Push Swap — A journey to find most efficient sorting algorithm

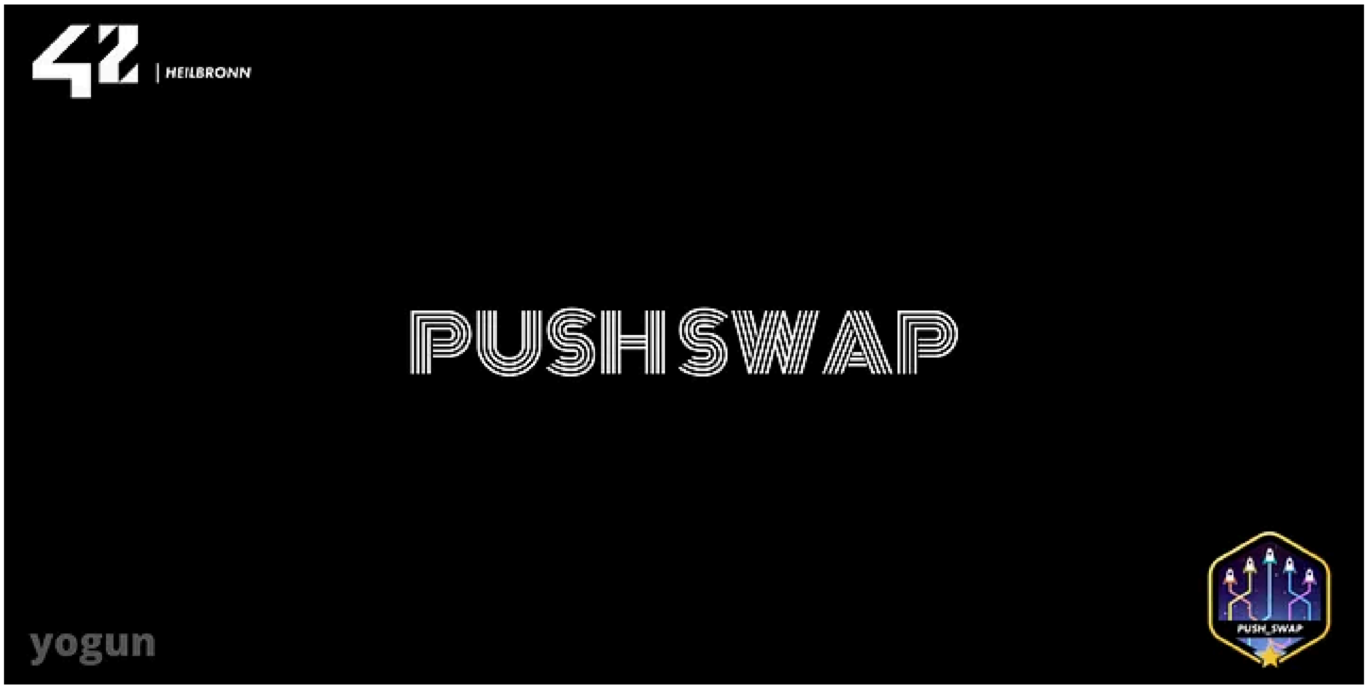
A sorting algorithm for the push_swap project of 42 Programming School

A. Yigit Ogun · Follow
11 min read · Sep 10, 2022

362 6

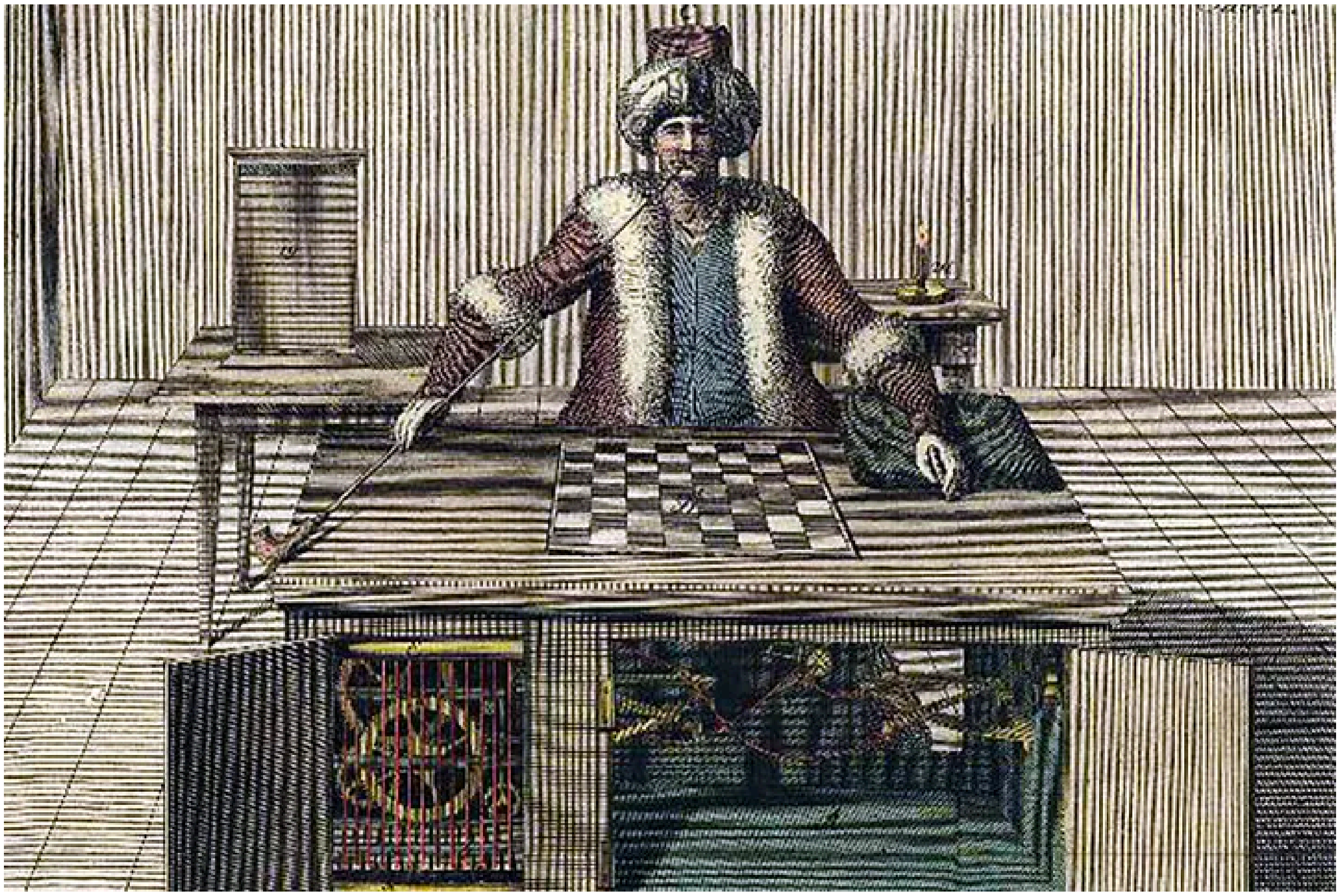
Push_swap is an algorithm project at school 42. I've passed with a full score of 125/125 and I'll try to explain how I've solved it.

I recommend you to read this article and apply my algorithm only if you are as **lazy** as I am, 😂. (I was way too lazy to break down the other complicated solutions such as Radix Sort.)



As a 42-Heilbronn student, I always search for some resources or blog posts when I start off a new project. Because I like to start my projects by reading some brief summaries. I've read some blog posts about the push swap project. One of them was suggesting to use **Radix Sort** while **another one** was suggesting to divide stacks in some little chunks. But I was looking for something more straight forward which would bring me 125 out of 125.

By the way, if you are wondering what is the push swap project and what are the rules, there are enough resources on the web. You can ***click here*** to read the subject and rules. If you don't know the topic and rules, you better read it first.


Mechanical Turk

A Brief Summary

You start with two empty stacks: **a** and **b**. You are given a random list of integers via command line arguments.

Only these moves are allowed:

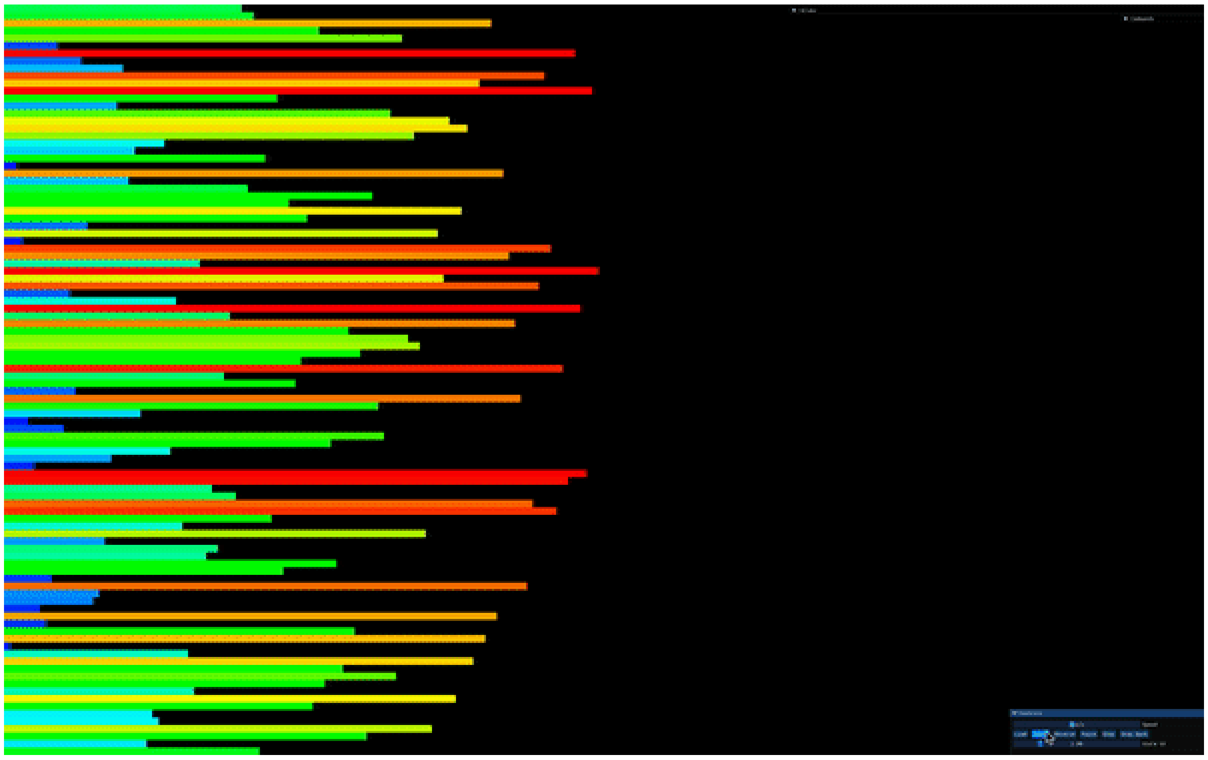
- `sa` : swap a - swap the first 2 elements at the top of stack a. Do nothing if there is only one or no elements).
- `sb` : swap b - swap the first 2 elements at the top of stack b. Do nothing if there is only one or no elements).
- `ss` : `sa` and `sb` at the same time.

- `pa` : push a - take the first element at the top of b and put it at the top of a. Do nothing if b is empty.
- `pb` : push b - take the first element at the top of a and put it at the top of b. Do nothing if a is empty.
- `ra` : rotate a - shift up all elements of stack a by 1. The first element becomes the last one.
- `rb` : rotate b - shift up all elements of stack b by 1. The first element becomes the last one.
- `rr` : `ra` and `rb` at the same time.
- `rra` : reverse rotate a - shift down all elements of stack a by 1. The last element becomes the first one.
- `rrb` : reverse rotate b - shift down all elements of stack b by 1. The last element becomes the first one.
- `rrr` : `rra` and `rrb` at the same time.

At the end, stack b must be empty empty and all integers must be in stack a, sorted in ascending order.

Which algorithm I've used?

Nothing special. Let's call it the Turk Algorithm. Not because of I'm Turkish but because of the Mechanical Turk story. I will call it like that because this algorithm is hard coded and it is not elegant. Likewise Mechanical Turk, lol. Okay I better not divert the subject.



The Turk Algorithm — I've used push swap visualizer in order to get this beautiful visualization

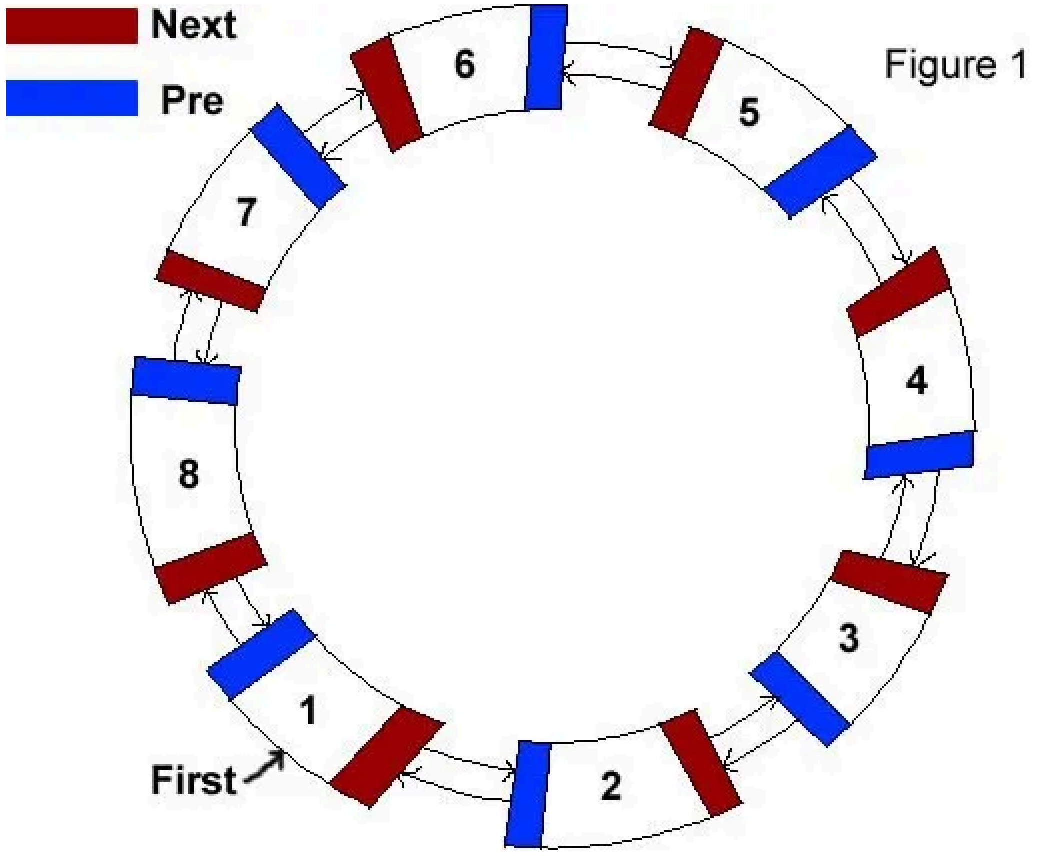
First of all, it's important to mention that I will walk through without explaining any code. But, you can **click here** to go to my repository and check my code to understand things better.

Getting started

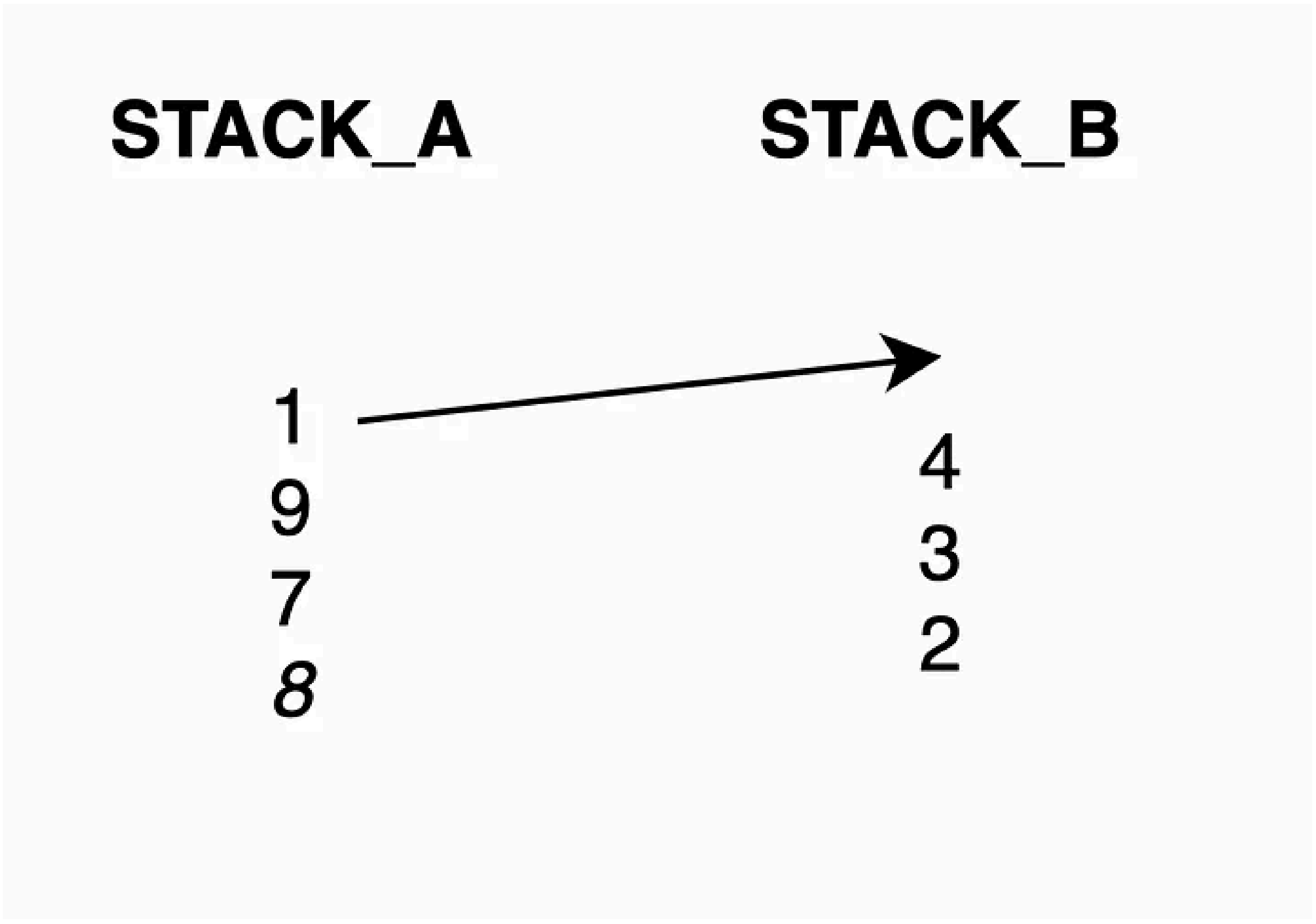
Let me start with mentioning some ground rules.

— I slowly push everything from STACK_A to STACK_B but in descending order. Why? Because after I push them back to STACK_A, they will be automatically sorted.

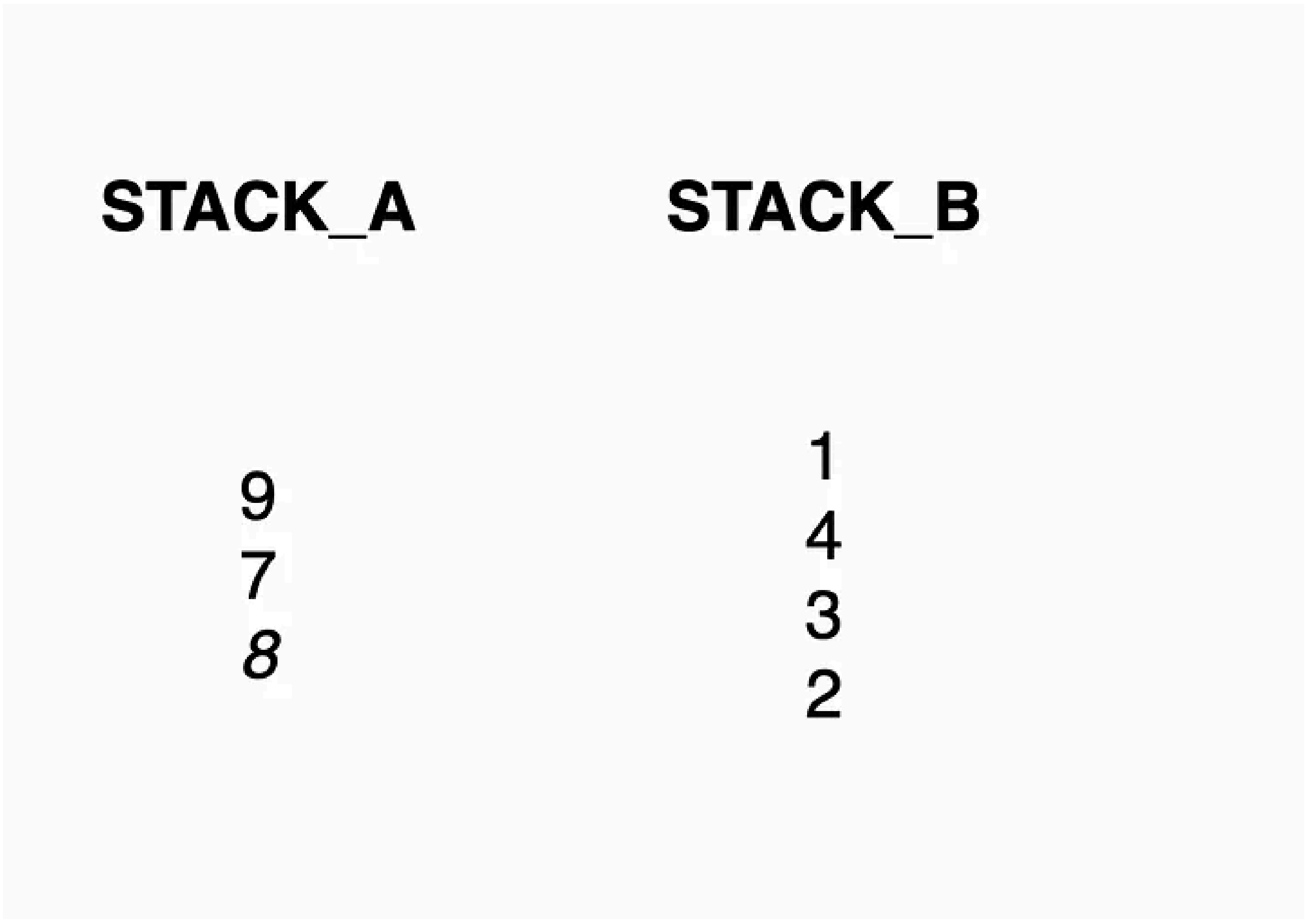
— Stacks are kind of circular linked list. It means, the last element of the stack is actually not the last element. It is actually an element before the first element.



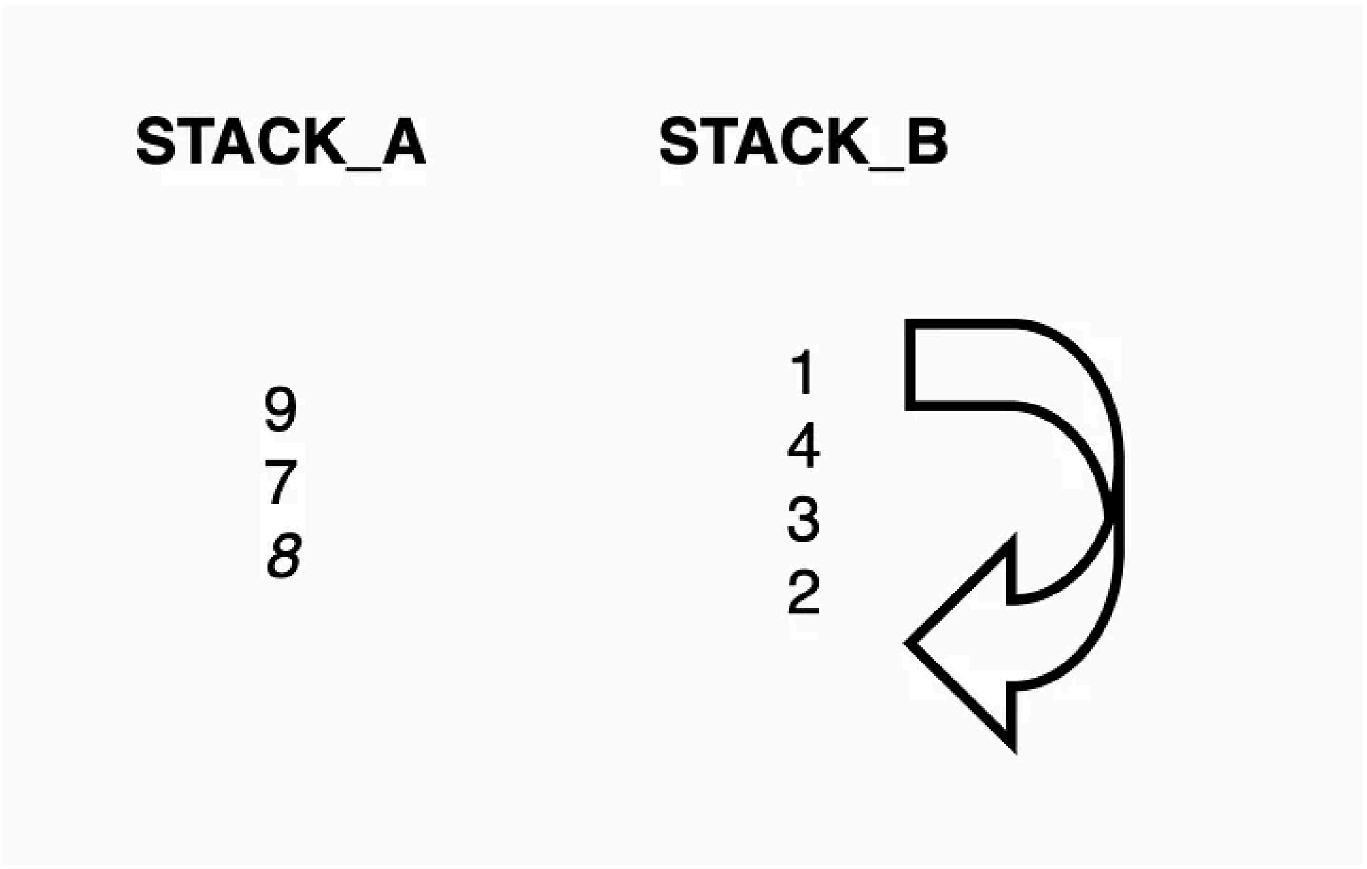
— If the number you push from STACK_A to STACK_B is going to be the new biggest or the smallest number, you should place it just above the old biggest number in the STACK_B.



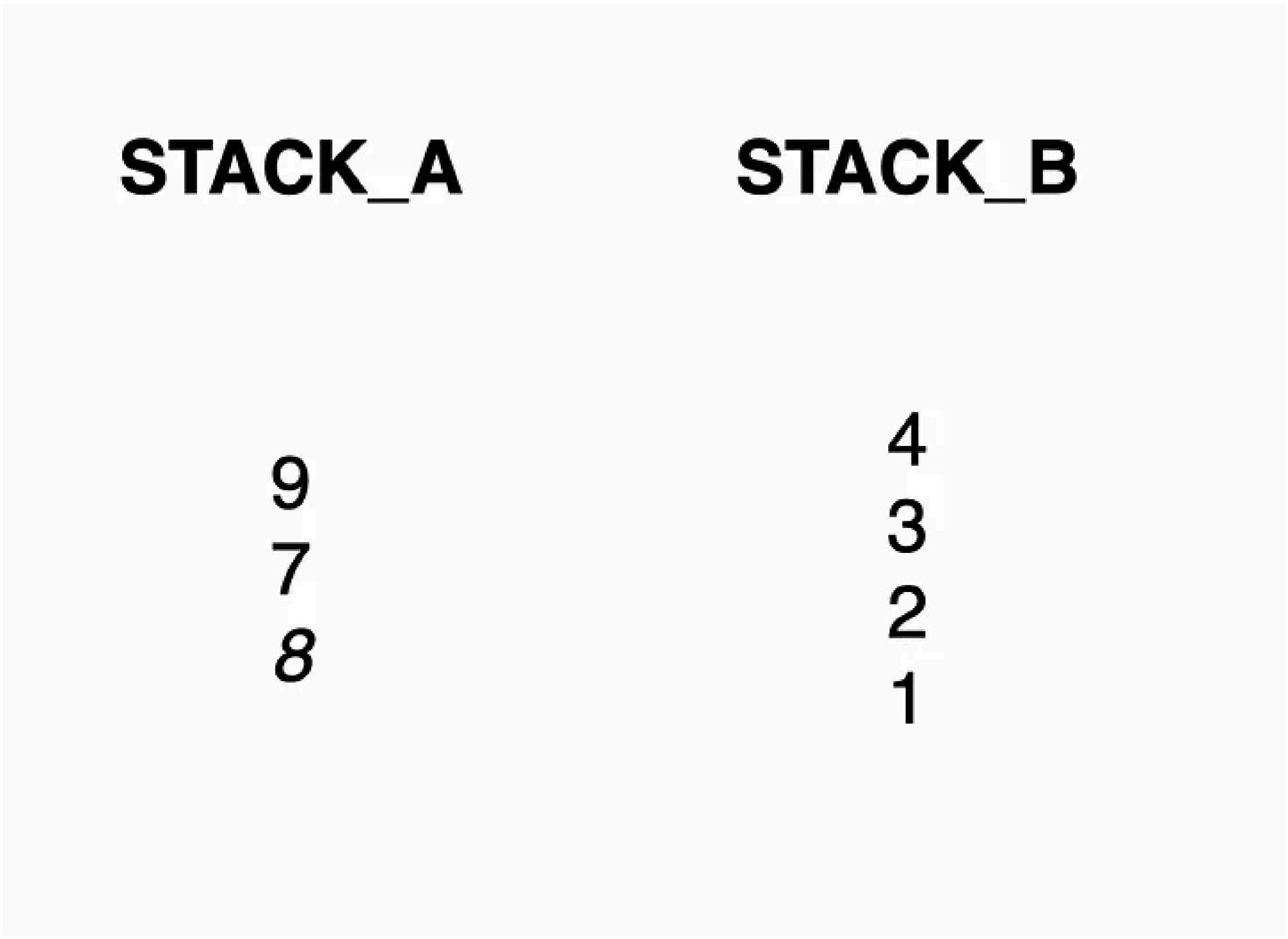
We will push number 1. Number 1 will be the new minimum number of the STACK_B. It means it should be placed on the top of the MAXIMUM number of the STACK_B.



Number 1 is pushed to the STACK_B. For you to see that it is the correct position, let me rotate the STACK_B.



Now I am rotating the STACK_B to bring it in the correct position.



Done. As you can see the correct position for the number 1 was the top of the MAXIMUM number. Rotating was redundant though. But I wanted to show you what I mean.

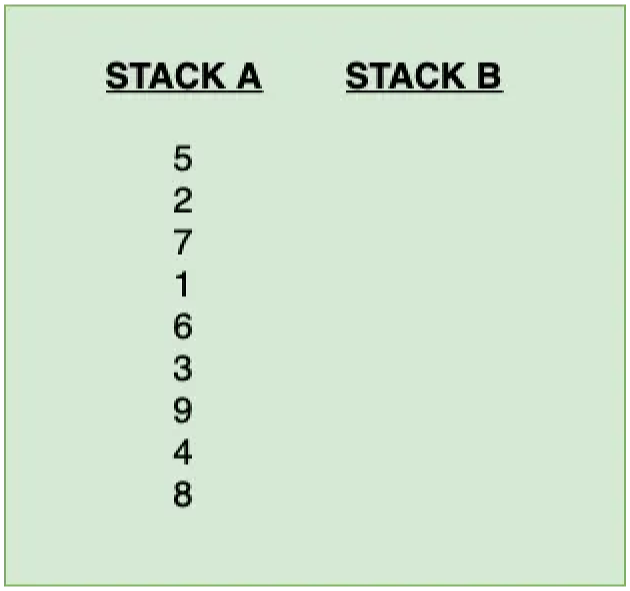
If we were going to push the number “9” instead of the number “1”, you can see that the correct position of the number “9” would again would be the right above the number “4” in STACK_B. The number “4” is the MAXIMUM number of the STACK_B. So, I think you can see now what I’m saying. If the the pushing number is neither MAXIMUM nor MINIMUM, it should be somewhere inside the STACK_B. We have to find it manually.

— The magic number is 3. If your stack size is three, you should understand one thing. When you have a stack in size of three, it means you don’t need to push something or overcomplicate things. Because if the stack size is three, you only need one operation to sort the stack. (Except two cases: which is a descending sorted stack such as 3–2–1. And a sequence where the greatest number and second greatest number switched places such as 1–3–2. Both scenarios would require two operations) So remember this magic number.

— We should be vigilant. So, let's imagine that we should rotate three times STACK_A and four times STACK_B. Basic math tells us, 4+3 equals to 7. It makes seven operations. This is why we will do a walk around solution. We will choose the least common number among these amount of rotations. So, instead of doing four times A rotate and three times B rotate, we can do three times rotating together. After that, if we rotate STACK_B one more time, we will get the desired result. But this time 3+1 will be 4 operations we did. Instead of doing 7 operations, we did 4 operations. Pretty efficient. This is why we will always calculate before rotation.

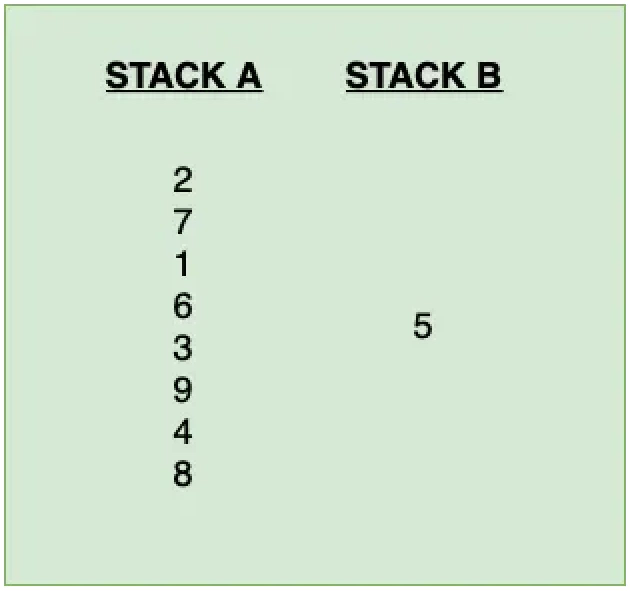
The Algorithm

Okay, I created a small stack with 10 members for you guys. And let us sort this stack with my algorithm.

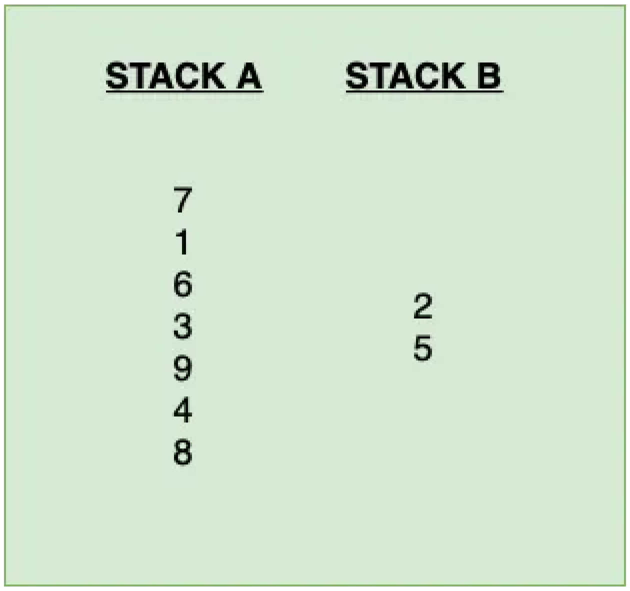


1. Push 2 number to STACK_A

As I mentioned previously, in order to determine a number's correct position in STACK_B, I need a maximum number in STACK_B. If the number I am pushing is neither MAXIMUM nor MINIMUM, the number has to be somewhere else inside the STACK_B. Hence, my algorithm starts with pushing two numbers from the top of the STACK_A to the STACK_B without checking anything.



pb



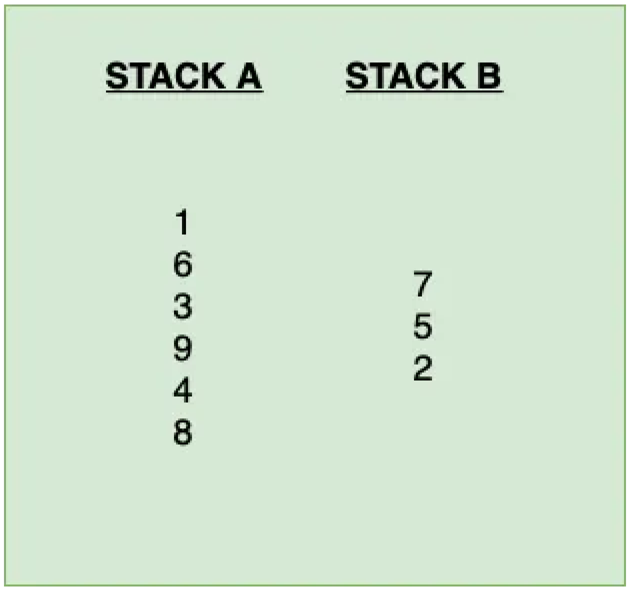
pb

2. Find the cheapest number

After that certain point, now we will count all the time. We will count and calculate for every number one by one before each push. We will calculate how many operations it would take to push the number into the correct position in the STACK_B. Let's start the calculations…

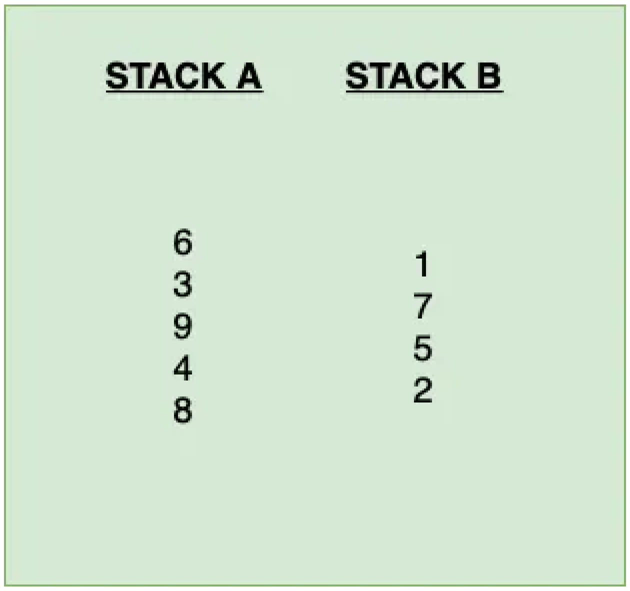
- **7** : We need to rotate STACK_B and then push it(Because 7 is going to be the biggest number in the STACK_B and this is why it should be placed above the current maximum number in the STACK_B which is 5. But number "5" is down below so this is why first we should rotate the STACK_B to bring the number "5" to the top. After I rotated, now I can push the number "7"). So it requires 2 operations. If we cannot find another number which requires less than 2 operations, we will push that number. Otherwise, we will push the number "7".
- **1**: We need to rotate STACK_B and STACK_A then push it. 1 time STACK_A rotation, 1 time STACK_B rotation, and then push the number to the STACK_B. It was "3" operations. If you remember the basic math I've explained previously, you can rotate two stacks at the same time. By this way instead of "3" operations, you could do "2" operations. But still it is not less than previous time. Still our champion is number "7".
- **6**: The number "6" will be the biggest number in STACK_B. So you can do the same logic as in the first step. You will see that it would take "3" operations to put the number "6" in correct position in STACK_B.
- **3:** This little friend needs "4" operations to put him in the correct position. Still it isn't cheaper.

No need to continue this. You can see that the cheapest number is "7". So, let us push him.



“7” is pushed to the STACK_B

Again start calculations one by one for each number in the STACK_A, and find the cheapest number. Well… The cheapest one is the number “1”. Because it is visible that it it requires only and only one operation. No need to check the rest of the stack. We already found the cheapest number. So, let’s quickly push him into STACK_B.

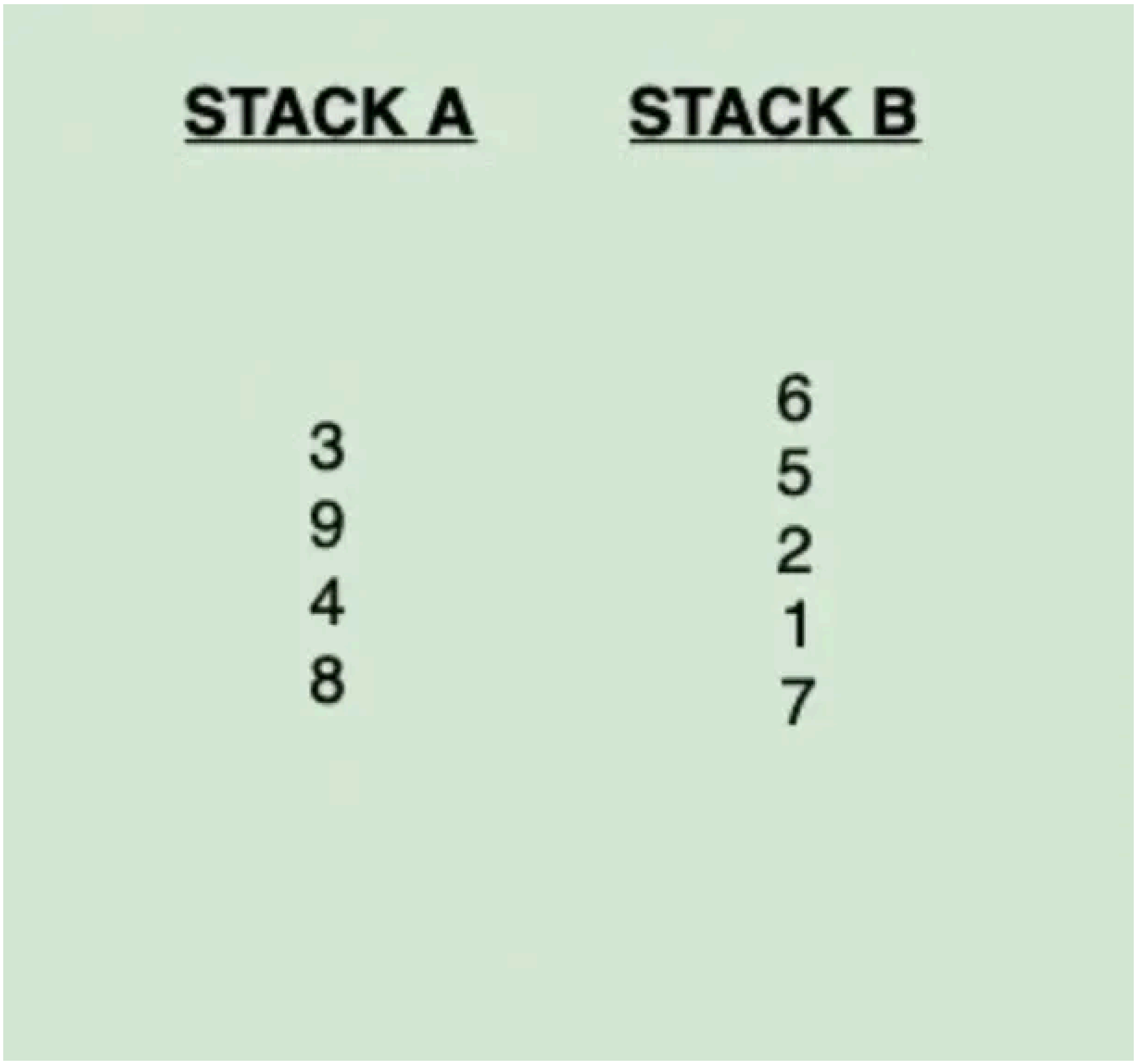


“1” is pushed to the STACK_B

Let’s start calculations again.

- **6:** It should be just above the number 5. It requires two times rotation of STACK_B and one push from A to B. Which means "3" operations are required.
- **3:** It should be just above the number "2". It requires one rotation of A, one reverse rotation of B and and one push from A to B. This would be "3" operations. The cheapest number is the number "6" for now.

You can continue to calculate each number till the end and realize that number **8** requires the same amount of operations as number **6**, but the way the algorithm works is that we only do operations if it's the cheapest, and we don't change it until a cheaper one is found. So, the first cheapest number we've found is six, there isn't anyone else cheaper. Let 's push him.



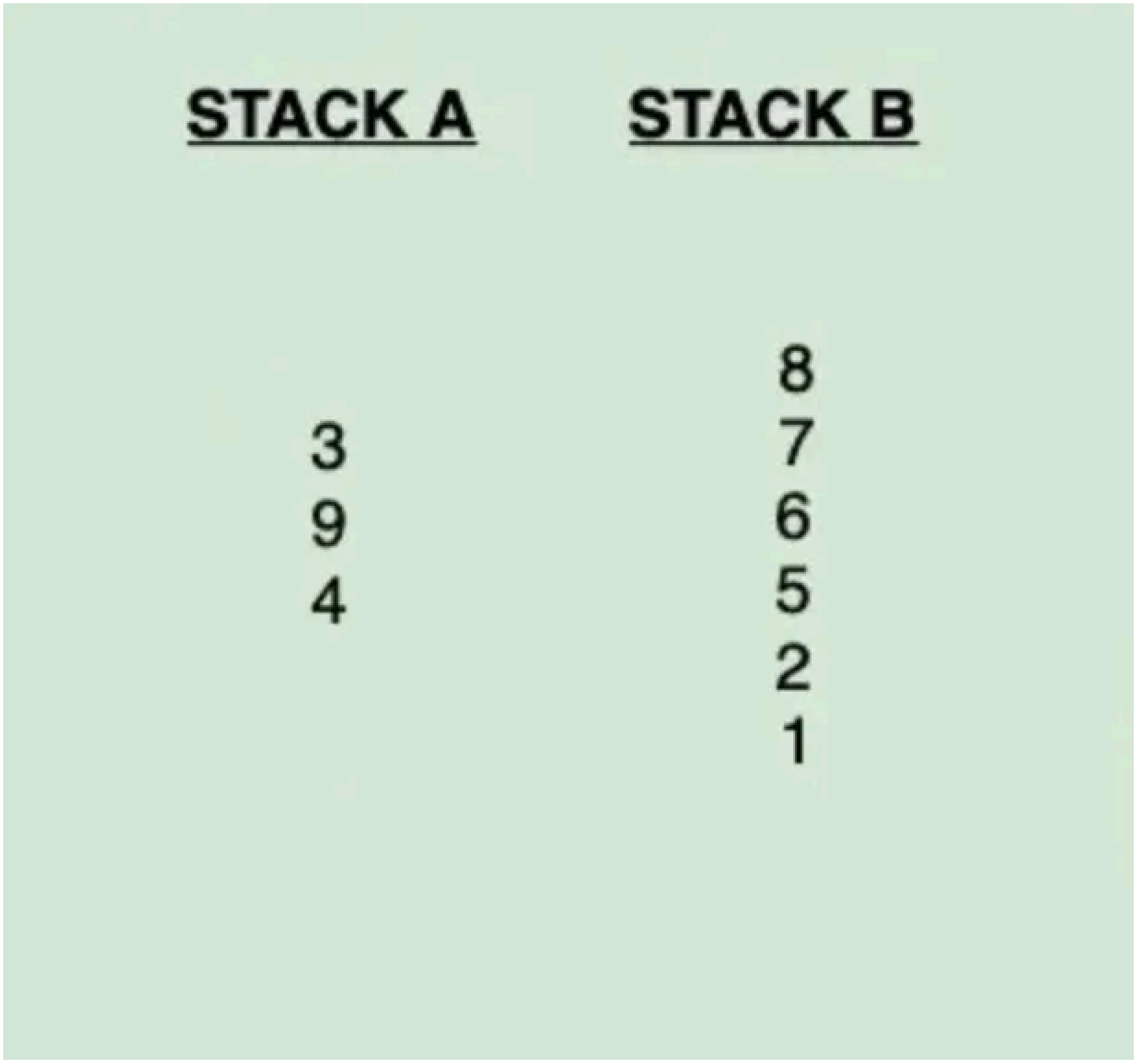
“6” is pushed to the STACK_B

Okay, one last time we will do our classic calculations. Yes, last time. Why? I will explain. Stay tuned. So, let’s dive into calculation:

- **3:** Two times STACK_B rotation and push operation. It would be three operations.
- **9:** One time STACK_B rotation, one time STACK_A rotation and push operation. It would be three operations.
- **4:** Two times STACK_B rotation, two times STACK_A rotation. It would be two simultaneous rotations and push. Three operations in total.

- **8:** One time STACK_B rotation, one time STACK_A rotation. It would be one simultaneous rotation and push. Two operations in total, we have a new winner!

Let ‘s push eight.

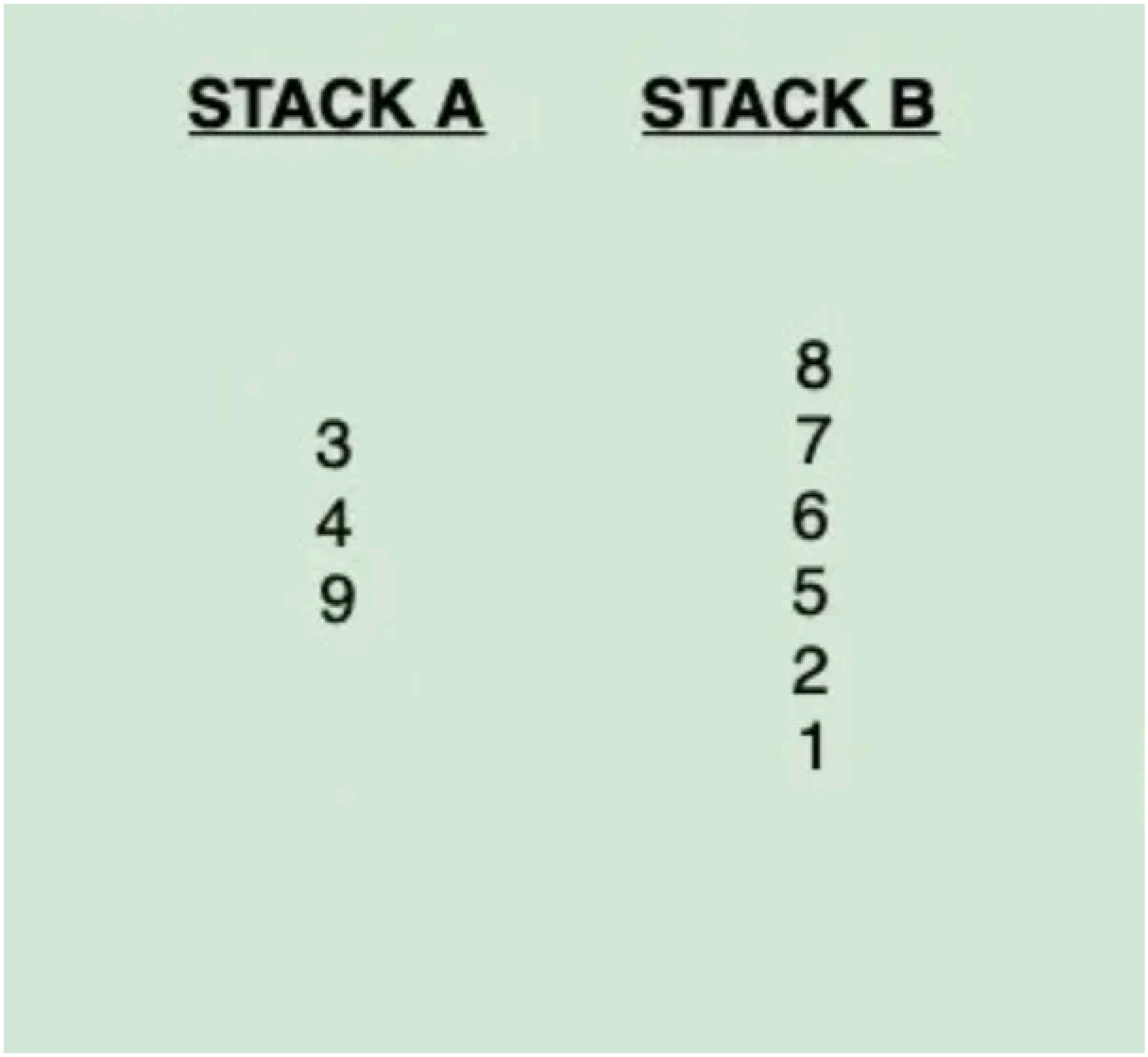


“8” is pushed to the STACK_B

3. Last three elements

Do you remember I’ve mentioned above the importance of three. Yup. This is the reason. This is why we stop at this point. We don’t need to push the numbers to the STACK_B anymore. Because we can sort these three numbers only with one operation. In the worst case, it would be only two operations. Okay, in our case this is the worst possible scenario that we could get. LOL.

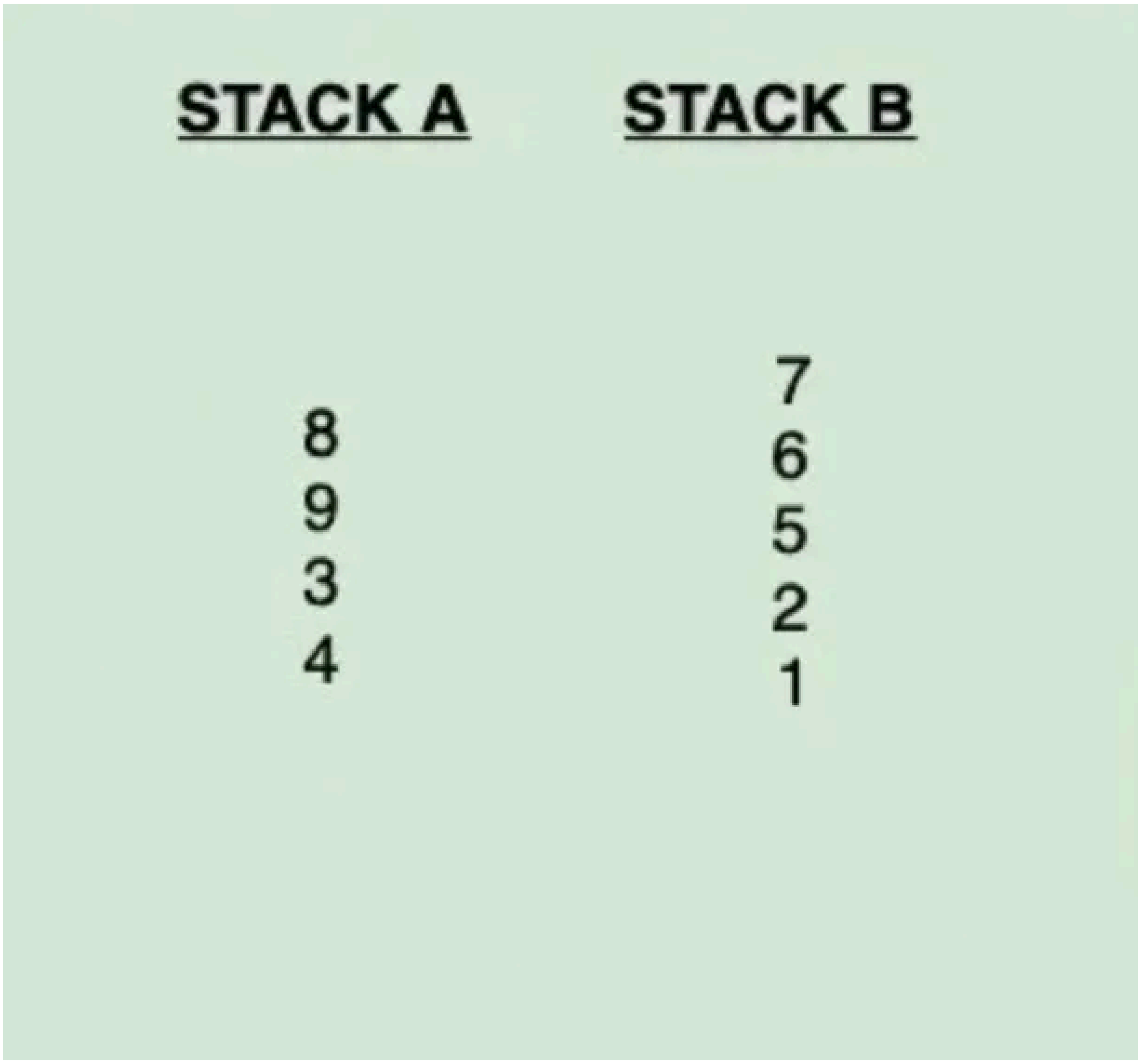
We have to do a swap and a rotation in STACK_A. After these two operations, STACK_A will be sorted. And it will look like this:



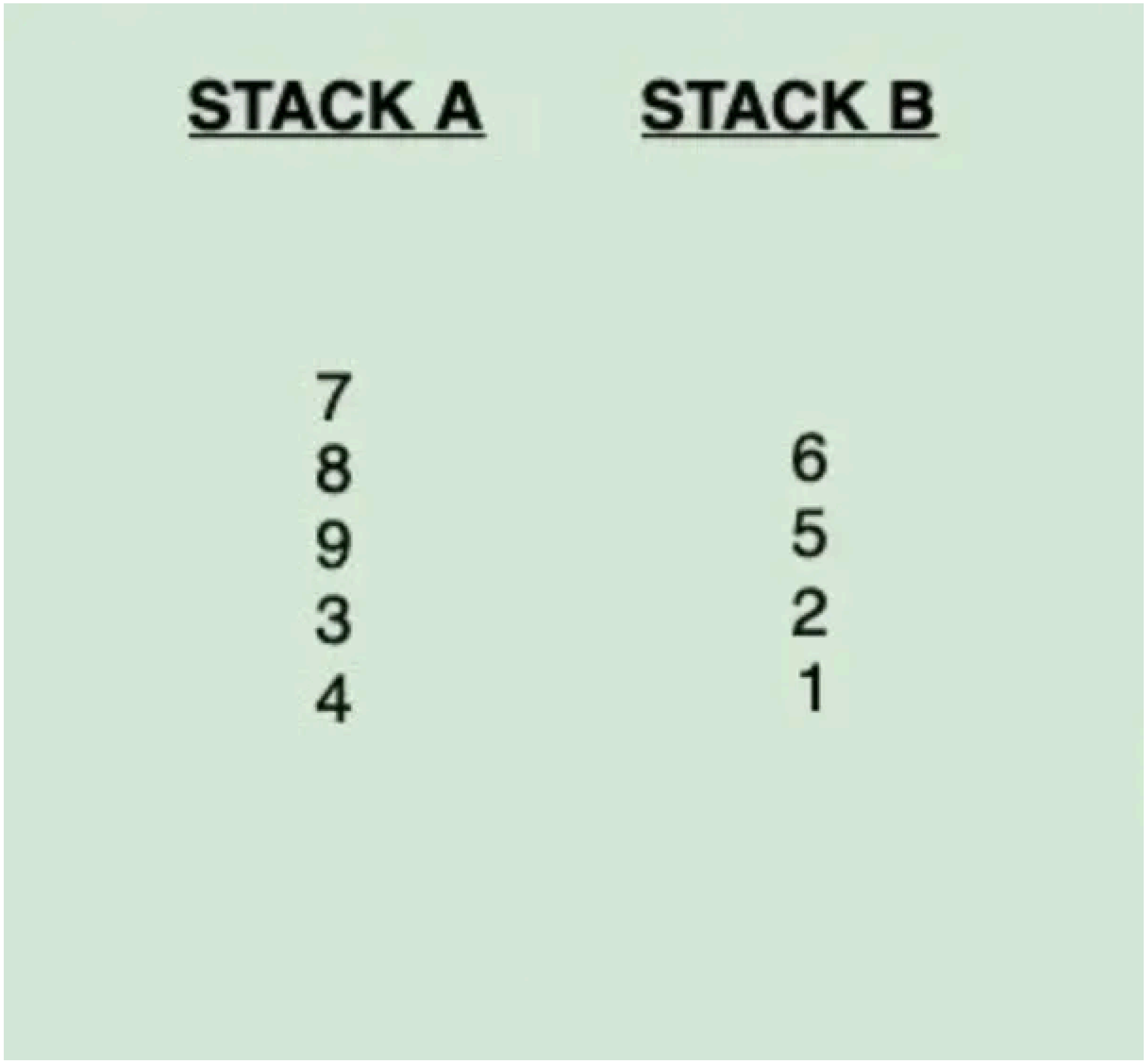
4. Time to push back to STACK_A

In this stage, we push everything from STACK_A to STACK_B. With one simple exception. Every time before we push a number, we check if it is being pushed to the correct position. If it is not, we rotate STACK_A until the correct position comes up. Let’s see how it is being executed.

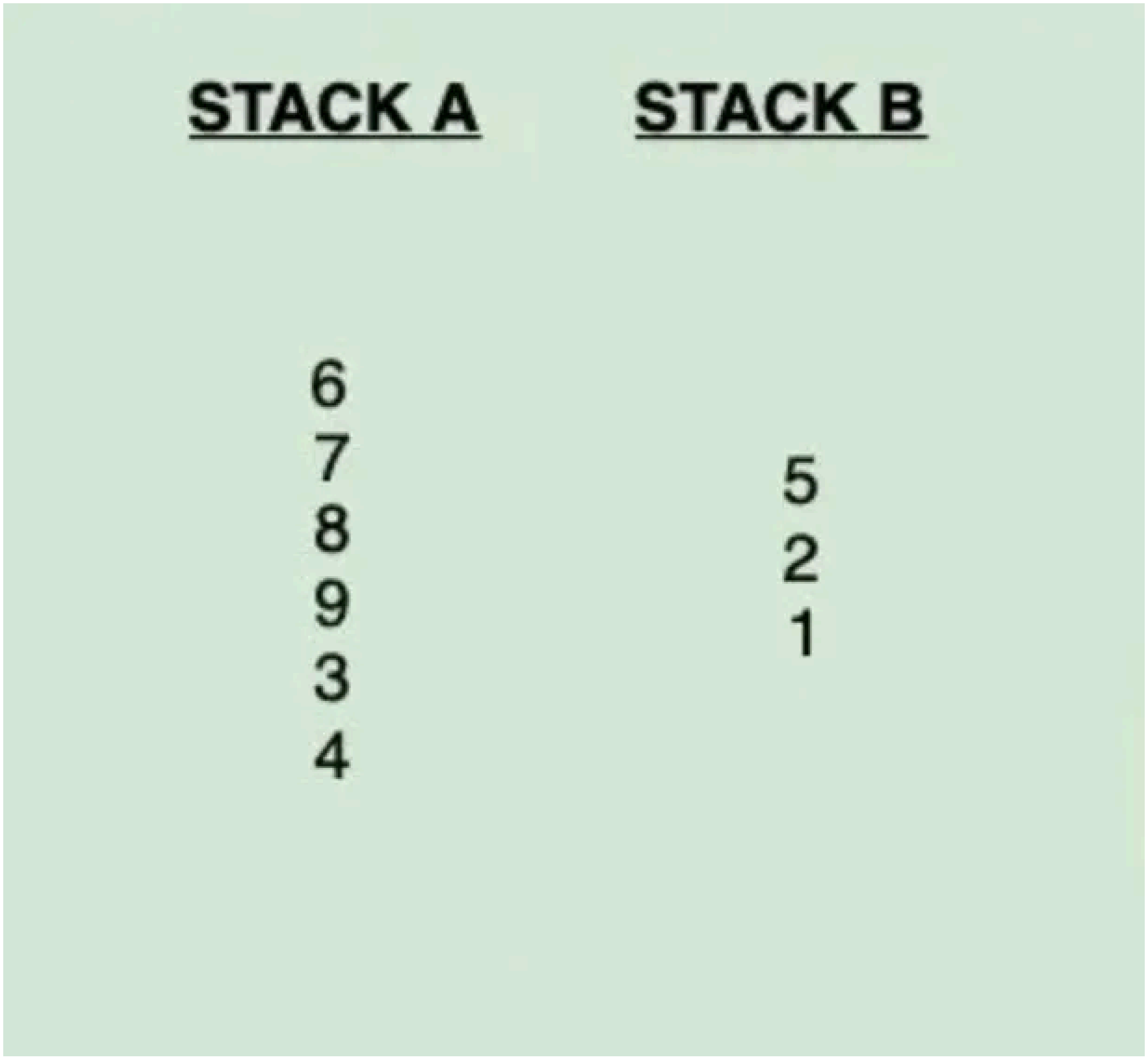
To put the number eight of the STACK_B in the correct position in STACK_A, it has to be above the nine of the STACK_A. So we need to do a reverse rotation in STACK_A to get nine to the top, then we just push eight from STACK_B.



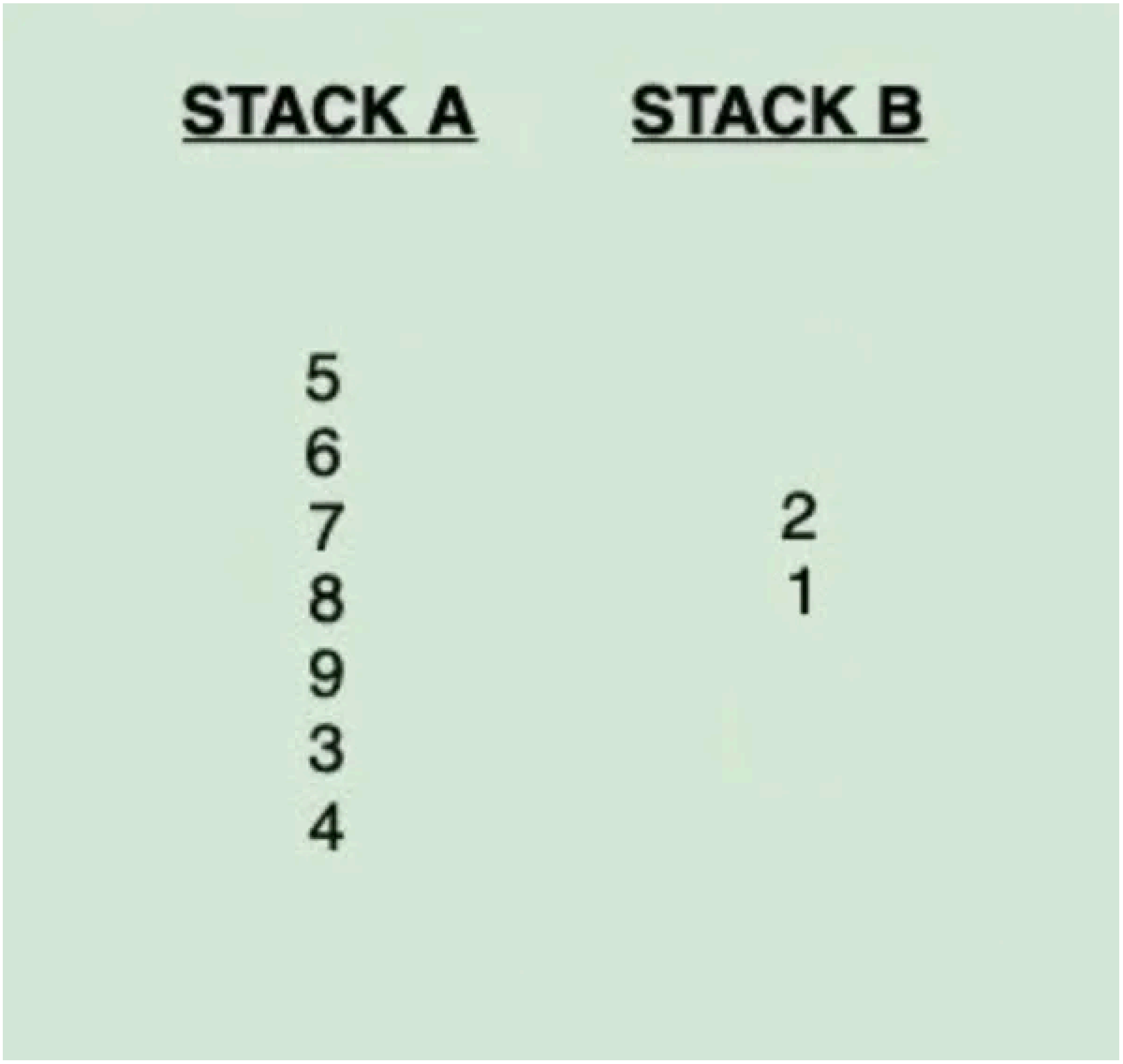
Reverse rotate STACK_A and “8” is pushed to the STACK_A



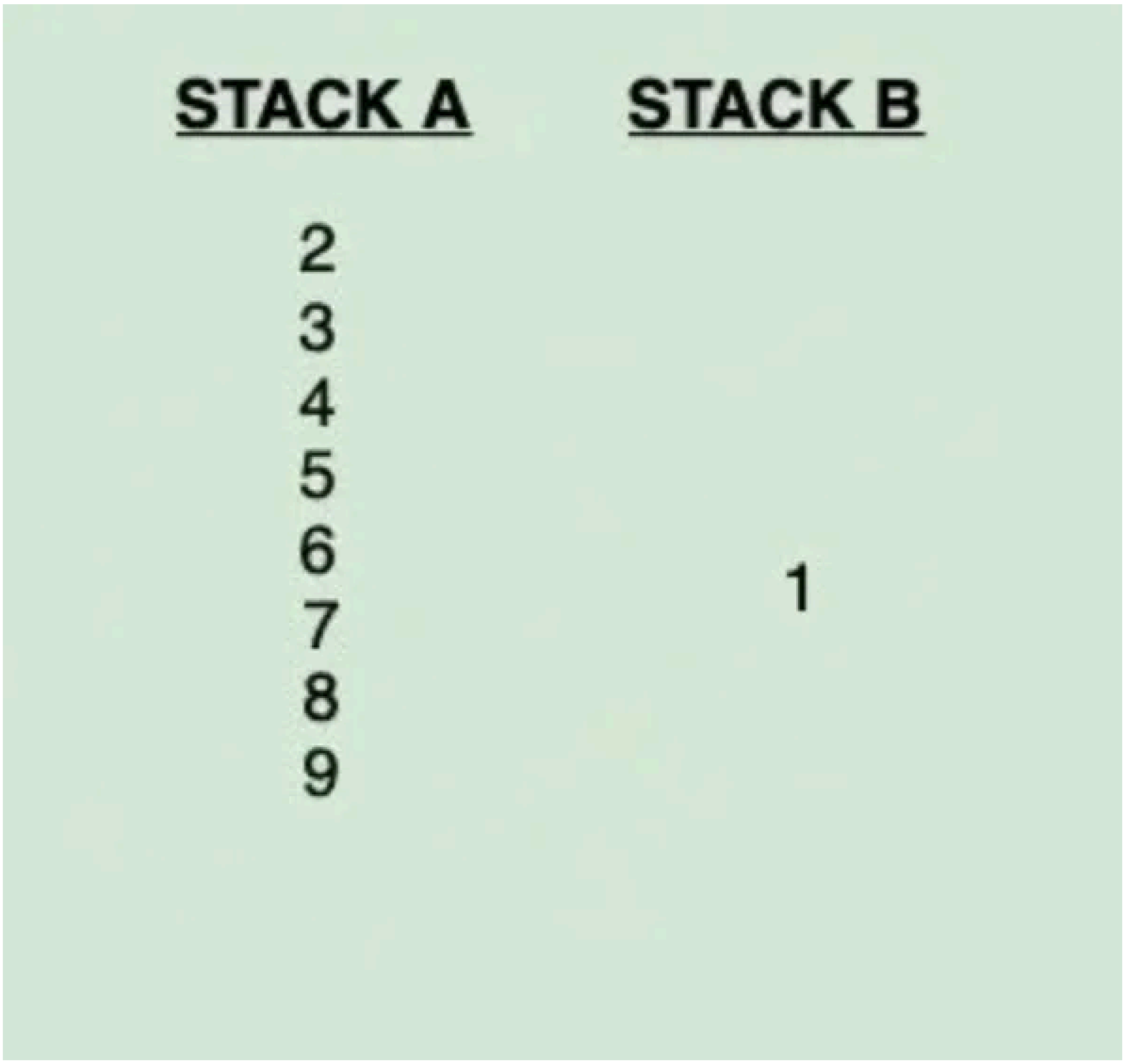
“7” is pushed to the STACK_A



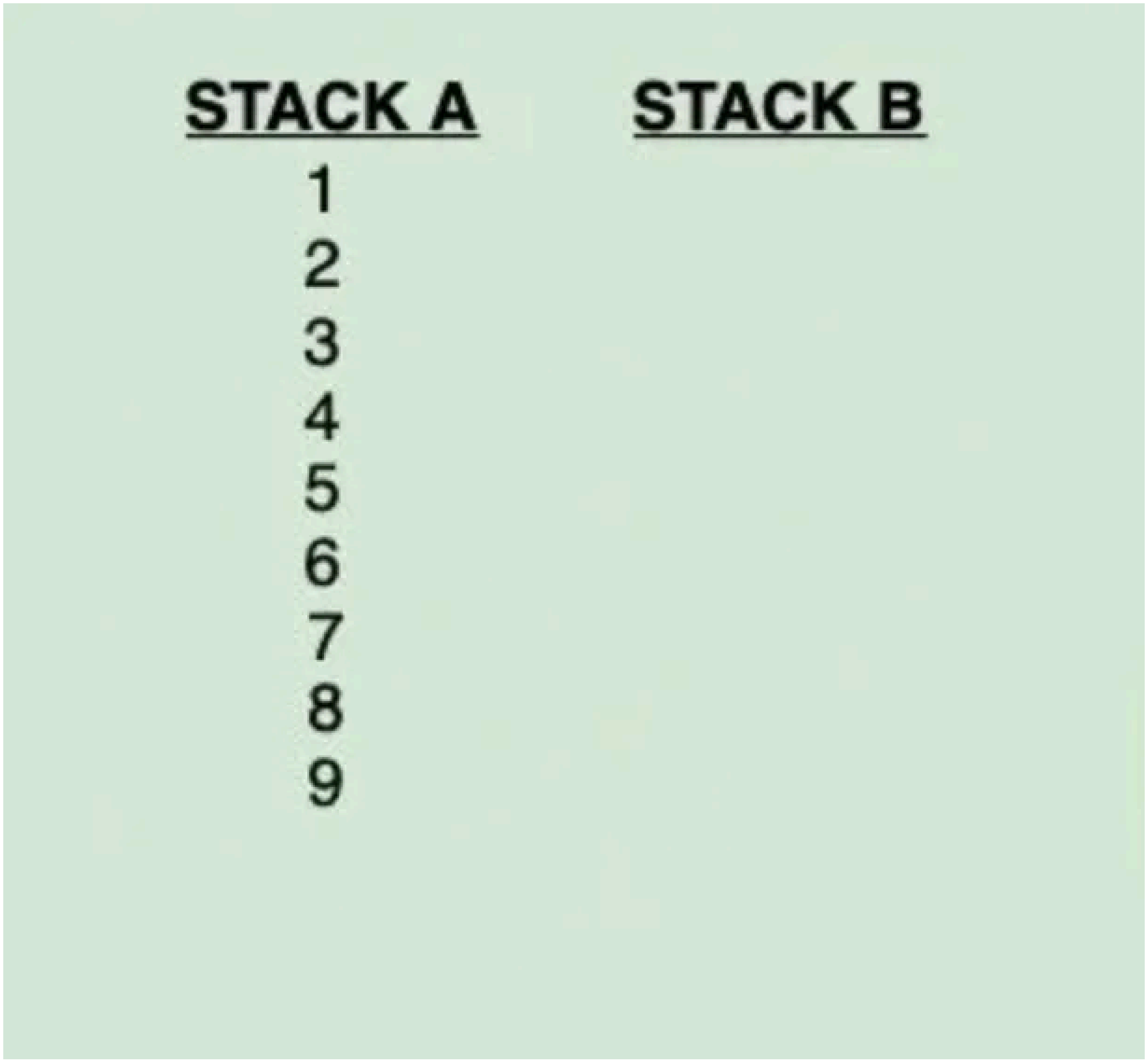
“6” is pushed to the STACK_A



“5” is pushed to the STACK_A



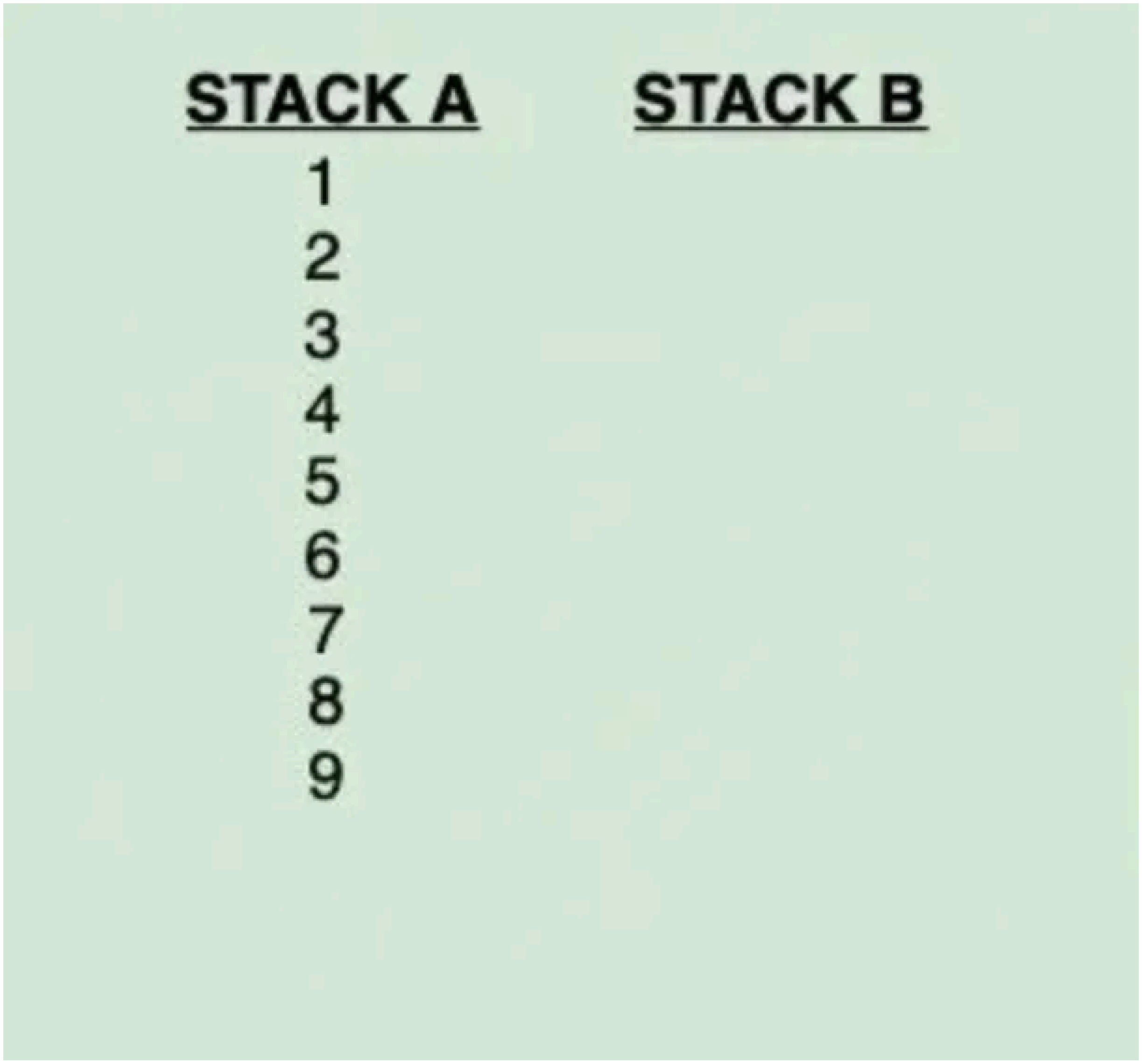
Reverse rotate STACK_A twice, to bring “3” to the top and push “2” to the STACK_A



“1” is pushed to the STACK_A

5. Final arrangement

We are almost done. In the final step, we do one simple thing. We should bring the minimum number of the stack to the top of the stack. It means the number “1” should be placed on the top of the STACK_A. We were lucky this time, our minimum number is already at the top, but this will not always happen. After making sure your minimum is at the top you will be sure your stack is sorted.



And here we are. You sorted the stack. If you understand the logic behind and code this algorithm, you will get a full score from the project which is 125/125. I hope you are satisfied and it is clear enough. I was hoping to answer questions such as “How to do the push swap project?” or “Which algorithm is better for push swap?”. I think there are plenty of answers to those questions. And I only answered based on my experience. In the project of push_swap, everybody has the same purpose: Finding a highly efficient sorting algorithm, or finding an algorithm to sort with the least amount of moves with two stacks. I gave you this one because I think it is simple and intuitive :).

As a 42 student with a peer-to-peer learning spirit owner, I learn a lot from the community and I hope to give it back. I hope you will do the same after you get from me. Good luck with the curriculum :)

Acknowledging Your Contributions

I penned this article shortly after I had submitted my push_swap project while heading to Poland while waiting for my train in Dortmund. The clock had struck 3 AM, and I found myself munching on some fries at McDonald's as I typed away. In my tired state, I made quite a few clumsy mistakes. However, I'd like to extend a special thanks to **Beatriz**. Also, take a moment to check her *GitHub profile*. She tirelessly worked to rectify numerous errors and even revamped entire paragraphs to enhance clarity for our readers. Her invaluable contributions made this article significantly more accessible and easy to read. Right on — this is the magic of open source!

Beatriz Dile - Medium

Read writing from Beatriz Dile on Medium. Github: https://github.com/beatrizdile. Every day, Beatriz Dile and...

medium.com

Push Swap   42 Ecole   Sorting Algorithm   Sorting   Algorithms

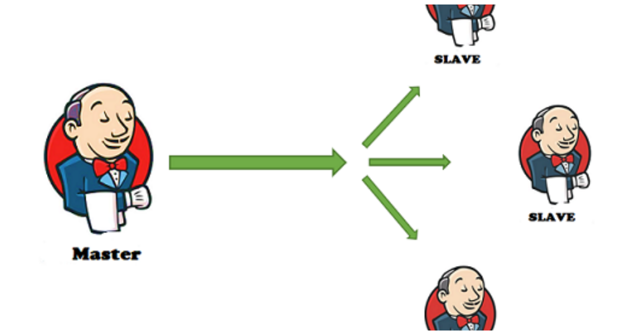
Written by A. Yigit Ogun

Follow

248 Followers

42 Heilbronn ✦ AWS Community Builder ✦ DevOps ✦ Cloud ✦ C/C++ ✦ GNU/Linux ✦ https://linktr.ee/ayogun

More from A. Yigit Ogun



A. Yigit Ogun

Master-Slave Database Architecture in a nutshell

The master-slave architecture has been around for a long time. Actually, this is a fairl...

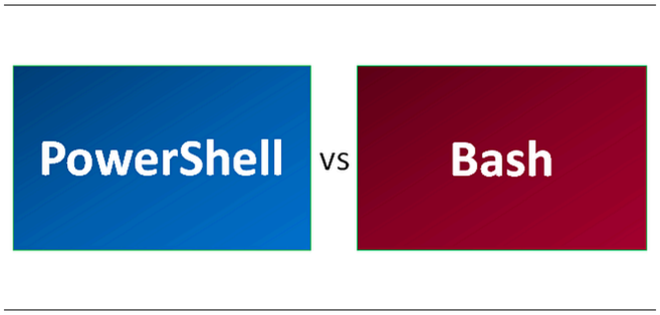
May 1, 2022 153 1

A. Yigit Ogun

What is Makefile and make? How do we use it?

If you have previously developed programs with C/C++ programming languages,...

May 1, 2022 108 1



A. Yigit Ogun

Shell vs BASH vs PowerShell vs CMD

Today I was talking with one of my friend about a shell script. I noticed that she doesn'...

May 1, 2022 151

A. Yigit Ogun

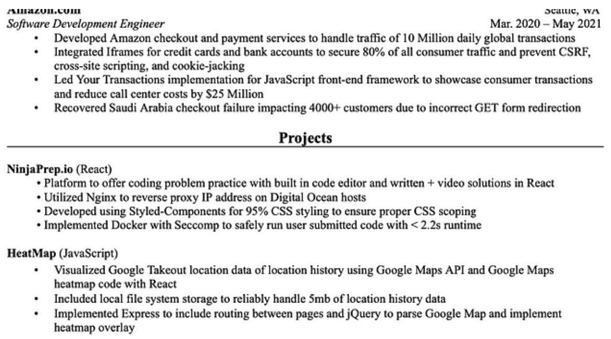
Makefile Basics: Beginner-Intermediate

First of all, I can't recommend enough reading the GNU user manual for make enough, it...

May 1, 2022 67

See all from A. Yigit Ogun

Recommended from Medium



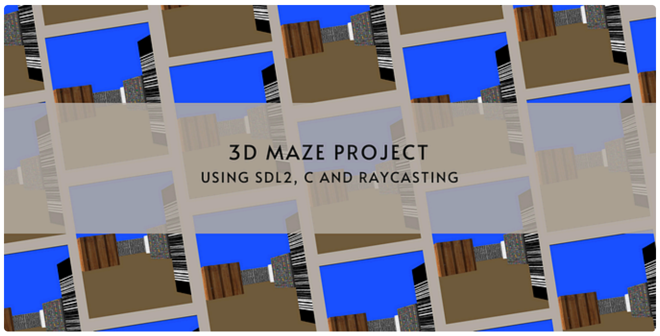
Amazon.com Seattle, WA
Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay



Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

Jun 1 17.4K 274

Kamogelo Ellen Kganakga

3D Maze Project

Introduction: Creating a 3D maze game from scratch is no small feat. It requires a blend of...

May 9

Lists

Practical Guides to Machine Learning

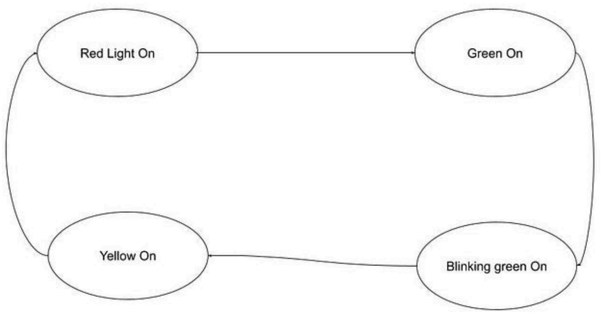
10 stories · 1746 saves

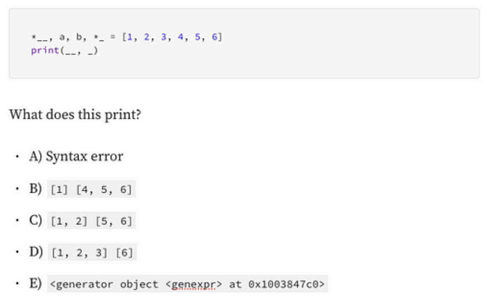
General Coding Knowledge

20 stories · 1472 saves

Staff Picks

713 stories · 1222 saves





```
*__, a, b, *_ = [1, 2, 3, 4, 5, 6]
print(__, _)
```

What does this print?

- A) Syntax error
- B) `[1] [4, 5, 6]`
- C) `[1, 2] [5, 6]`
- D) `[1, 2, 3] [6]`
- E) `<generator object <genexpr> at 0x1003847c0>`

Gealleh

Implementing a Hierarchical Finite State Machine in C++

In software engineering, finite state machines (FSMs) are a powerful technique for modelin...

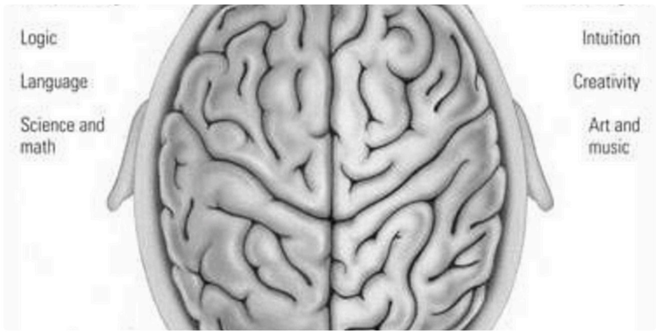
May 3 33 1

Liu Zuo Lin

You're Decent At Python If You Can Answer These 7 Questions...

# No cheating pls!!

Mar 6 6.8K 30



James Koh, PhD in Towards Data Science

Artificial Bee Colony—How it differs from PSO

Intuition and code implementation for ABC, and exploring where it outperforms Particle...

Dec 18, 2023 524 3

Sufyan Maan, M.Eng

What Happens When You Start Reading Every Day

Think before you speak. Read before you think.—Fran Lebowitz

Mar 12 29K 700

See more recommendations