D- 17/07
2021

Leetcode

1st class

Array - মেমোরির স্পেসে - দুইটির মধ্যে থাকে Data গুলো
1) HDD (stores)
↳ Data sequentially থাকে 2) RAM (run)

memory Address (আছে)

0	1	2	3		

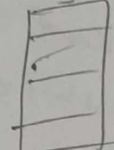
python - list a = [ ] empty list
↳ run
↳ Ram থেকে empty জায়গা নেয়

append → add data
[5] → data contain
a. append (5)
a. append (2)
value [5 | 2]
index → start
↳ কোথা থেকে শুরু (০ থেকে)

5	6	1	2	3
0	1	2	3	4

a. pop ( ) - delete → Last element

complexity
Big O - notation
↳ mainly (worst case)



O (size of arr/list)) ↑ search

Stack (Lifo)

index vs memory address
↳ language base

1	2	3
0	1	2

offset
address + index

length → ~~a. length~~ len(a)

C → array size
python → no size (init) Dynamic array

index access

a[index] (opn) search
time compl. (O(n))
complexity - memory "

O(1) → constant
↳ একটা opn

constant

a = [6, 5, 4, 3, 10]

~~Search~~ Search

for loop

a = [6, 5, 4, 3, 10]

target = 10

for element in a:
    print(element)

indexwise

a
start = 0
end = len(a)
while start < end:
    if a[start] == target
        print("found")
    start += 1

start	end	target	a[start]
0	5	10	0 · 0
0	5	10	6 a[1]
1	5	10	5 a[2]
2	5	10	4 a[2]
3	5	10	3 a[3]
		10	10 a[4]
4			

reverse:

a = [1, 2, 3, 4]

new array

res = [] → mem O(1)
start = len(a) - 1
end = 0
while start >= 0:
    result.append(a[start])
    start -= 1
print(result)

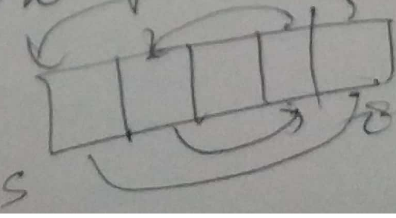
res	start
[ ]	3
[4]	2
[4, 3]	1
[4, 3, 2]	0
[4, 3, 2, 1]	-1

Hadoop

O(n - 2)
O(C * n)
O(C + n)
} → O(n)

Q. Array reverse.
M → O(n) → O(1)
T → O(n) → O(n)

in place swap $O(\frac{n}{2})$



2. Two pointer approach
→ Algorithm

H.W
Le → 1
Time complexity
space ~~ln~~
Github e push করো

a = [1, 2, 3, 4]

reverse in a: O(n)

O(1)
s = 0
e = len(a) - 1 (O1)

while s < e:
    temp = a[s]
    a[s] = a[e]
    a[e] = temp
    s += 1
    e -= 1

O(n/2) (beside the loop body)

O(1) space

O(1) return a

T: O(1) + O(1) + O(n/2) + O(1) = O(n/2) = On

Summary:

1. Array → list python
   - append, pop, length - len
   - index, element access, for, while loop
2. memory address, Big O notation (worst case)
   - Time, memory, space complexity
   - অনেকগুলি complexity থাকলে higher টা হবে complexity
3. Array te searching complexity
4. Array reverse
   - new array নিলে space — O(n) হবে
   - 2. pointer approach follow করলে — O(1) হবে