push_swap

The project that will make you sort numbers

Push_swap is a project that teaches you about sorting algorithms and how to optimize them. The project involves sorting a stack of integers using a limited set of operations (push, swap and rotate) and minimizing the number of moves.

At the beginning, you must have one stack, called `stack a` with all your numbers placed next to each other. With the help of the operations below, at the very end, all the numbers in your `stack a` will have to be sorted.

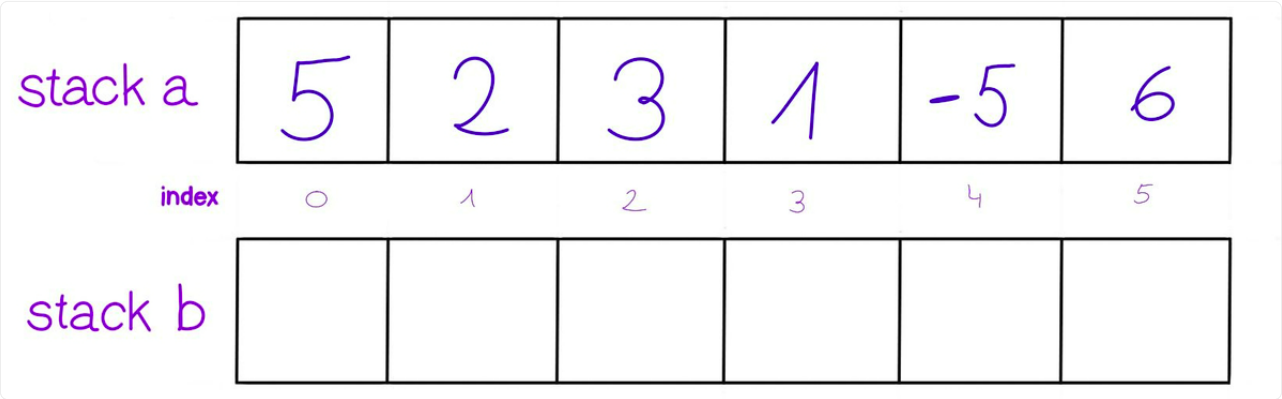
> ⚠ At the end, you don't need to return the sorted list, just the sequence of operations to sort the list.
>
> You can normalize the list if you need to

To do this, you can have a second empty stack, called the `stack b`, on which you can temporarily send elements. Here are the operations you can use (and that you will have to code):

- `sa (swap a)` : Swap the first 2 elements at the top of the stack a. Does nothing if there is only one or none.
- `sb (swap b)` : Swap the first 2 elements at the top of the stack b. Does nothing if there is only one or none.
- `ss` : sa and sb at the same time.
- `pa (push a)` : Takes the first element on top of b and puts it on a. Does nothing if b is empty.
- `pb (push b)` : Takes the first element on top of a and puts it on b. Does nothing if a is empty.
- `ra (rotate a)` : Shifts all the elements of the stack a up by one position. The first element becomes the last.
- `rb (rotate b)` : Shifts all the elements of the stack b one position upwards. The first element becomes the last one.
- `rr` : ra and rb at the same time.
- `rra` (reverse rotate a): Shifts all elements of the stack down one position. the stack a. The last element becomes the first.
- `rrb` (reverse rotate b): Shifts all the elements of the stack b one position downwards. the stack b. The last element becomes the first.
- `rrr` : rra and rrb at the same time.

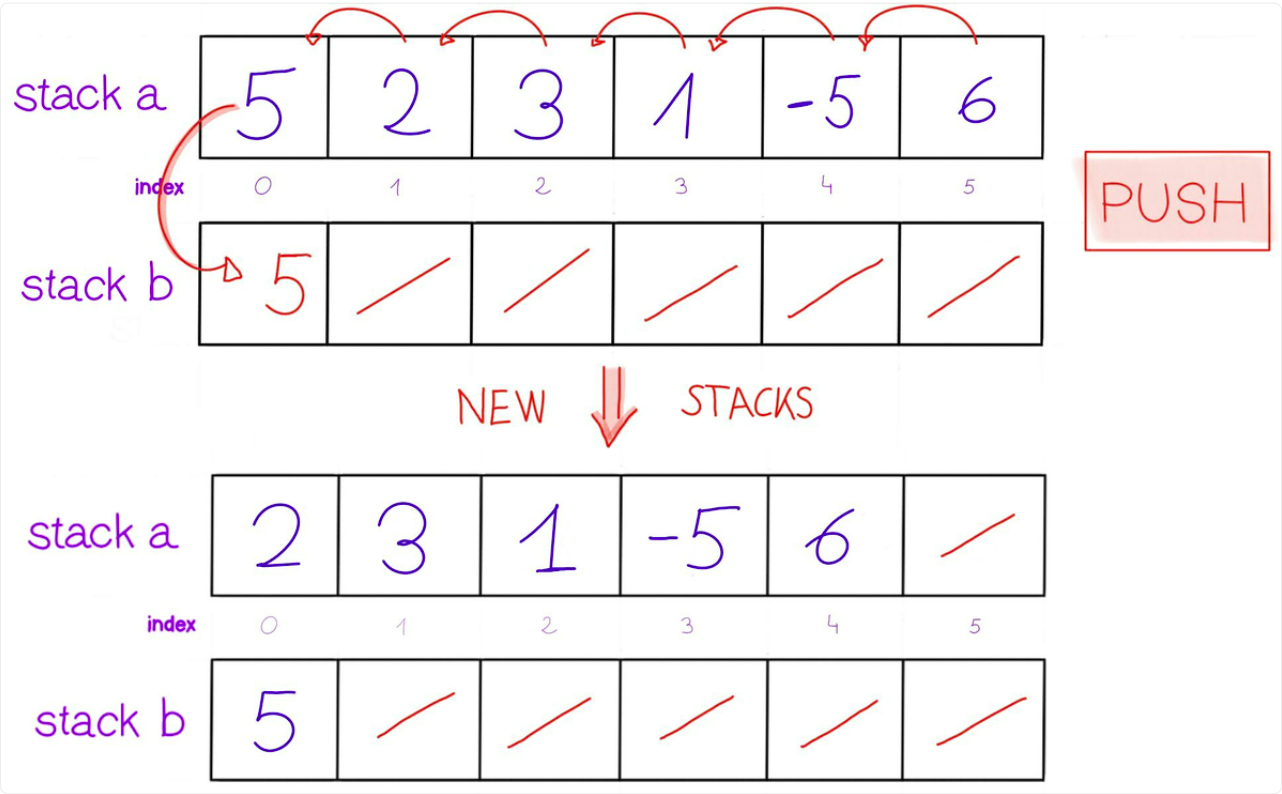
Examples

We have 6 random numbers, placed on a first stack, called stack A, as follows. Obviously, stack B is empty at the beggining. Stack B will be used temporarily throughout your program, but must be empty again at the end, with the list sorted into stack A.



initial stacks

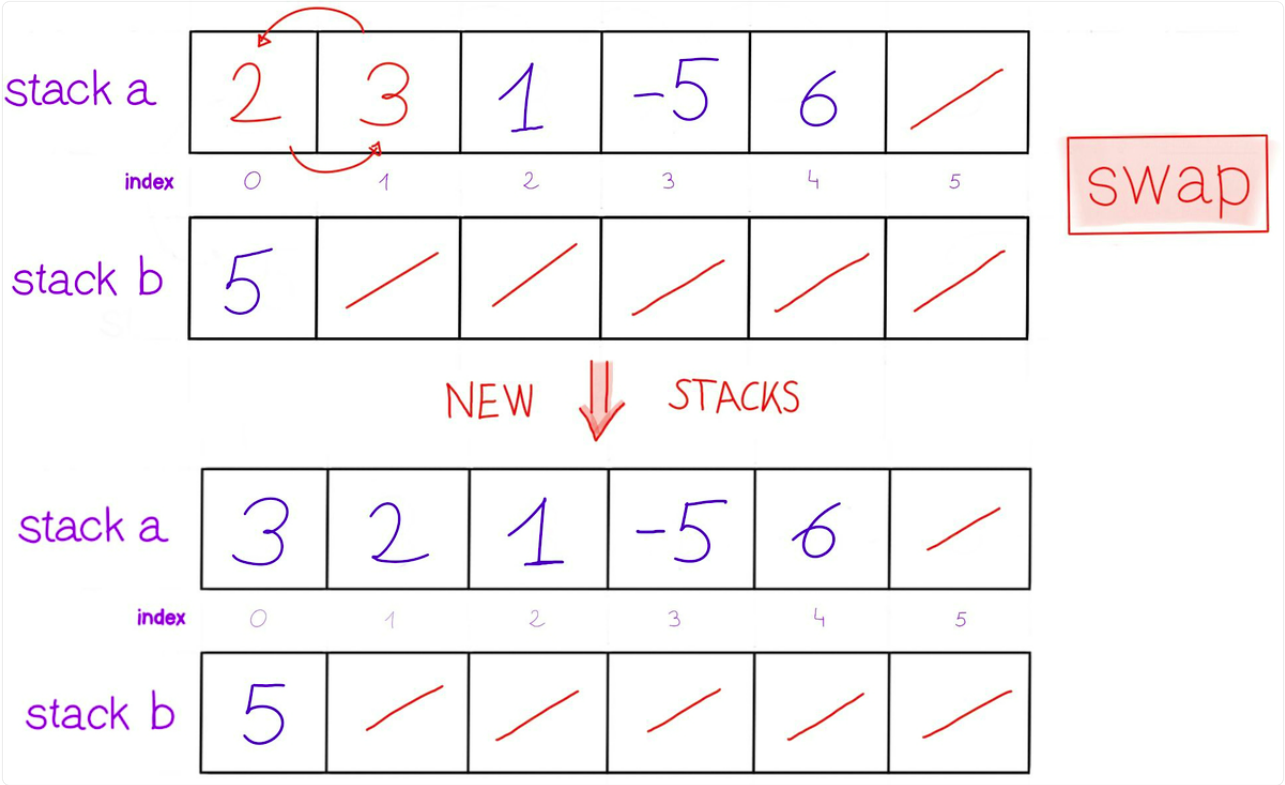
Let's say we want to move an element from stack A to stack B. This can be done with the "PUSH" operation.



PUSH example

The first element of stack A becomes the first element of stack B. Don't forget then that the second element of the A stack becomes the first, and so on. Here stack B was empty, but if there were numbers in it, the first element of stack B would become the second and so on. The process is the same when you want to send an element from list B to list A.

Let's continue with the modified list above. **We can also swap the first two numbers of a stack** by using the "SWAP" operation.

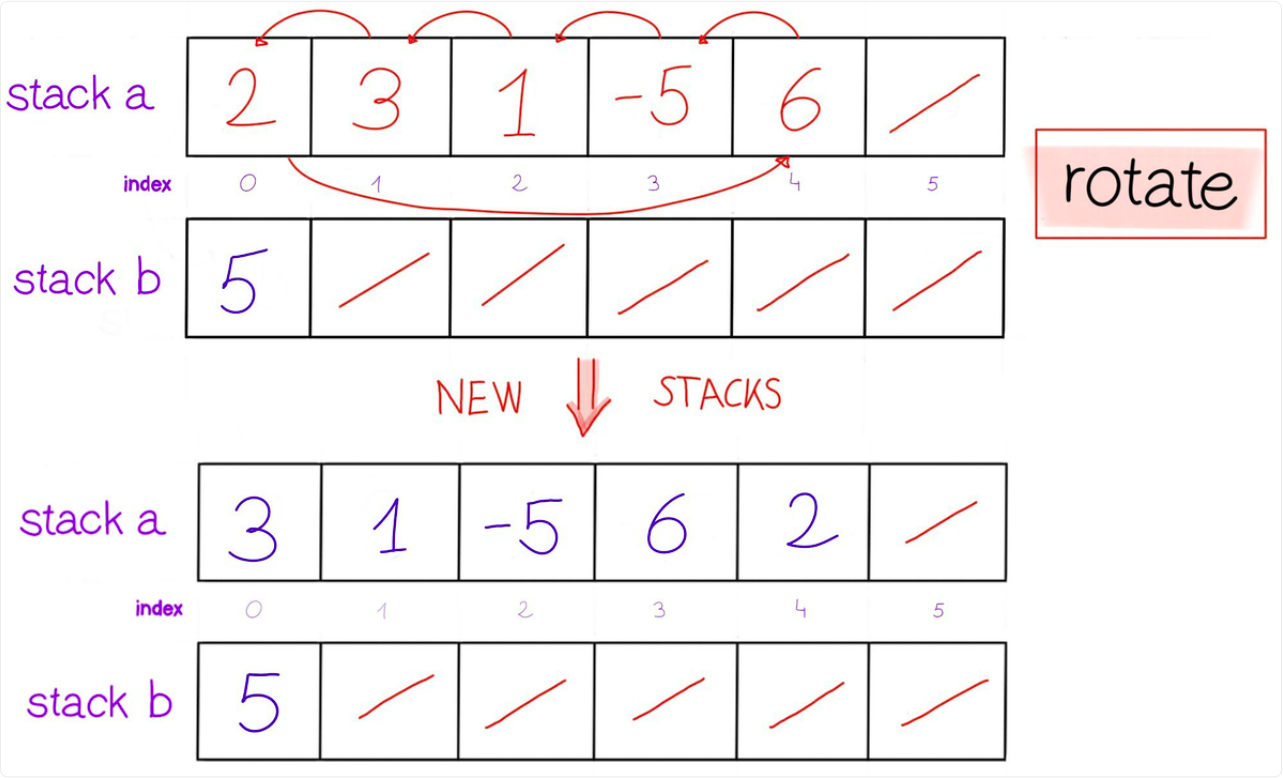


SWAP example

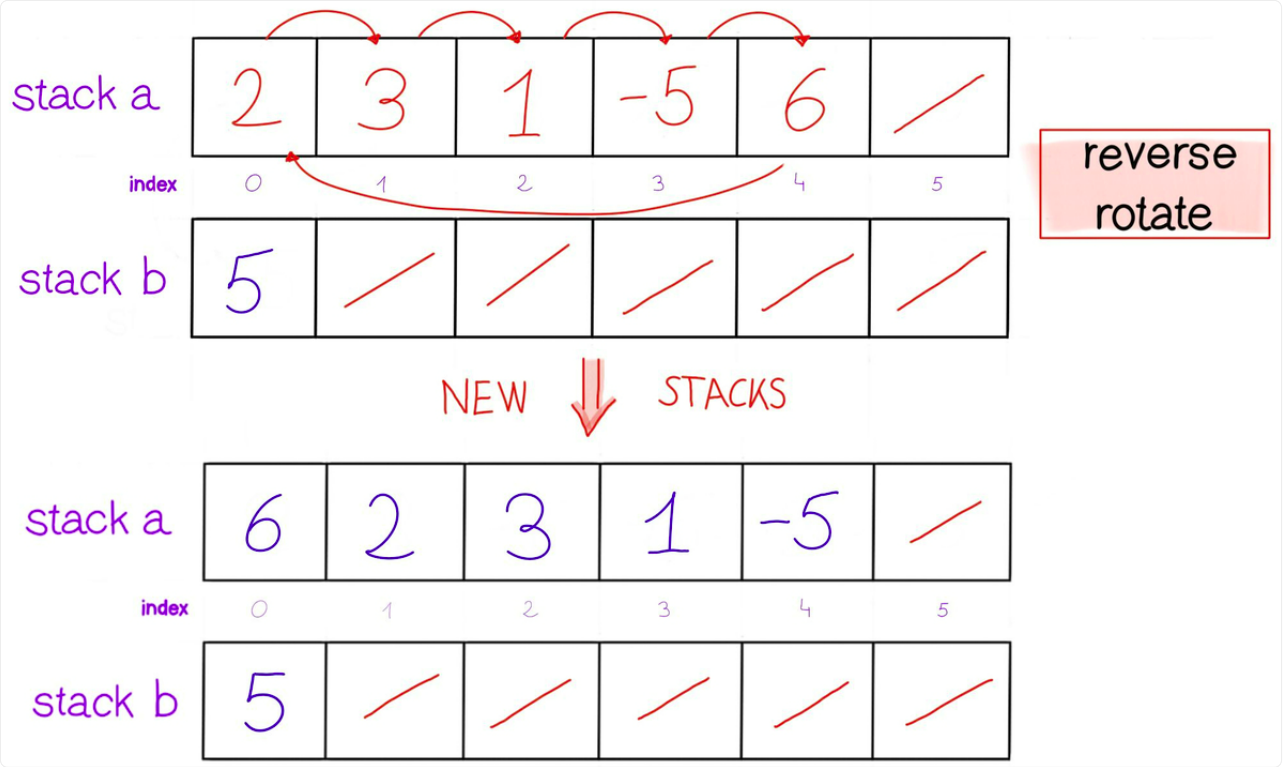
Here we can see that the only two numbers that have been impacted are the first two in the A stack. The logic remains the same when swapping two elements of the B stack. This one is easy.

One last type of operation can be done. **The rotation**. There are two types of rotations: rotation and reverse rotation. All the numbers in the stack are affected this time.

Normal rotation allows you to push all the numbers up and reverse rotation, all the numbers down. Here is a schematic example of both rotations:



ROTATE example



REVERSE ROTATE example

And that's it for the instructions. Of course you will need to combine all the instructions in order to write your own algorithm. I hope that these few examples will help you to better understand this project and make coding easier for you. Let's start now !

Previous
Building the thing

Next
Algorithms

Last updated 14 days ago