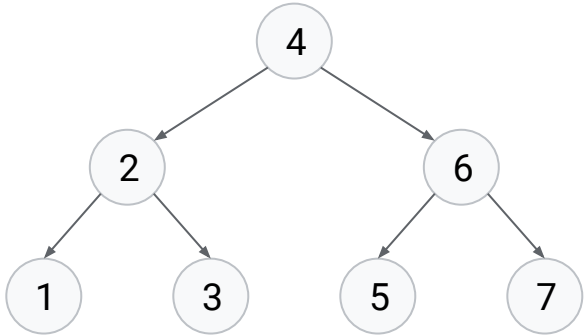
Stack



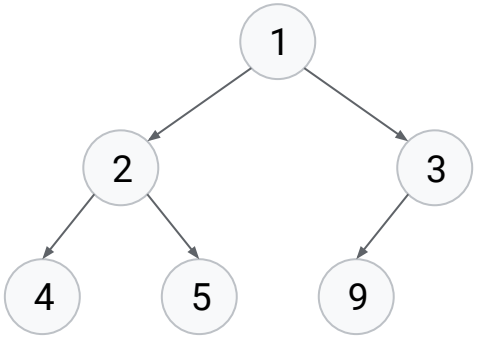
Queue



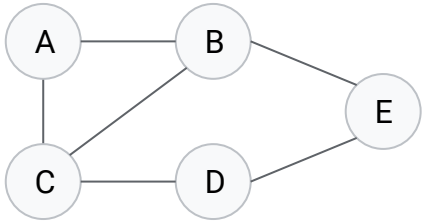
Tree



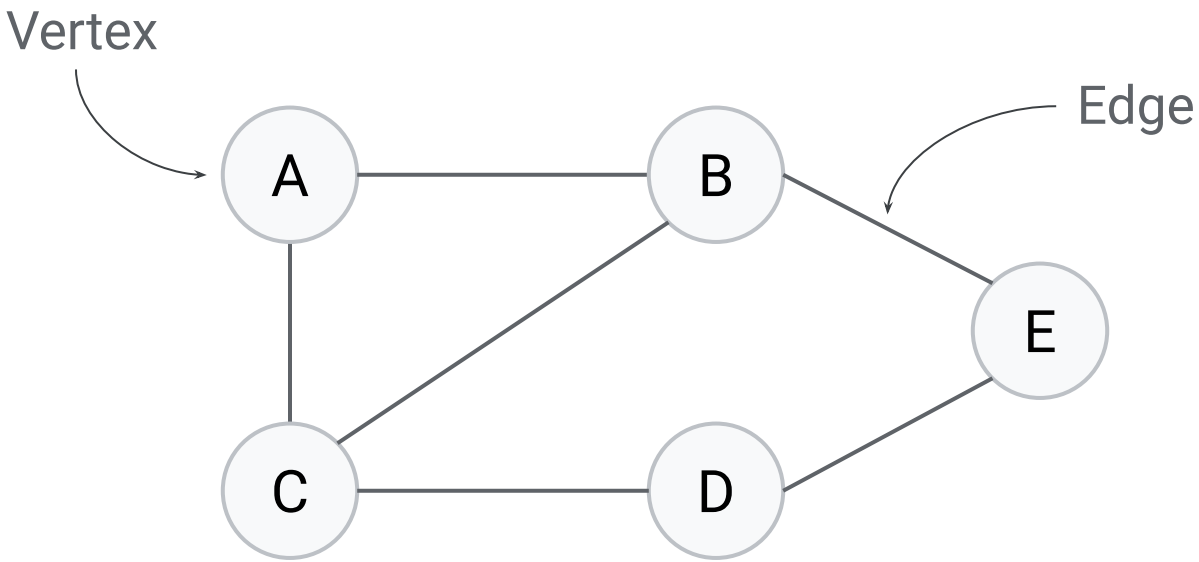
Heap



Graph



Graph

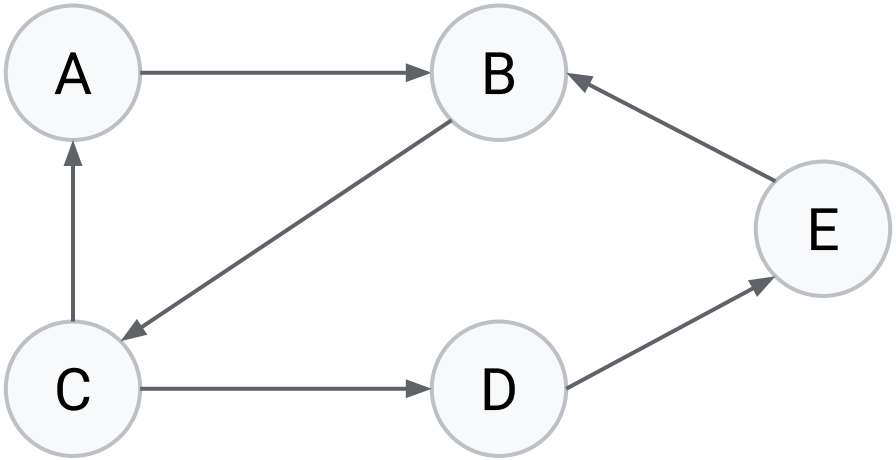


Graph: Characteristics

Directed vs. Undirected

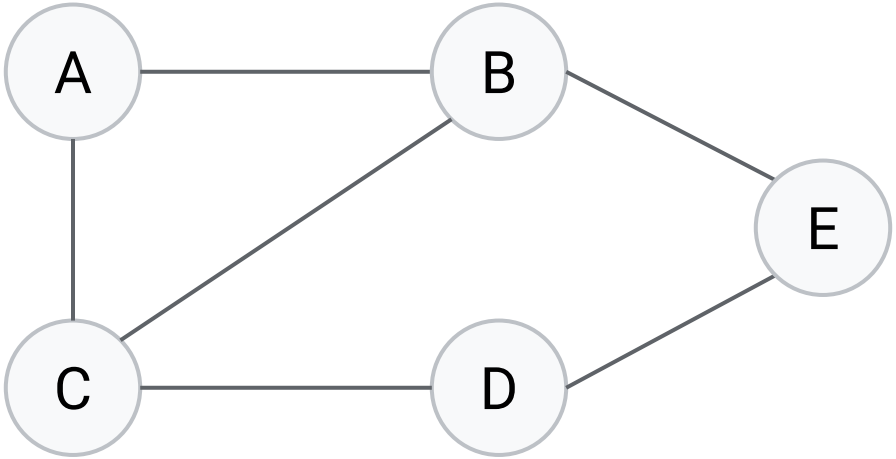
Graph: Characteristics

Directed vs. Undirected



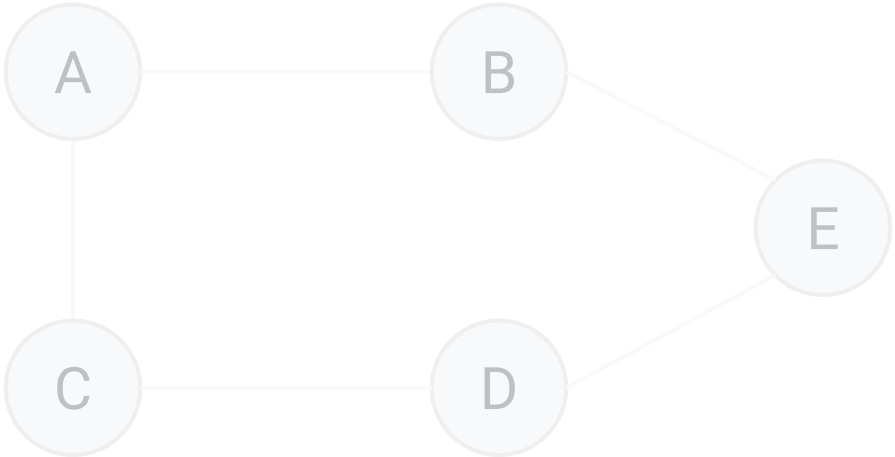
Graph: Characteristics

Directed vs. Undirected



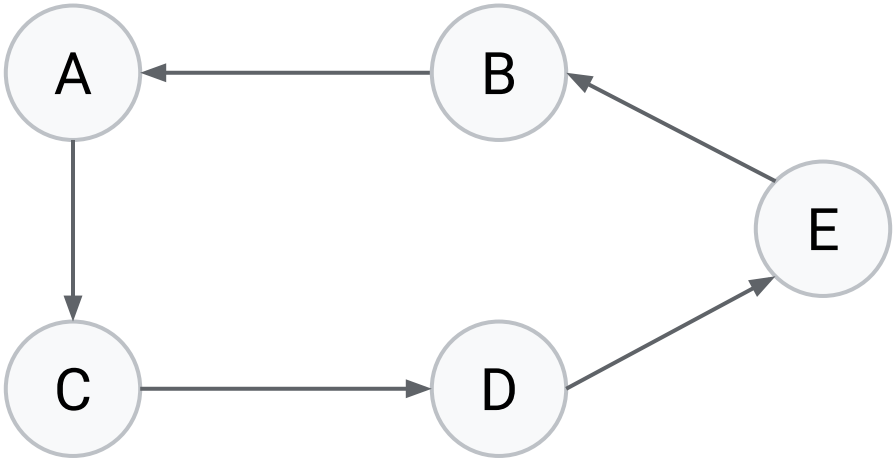
Graph: Characteristics

Cyclic vs. Acyclic



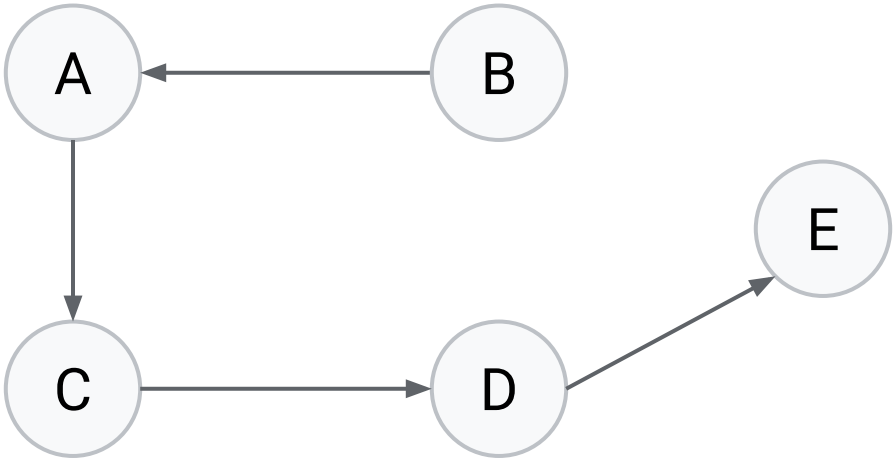
Graph: Characteristics

Cyclic vs. Acyclic



Graph: Characteristics

Cyclic vs. Acyclic



Graph: Algorithms

Depth-First Search (DFS)

Floyd-Warshall

Topological sort

Bellman-Ford

Minimum cut

Dijkstra

Breadth-First Search (BFS)

Graph: Algorithms

Depth-First Search (DFS)

Floyd-Warshall

Topological sort

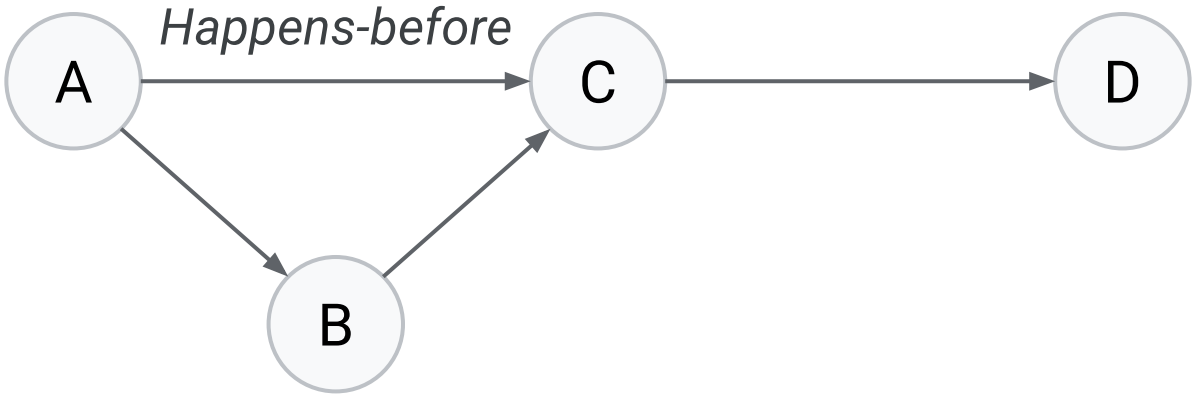
Bellman-Ford

Minimum cut

Dijkstra

Breadth-First Search (BFS)

Topological Sort



Applying topological sort:

Topological Sort

2022 day 21: list of yelling monkeys

```
root: foo + bar
baz: 5
cczh: sllz + lgvd
zczc: 2
ptdq: humn - dvpt
dvpt: 3
lfqf: 4
humn: 5
ljgn: 2
bar: drzm * baz
sllz: 4
foo: cczh / lfqf
lgvd: ljgn * ptdq
drzm: hmdt - zczc
hmdt: 32
```

Topological Sort

2022 day 21: list of yelling monkeys Problem: What is `root` yelling?

```
root: foo + bar
baz: 5
cczh: sllz + lgvd
zczc: 2
ptdq: humn - dvpt
dvpt: 3
lfqf: 4
humn: 5
ljgn: 2
bar: drzm * baz
sllz: 4
foo: cczh / lfqf
lgvd: ljgn * ptdq
drzm: hmdt - zczc
hmdt: 32
```

Topological Sort

2022 day 21: list of yelling monkeys Problem: What is `root` yelling?

```
root: foo + bar
baz: 5
cczh: sllz + lgvd
zczc: 2
ptdq: humn - dvpt
dvpt: 3
lfqf: 4
humn: 5
ljgn: 2
bar: drzm * baz
sllz: 4
foo: cczh / lfqf
lgvd: ljgn * ptdq
drzm: hmdt - zczc
hmdt: 32
```

Topological Sort

2022 day 21: list of yelling monkeys Problem: What is `root` yelling?

```
root: foo + bar
baz: 5
cczh: sllz + lgvd
zczc: 2
ptdq: humn - dvpt
dvpt: 3
lfqf: 4
humn: 5
ljgn: 2
bar: drzm * baz
sllz: 4
foo: cczh / lfqf
lgvd: ljgn * ptdq
drzm: hmdt - zczc
hmdt: 32
```

Topological Sort

2022 day 21: list of yelling monkeys Problem: What is `root` yelling?

```
root: foo + bar
baz: 5
cczh: sllz + lgvd
zczc: 2
ptdq: humn - dvpt
dvpt: 3
lfqf: 4
humn: 5
ljgn: 2
bar: drzm * baz
sllz: 4
foo: cczh / lfqf
lgvd: ljgn * ptdq
drzm: hmdt - zczc
hmdt: 32
```

Topological Sort

2022 day 21: list of yelling monkeys

```
root: foo + bar
baz: 5
cczh: sllz + lgvd
zczc: 2
ptdq: humn - dvpt
dvpt: 3
lfqf: 4
humn: 5
ljgn: 2
bar: drzm * baz
sllz: 4
foo: cczh / lfqf
lgvd: ljgn * ptdq
drzm: hmdt - zczc
hmdt: 32
```

Problem: What is `root` yelling?

How can we solve this problem?

1. **Topological sort**

```
root->bar->baz->drzm->zczc->hmdt->foo->
lfqf->cczh->lgvd->ptdq->dvpt->humn->ljgn->
sllz
```

2. Backwards **traversal**

Topological Sort

Different applications:

- Task scheduling
- Package dependency resolution
- Building an execution plan from a DB query
- Etc.

Recursion

2023 day 12:

`?????..## 2,1,2`

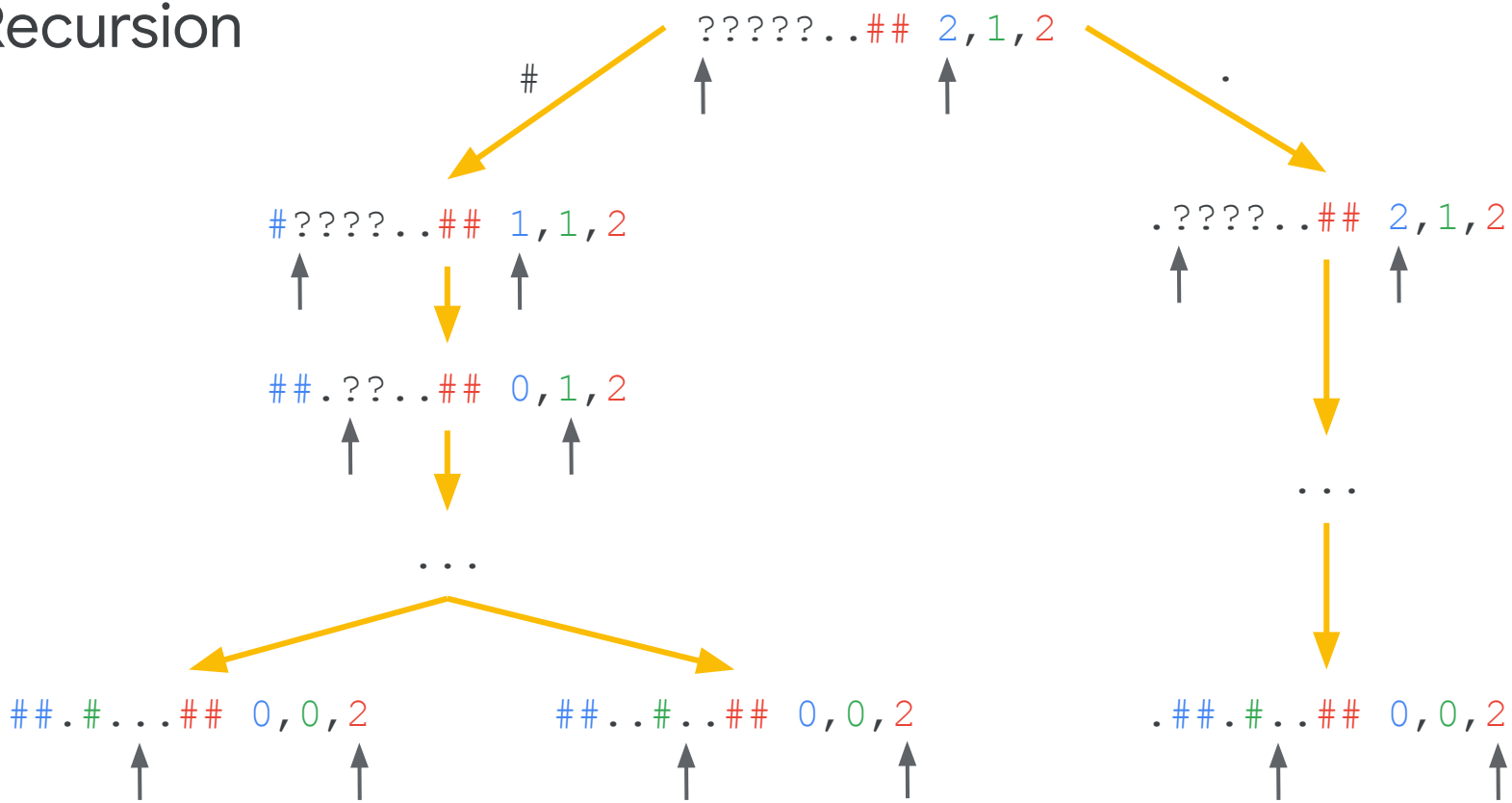
Count the number of **valid arrangements**

Constraint: Set of blocks have to be separated by at least one space

Solution 1:	Solution 2:	Solution 3:
##.#...##	##..#..##	.##.#..##

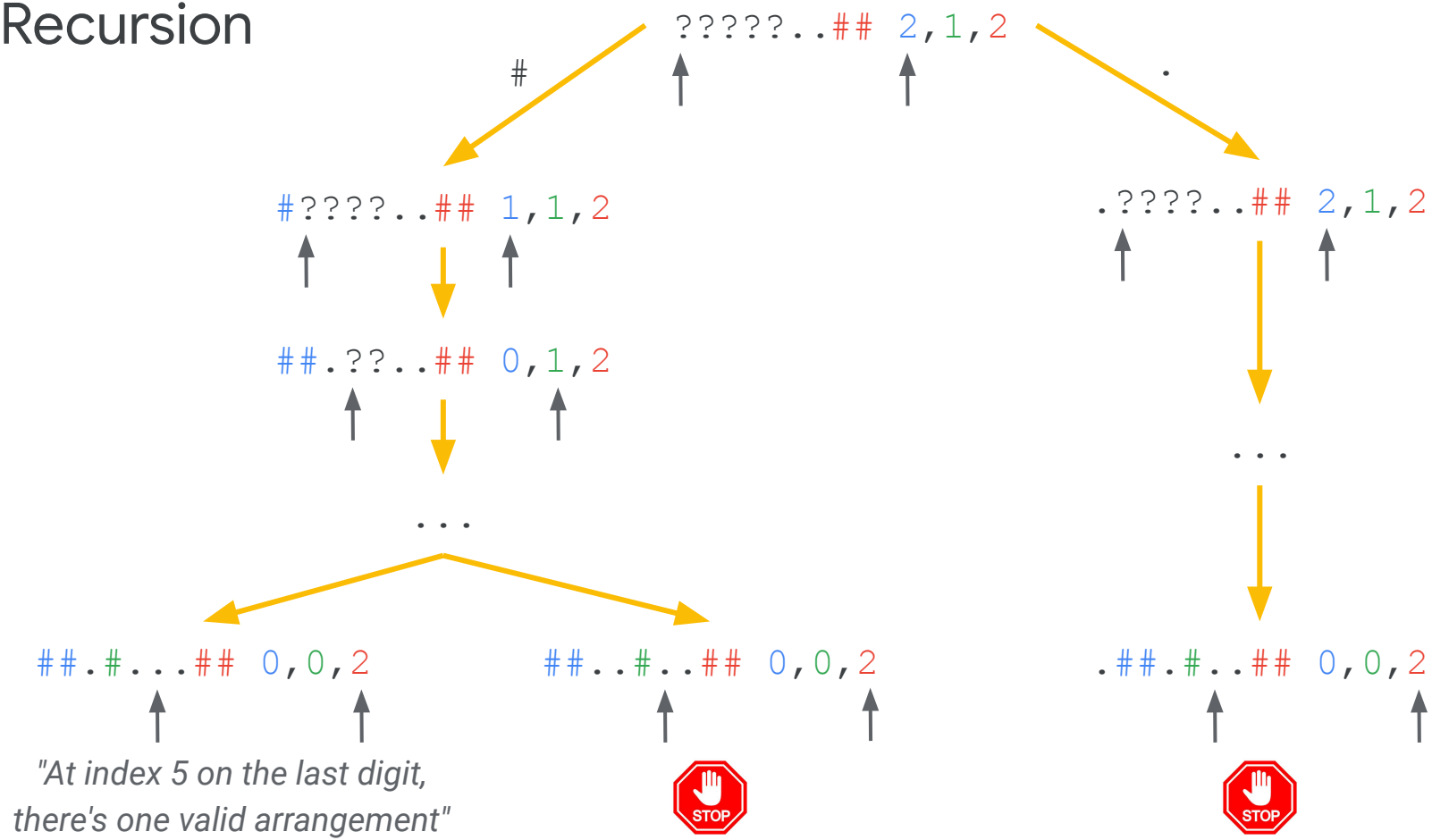
Recursion to the rescue

Recursion



At this stage, all three sub-problems are **identical**

Recursion


?????..## 2,1,2
#
.
#????..## 1,1,2
.????..## 2,1,2
##.??..## 0,1,2
...
...
##.#...## 0,0,2
##..#..## 0,0,2
.##.#..## 0,0,2
"At index 5 on the last digit, there's one valid arrangement"
STOP
STOP

Recursion

Recursion is a key concept

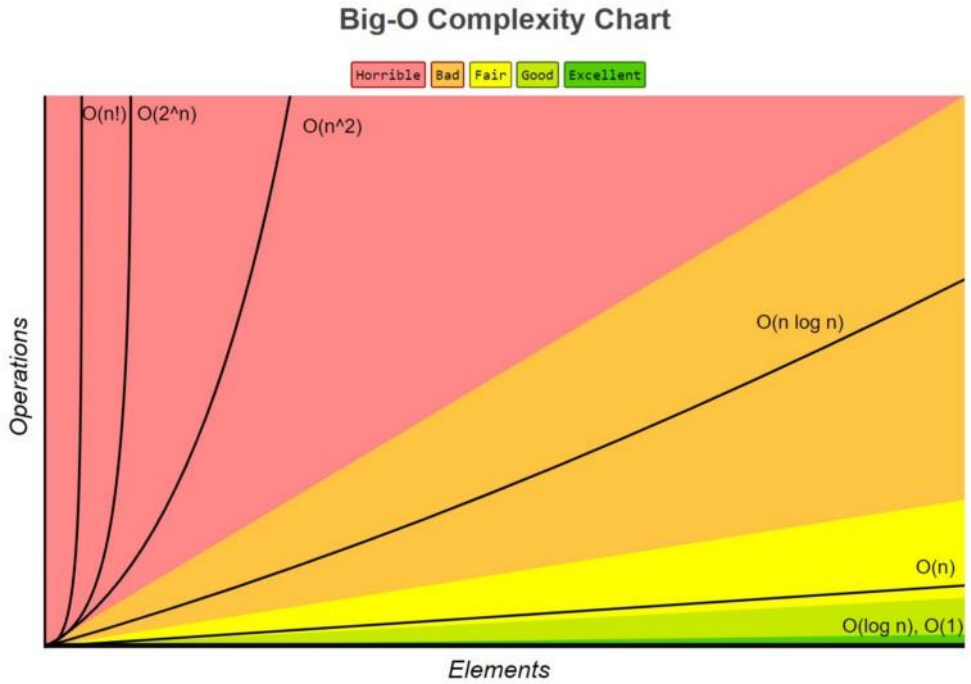
Start simple, think about DP next

DP mirrors a fundamental CS concept

trading time for space

Big O

Used to model how an algorithm will **scale**



Big O

2023 day 5:

- Iterate over a range `r` of numbers
- For each number, perform `n` transformations

First solution	Second solution	Third solution
Brute force	Multithreading to parallelize the iteration on `r`	Using binary search on `n`, no concurrency
Time complexity: O(`r`*`n`)	Time complexity: O(`r`*`n`)	Time complexity: O(`r`*log(`n`))
Execution: 84 secs	Execution: 24 secs	Execution: < 1 sec

Big O

- Optimize execution **after** algorithmic optimization
- The right data structures for the right job
- Big O is also for space complexity

Beyond Big O

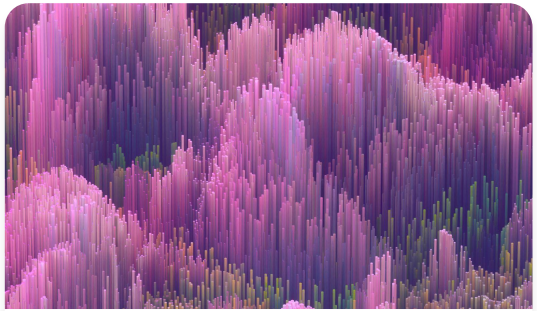
- How long does it take to get **100 million random entries** from a **map** containing **~1,000 values**?
- Case in point: 2017 day 15 => 40 million operations in 20ms
- Regular practice builds intuition for **runtime estimation**

DO *YOU* KNOW HOW MUCH YOUR COMPUTER CAN DO IN A SECOND?

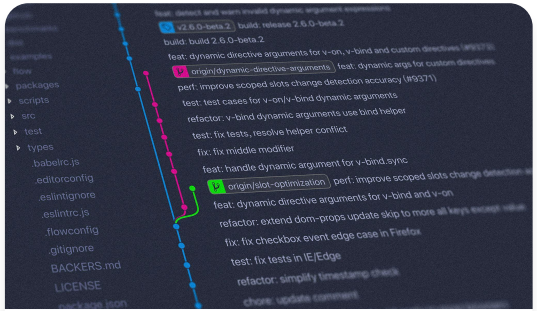
computers-are-fast.github.io

Algorithms & Data Structures

Crucial in many technical interviews



Proficiency in data structures is **key** in our **data-centric world**



Comfort with these topics
⇒ **Better developer**

Agenda

- Algorithms & Data Structures
- Coding
- Beyond Coding

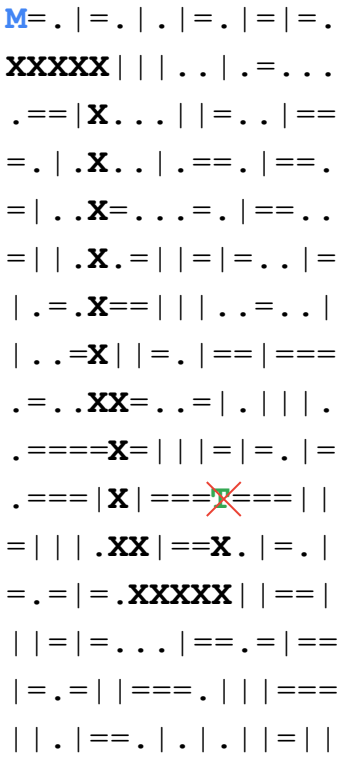
Navigating Future Requirements

- 2 stars per day
- When solving part 1, creating a **generic solution** is tempting
- Almost never works
- Holds true in our **daily work**

Navigating Future Requirements

- Predicting the future is complex
- **Speculative** or **premature work** is the root of all evil 😈
- YAGNI

Mutability as a Source of Mistakes



2018 day 22: Find the path from **M** to **T**

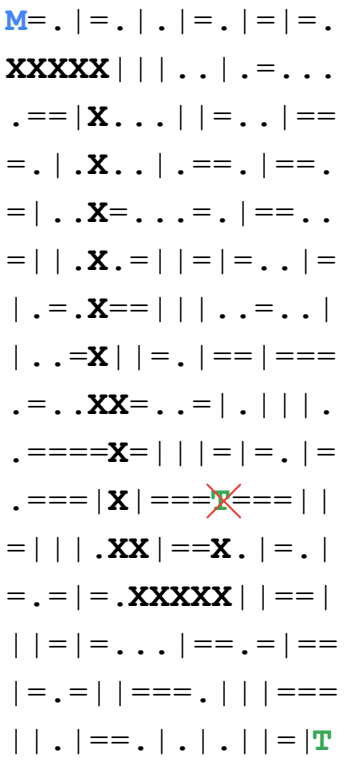
Map is **generated**

Path can go below **T**

```
func solve(targetCol, targetRow int) int {
  const extraBuffer = 100

  targetCol += extraBuffer
  targetRow += extraBuffer

  // Generate the map from (0,0) to (targetCol, targetRow)

  // ...
}
```

⁉️

Mutability as a Source of Mistakes



2018 day 22: Find the path from `M` to `T`

Map is **generated**

Path can go below `T`

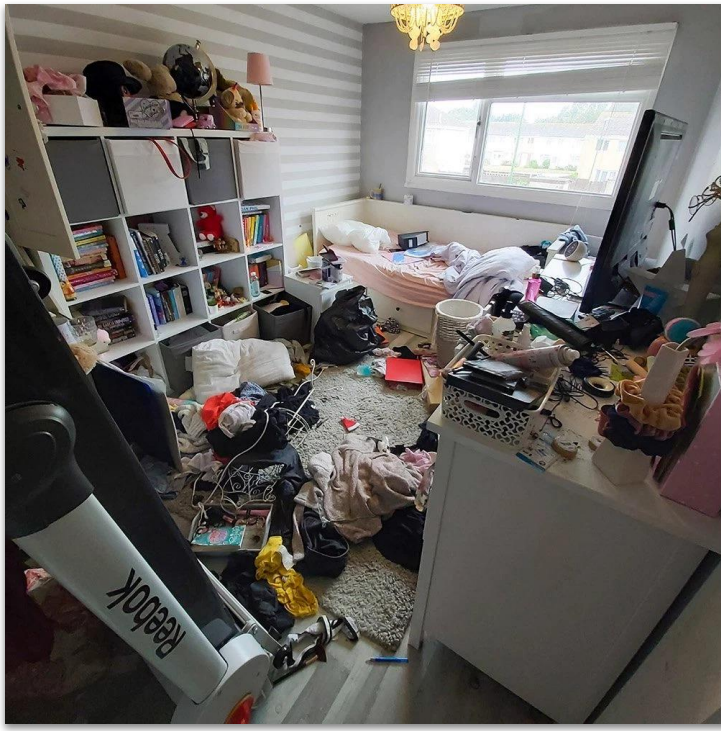
```
func solve(targetCol, targetRow int) int {
  const extraBuffer = 100

  targetCol += extraBuffer
  targetRow += extraBuffer

  // Generate the map from (0,0) to (targetCol, targetRow)

  // ...
}
```

⁉️

Mutability as a Source of Mistakes

Refactoring

My solution
when I get the **2 stars**



Refactoring

My solution
when I get the **2 stars**

My solution
when I **publish it on Github**

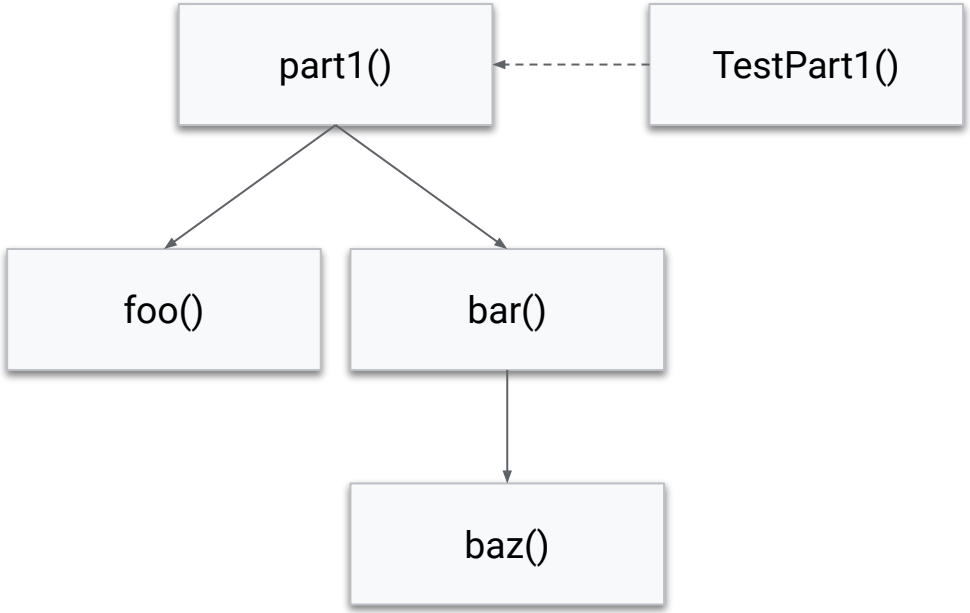
Refactoring

First version:

Refactoring

Final version:

Made possible thanks to **behavior testing**



Refactoring

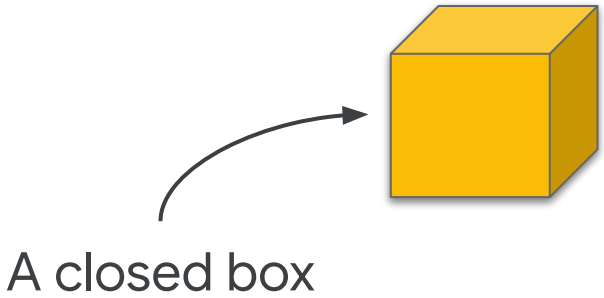
Behavior testing != Behavior-Driven Development (BDD)

- ~~BDD: **promote collaboration** between tech and non-tech people~~

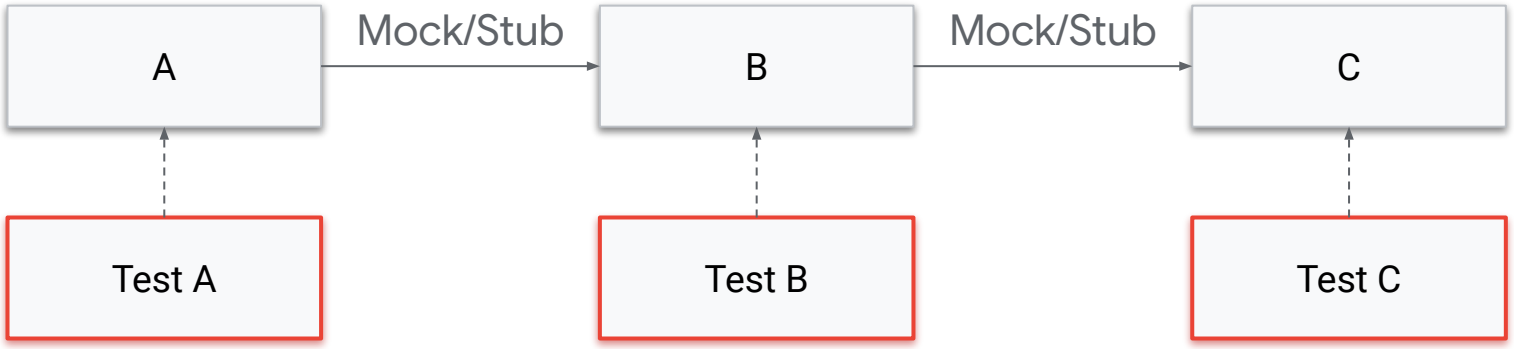
Refactoring

Behavior testing != Behavior-Driven Development (BDD)

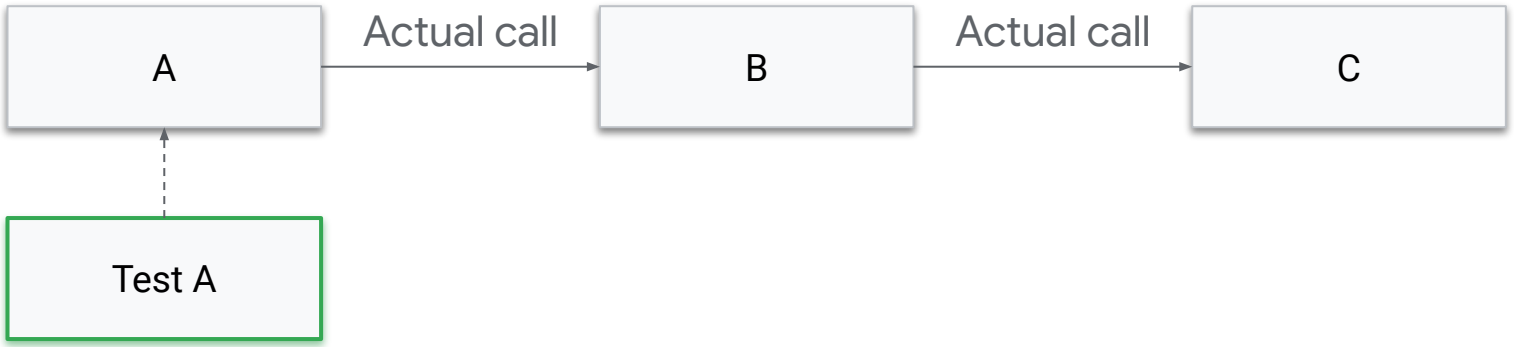
- ~~BDD: **promote collaboration** between tech and non-tech people~~
- Behavior testing: testing technique to focus on how the system must **behave externally**



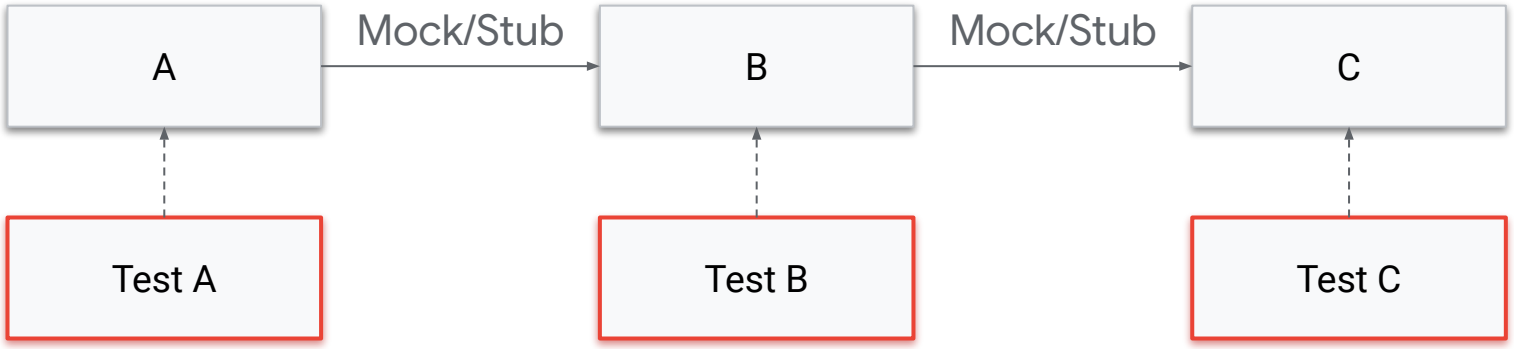
The approach I try **to avoid**:



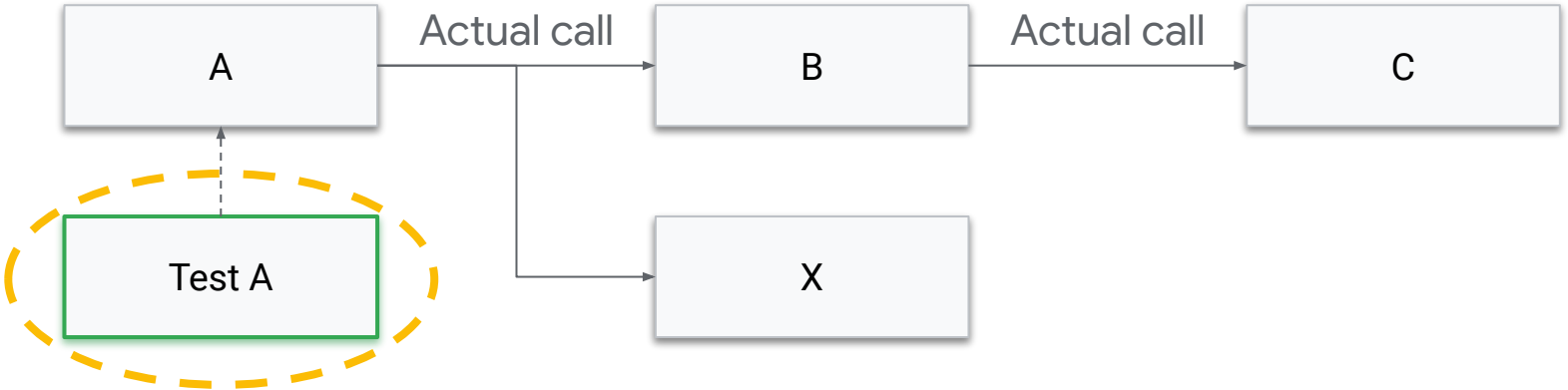
The approach I try **to follow**:



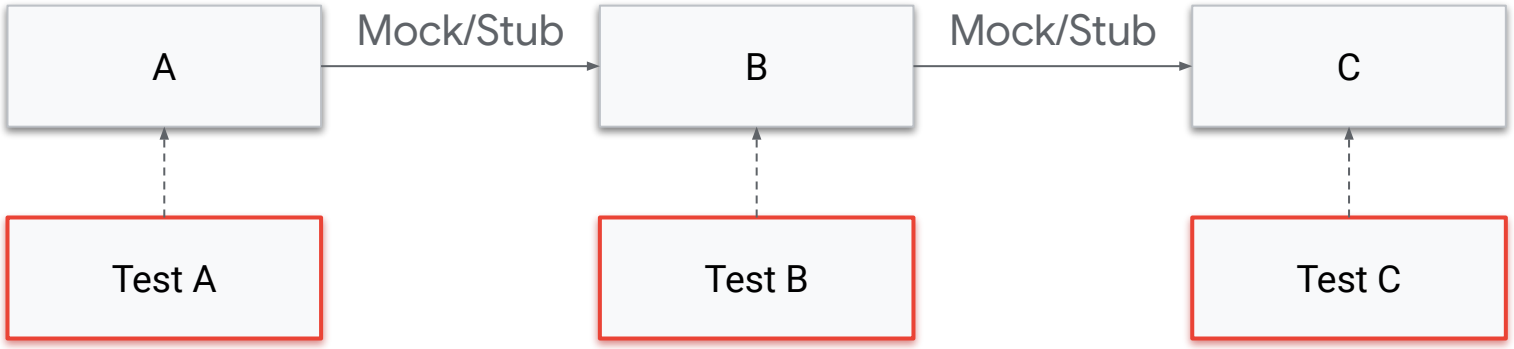
The approach I try **to avoid**:



The approach I try **to follow**:

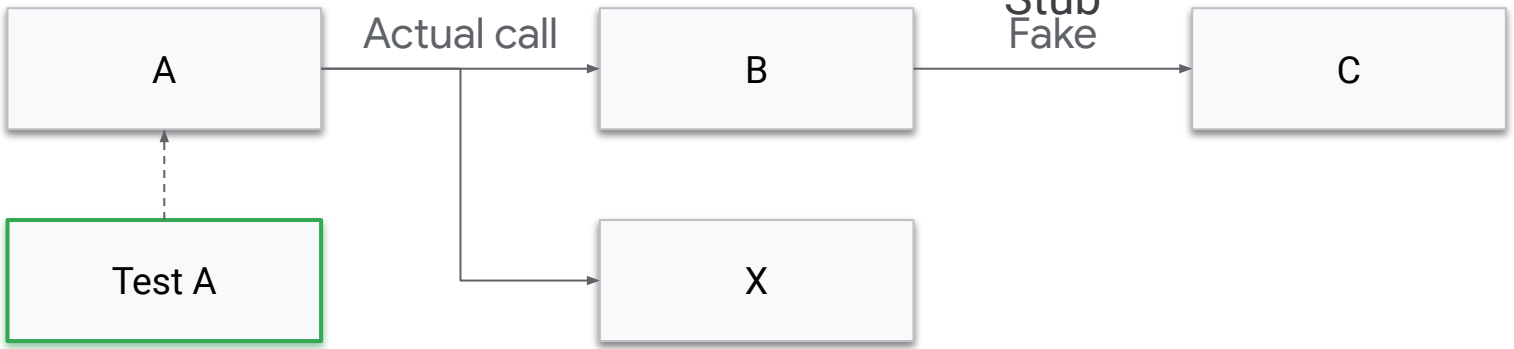


The approach I try **to avoid**:



The approach I try **to follow**:

❗ Fake > Mock & Stub



Behavior testing

Different from BDD

Clearer and more maintainable tests

Enables refactoring

Agenda

- Algorithms & Data Structures
- Coding
- **Beyond Coding**

Fostering a Problem-Solving Mindset

Subconscious
problem solving

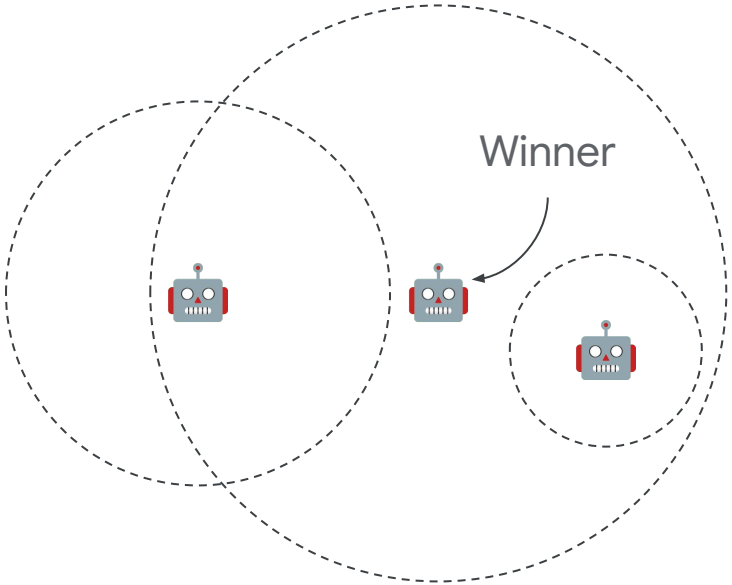
Developing the skill of seeing the **big picture** in a problem

The power of creativity and thinking **outside the box**

Fostering a Problem-Solving Mindset

2018 day 23: list of robots with a 3D coordinate and a range

Which one has the **most** robots in its range?



One *slight* issue…

```
pos=<-34870395,34498817,-2843154>, r=96244937
pos=<-52741579,9875242,37136273>, r=89509114
pos=<23303891,41664349,2510522>, r=63042453
pos=<10573027,54782809,49932958>, r=97928881
pos=<30268215,-1711562,83940876>, r=91888282
pos=<33400284,54035761,31240407>, r=56780390
pos=<56263947,16641002,21741916>, r=78212857
pos=<52254266,-17802752,31767960>, r=98620322
pos=<74669464,32372925,70139553>, r=93403053
pos=<41042577,51422528,45494381>, r=54180555
...
```

Fostering a Problem-Solving Mindset

2018 day 23: list of robots with a 3D coordinate and a range

Which one has the **most** robots in its range?

Divide by 10,000,000

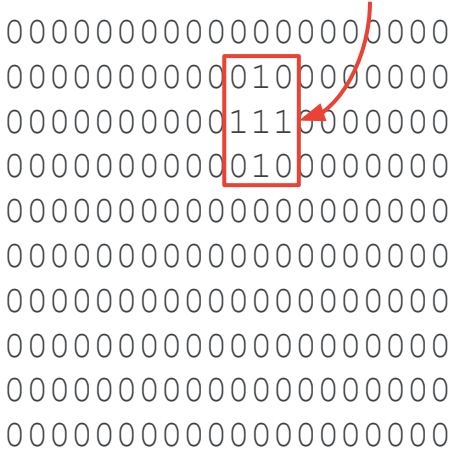
```
pos=<-34870395,34498817,-2843154>, r=96244937
pos=<-52741579,9875242,37136273>, r=89509114
pos=<23303891,41664349,2510522>, r=63042453
pos=<10573027,54782809,49932958>, r=97928881
pos=<30268215,-1711562,83940876>, r=91888282
pos=<33400284,54035761,31240407>, r=56780390
pos=<56263947,16641002,21741916>, r=78212857
pos=<52254266,-17802752,31767960>, r=98620322
pos=<74669464,32372925,70139553>, r=93403053
pos=<41042577,51422528,45494381>, r=54180555
...
```

Fostering a Problem-Solving Mindset

2018 day 23: list of robots with a 3D coordinate and a range

Which one has the **most** robots in its range?

Delve into this cluster



Divide by 10,000,000

```
pos=<-34870395,34498817,-2843154>, r=96244937
pos=<-52741579,9875242,37136273>, r=89509114
pos=<23303891,41664349,2510522>, r=63042453
pos=<10573027,54782809,49932958>, r=97928881
pos=<30268215,-1711562,83940876>, r=91888282
pos=<33400284,54035761,31240407>, r=56780390
pos=<56263947,16641002,21741916>, r=78212857
pos=<52254266,-17802752,31767960>, r=98620322
pos=<74669464,32372925,70139553>, r=93403053
pos=<41042577,51422528,45494381>, r=54180555
...
```

Fostering a Problem-Solving Mindset

2018 day 23: list of robots with a 3D coordinate and a range

Which one has the **most** robots in its range?

Delve into this cluster

Divide by 1,000,000

```
pos=<-34870395,34498817,-2843154>, r=96244937
pos=<-52741579,9875242,37136273>, r=89509114
pos=<23303891,41664349,2510522>, r=63042453
pos=<10573027,54782809,49932958>, r=97928881
pos=<30268215,-1711562,83940876>, r=91888282
pos=<33400284,54035761,31240407>, r=56780390
pos=<56263947,16641002,21741916>, r=78212857
pos=<52254266,-17802752,31767960>, r=98620322
pos=<74669464,32372925,70139553>, r=93403053
pos=<41042577,51422528,45494381>, r=54180555
...
```

Fostering a Problem-Solving Mindset

2018 day 23: list of robots with a 3D coordinate and a range
Which one has the **most** robots in its range?

Thinking **outside the box**

Repeat the process to find the solution

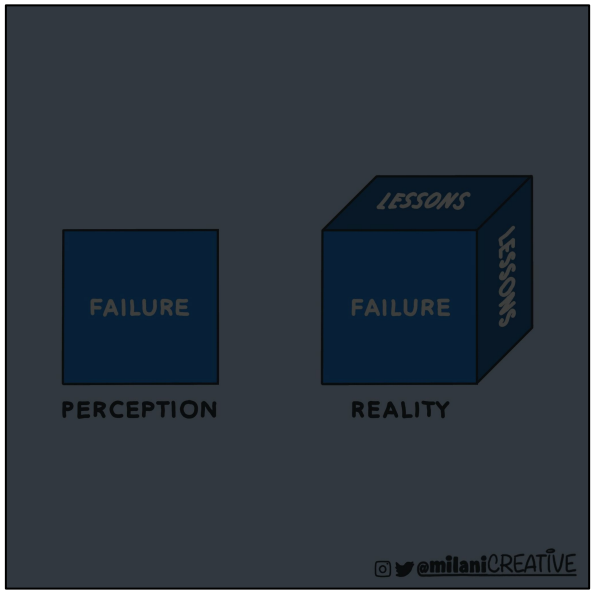
```
pos=<-34870395,34498817,-2843154>, r=96244937
pos=<-52741579,9875242,37136273>, r=89509114
pos=<23303891,41664349,2510522>, r=63042453
pos=<10573027,54782809,49932958>, r=97928881
pos=<30268215,-1711562,83940876>, r=91888282
pos=<33400284,54035761,31240407>, r=56780390
pos=<56263947,16641002,21741916>, r=78212857
pos=<52254266,-17802752,31767960>, r=98620322
pos=<74669464,32372925,70139553>, r=93403053
pos=<41042577,51422528,45494381>, r=54180555
...
```

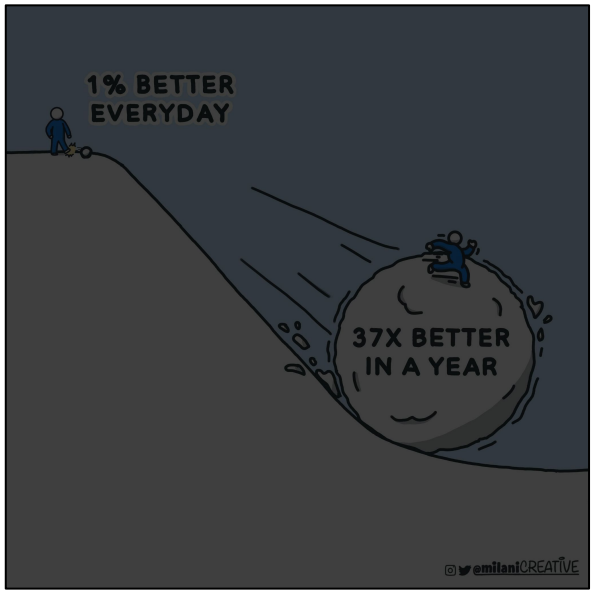
A Journey of Resilience

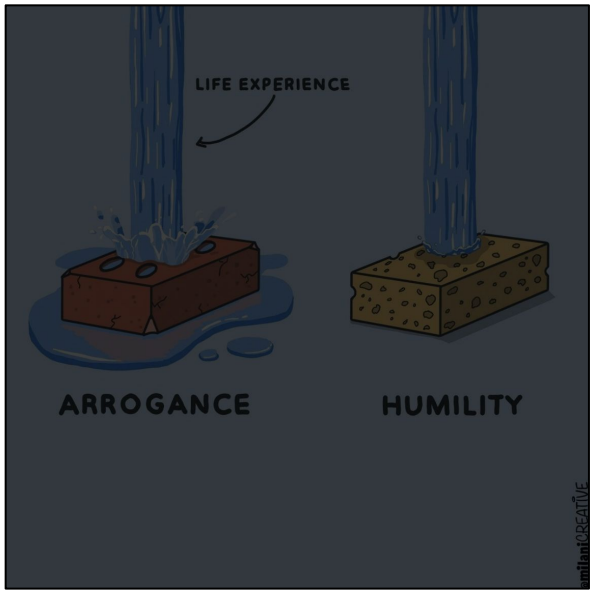
> *It's the first time I am actively proceeding with AoC but I think I won't be able to go far. That's why I'm questioning my abilities.*

> *Does it make me a bad programmer if I can't solve some of these?*

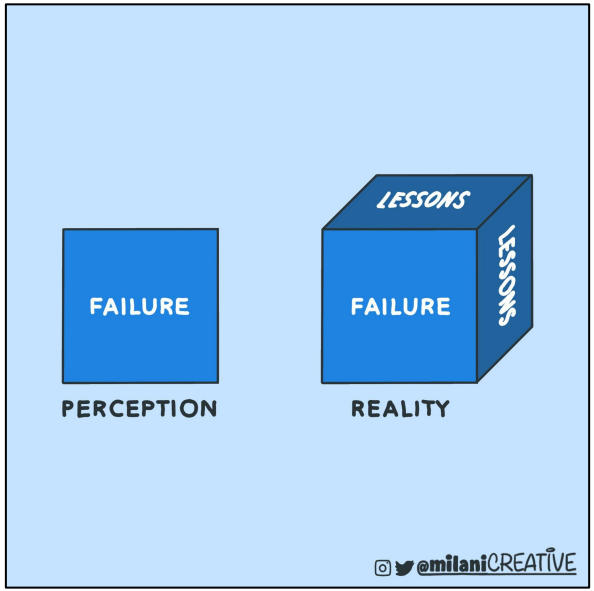
A Journey of Resilience

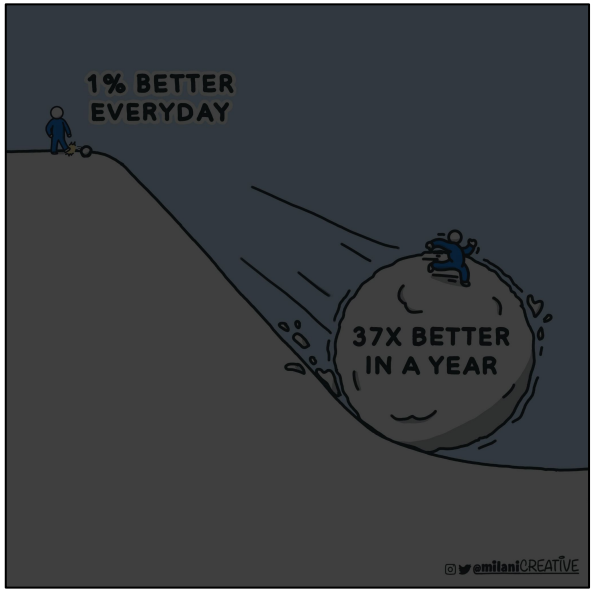


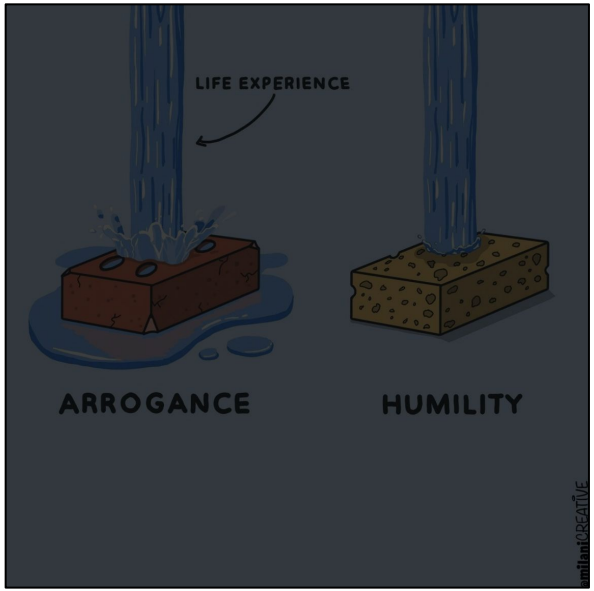




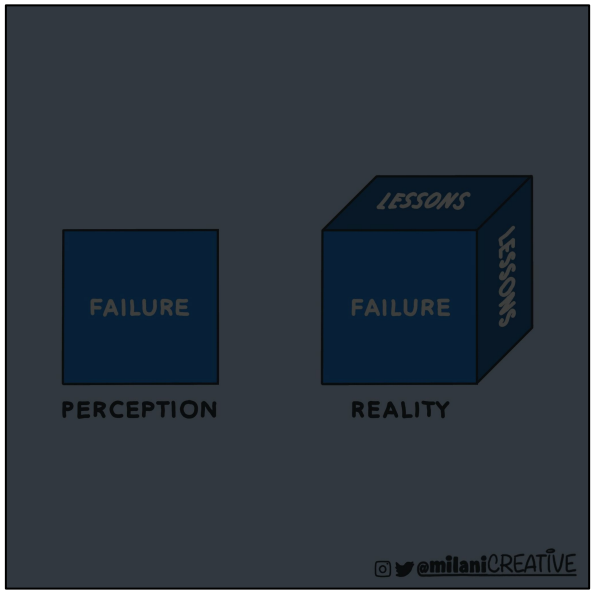
A Journey of Resilience

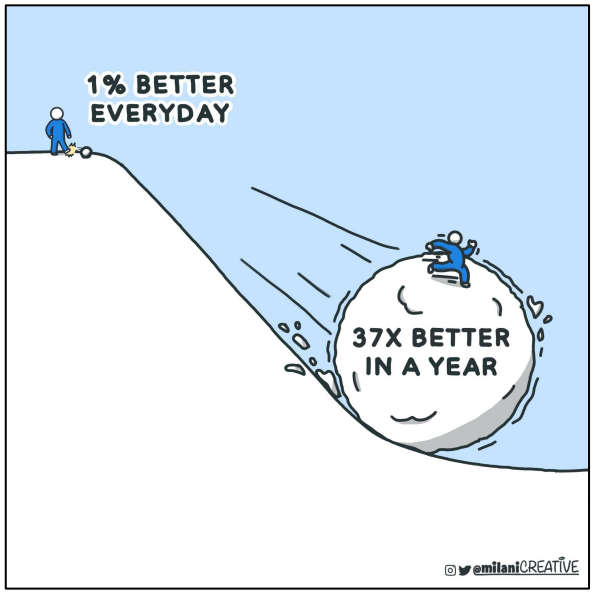


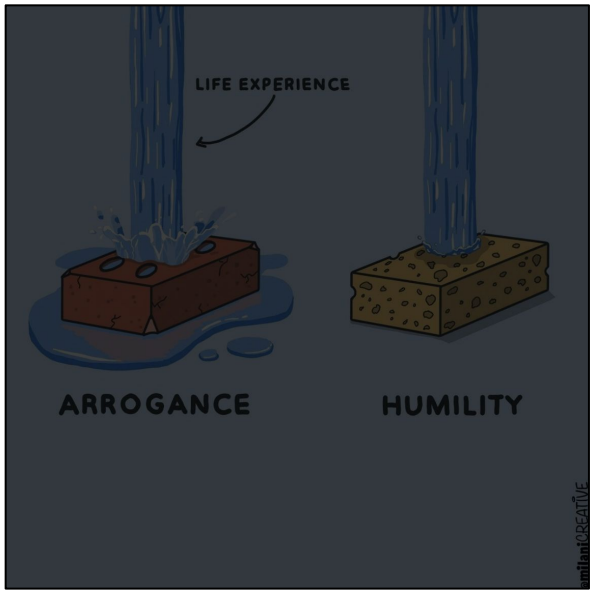




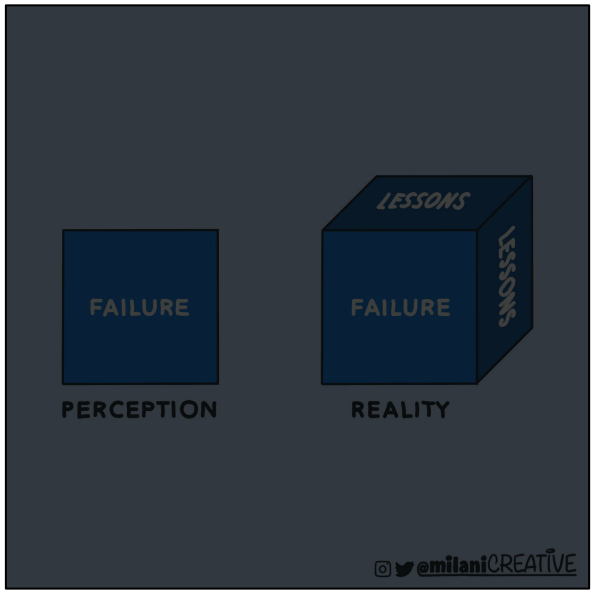
A Journey of Resilience

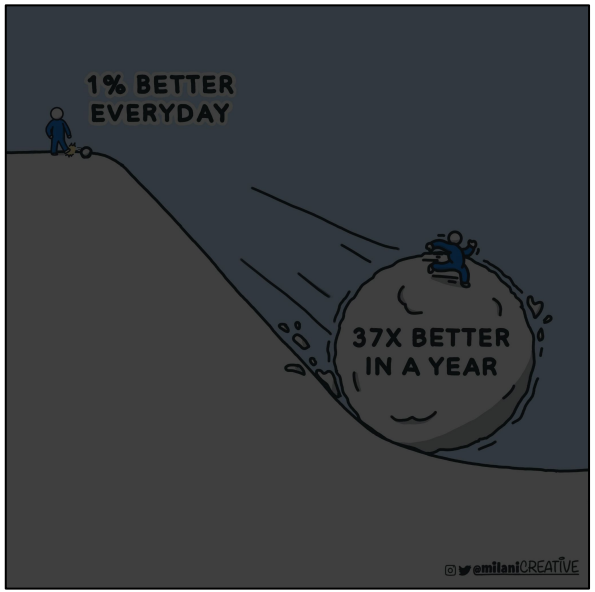


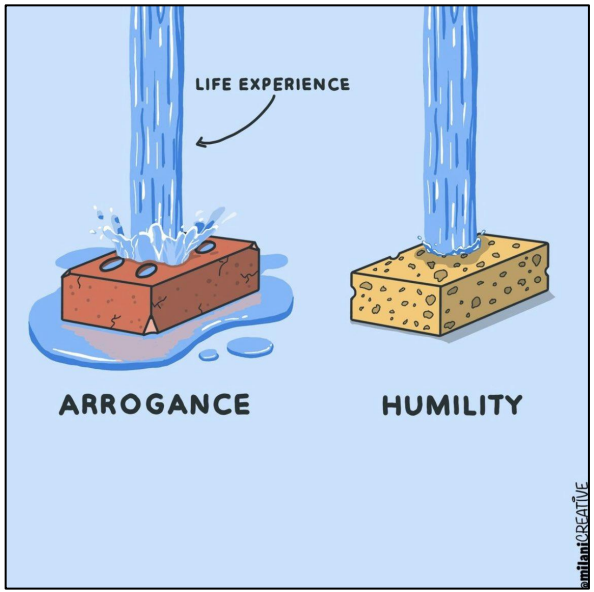




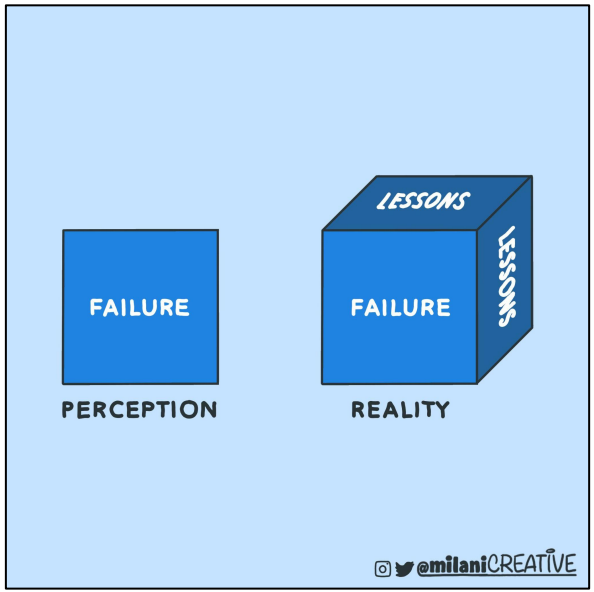
A Journey of Resilience

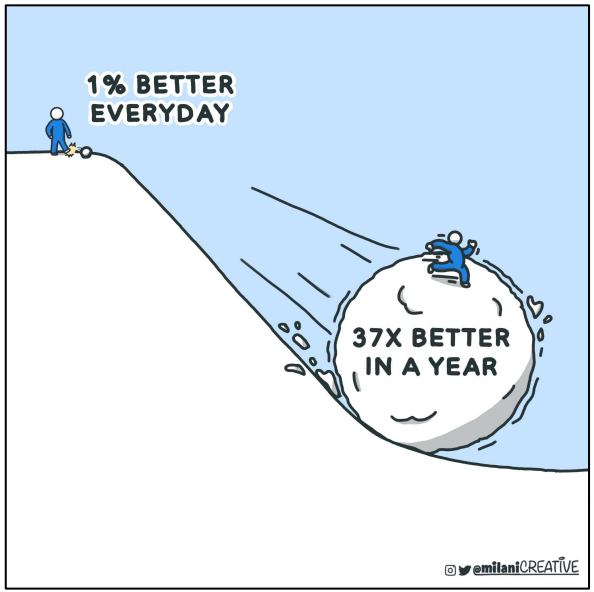


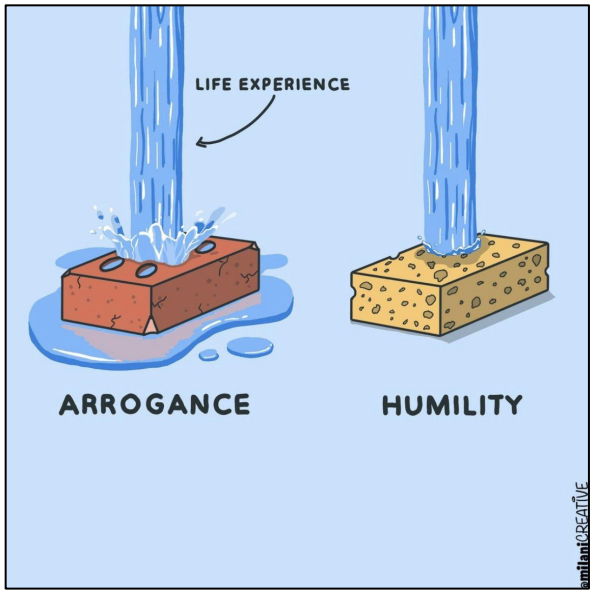




A Journey of Resilience







teivah.io/confoo-aoc

teivah

Any feedback?