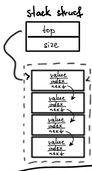
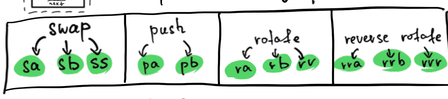


- parsing arguments
- validation
- constructing list based on arguments
- check for duplicates
- implementation of operations



- logic

---

when n==3 it is pure logic. We just compare three values calculating best moves and then execute these moves. b stack is not needed

---

when n==4 we try to reduce to 3. How? We find the smallest node by value push it to b then apply n==3 logic and at the end we push back.

---

when n==5 we find 2 smallest values push them to b, apply n==3 logic to a then push back.

---

when n>5 we apply radix sort for base 2 index nodes i.e. we treat indexes as binary numbers.

(1) we find "how bits we need to represent highest index?" Why? Because it determines how many rounds we should have.

(2) In the beginning of each round we should save the size of stack to avoid additional wrong iterations.

(3) we start iterating over each node and decide whether we keep the node or push. This decision is based on the bit where each round is from lowest to highest bit.

For stack a: we keep if 0 or push if 1	stack a: - keep: ra - push: pb, rb
For stack b: we keep if 1 or push if 0	stack b: - keep: rb - push: pa, ra

(4) after all rounds if there is a node in stack b we push back.

I tried to derive average time complexity. Why average? Because initial order of numbers can be different. So this is what I got:

$$\Theta(n) = \left(\left\lfloor \log_2(n-1) \right\rfloor + 1\right) \cdot \left(\frac{n}{2} \cdot 2 + \frac{n}{2} \cdot 1\right) + \frac{n}{2} \cdot 2$$

- $\left(\left\lfloor \log_2(n-1) \right\rfloor + 1\right)$ — number of rounds
- $\frac{n}{2} \cdot 2 + \frac{n}{2} \cdot 1$ — number of iterations in single round
- $\frac{n}{2} \cdot 2$ — push back

Example * keep == k , * push == p

- arguments:
  - values: -23, 123, 8, 12233, -5, 32, 5667, 98
  - indexes: 0 5 2 7 1 3 6 4

stack a		stack b
0 000 k		is empty
5 101 p		
2 010 k		
7 111 p		
1 001 p		
3 011 p		
6 110 k		
4 100 k		

base 10 ↑ base 2 ↑

after round 1 ⇒

stack a	stack b
000 k	101 p
010 p	111 k
110 p	001 p
100 k	011 k

⇒

after round 2 ⇒

stack a	stack b
000 k	010 p
100 p	110 k
101 p	111 k
001 k	011 p

⇒

after round 3 ⇒

stack a	stack b
000	100
001	101
010	110
011	111

⇒

since nodes are left in b we push them back ⇒

stack a		stack b
000	0	empty
001	1	
010	2	
011	3	
100	4	
101	5	
110	6	
111	7	

stack is sorted