TECONISE CLASS

Saturday, July 17, 2021 10:01 AM

CLASS-1

Date-17.07.21

Leet code practice: https://leetcode.com/

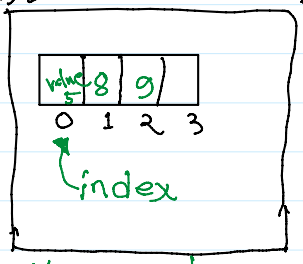
PART-1: 10-12 am

ARRAY:

1. In python it is called list.

a = [ ] → Empty list

a.append(5)
a.append(8)
a.append(9)



To insert a value at the end

a.pop() → To delete last element

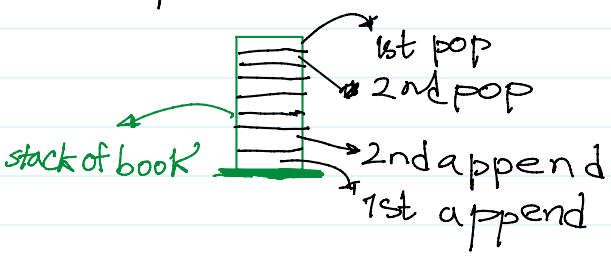
2. Two term in list
   i) index → start from zero.
   ii) value

append works → (9,a), (8,a), (7,a), (10,a). Pop → P, P, P

Li = [9,8,7,10]

Li = [9]

3. push(append) and pop can compare to stack of book.



4. Li = [4,2,3,4]
   print(Li) → it will print the memory address.

print(li) → it will print the memory address.

print(li[0]) → it will print 4

a.append(7)

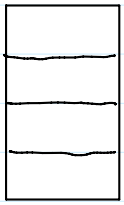
print(len(a)) → 5

5. DINAMIC ARRAY: Change the size of array wot data. ? How works ?

COMPLEXITY:

Big O Notation:

we have stack of book,



Worst case → our needed book is in bottom of the stack.

a = | 5 | 6 | 8 | 9 | 11 | 13 |
0 1 2 3 4 5

→ when we search 11. Then it will search 1st position then 2nd, 3rd. In fourth position we get 11.

```
for i in len(a):
    if a[i]==11:
        print("11")
```

O (of this array) ↔ order

worst case → depends on size of array

* If we search 1K list element, we will need 1K iteration.

complexity:

i) Time complexity: O(n), O(1)

Takes n iteration for worse case

Takes one iteration

i) Time complexity: O(n), O(1)
Takes one iteration when search.
Constant time (Best case)

ii) Memory Complexity: Write with the Big O notation

Q: When a element is multiple occurs multiple times? 0 for duplicate?

a: When we have one item in list what is time complexity?
It not always O(1). The correct ans it O(n)
Because list size not always one.

If n is always 1 then the complexity O(1)

QUESTION:

```
a = [3,4,8,10,11,13]
target =8
for element in a: Membership fracking?

    if target == element:
        print("Found")

    else:
        print("Not found")
```

while loop: M.C → O(1) size always same.
T.C → O(n)

```
a = [1,8,13,5,4,6]

start = 0
end = ln(a)
target = 5
while start<end :
    if a[start] == target :
```

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

    start += 1

start	end	target	a[start]	
0	6	5	1	a[0]
1	6	5	8	a[1]
2	6	5	13	a[2]
3	6	5	5	a[3]
4	6	5	4	a[4]
5	6	5	6	a[5]

print found . .

* viva te google does a program লিখতে হবে কয়: বলতে হবে প্রত্যেক লাইন কিভাবে লিখতেছি. Have to debug that program with table:

Q: Reverse a=[1,2,3,4]

ways:

i) Loop from len(a) to 1.

ii) a[::-1]

iii) inplace swap → last index → first (two pointer approach)

i)

Memory block 1 [O(1)]

a=[1,2,3,4]

result=[ ] ← নতুন একটা Memory নিয়েছে।

start = len(a) - 1

end = 0

while start >= 0 :

    result.append(a[start])

```
while start>=0:
    result.append(a[start])
start -=1
print(a)
```

time complexity → O(n)
Memory complexity → O(n)
* Memory every iteration a increase হচ্ছে।

start	a[start]	value of a[start]
3	a[3]	4
2	a[2]	3
1	a[1]	2
0	a[0]	1

Q: এখানে start -=2 হলে time and memory complexity কত হবে?

Input array size n
output array size is n/2
Here you can ignore the coefficient of n --> 1/2
Because if you make n larger and larger, multiplying half does not matter.
So it's still O(n).

1 কোটি data তে search করতে 31 year লাগে

$O(n/2) \equiv O(n) \equiv O(1.5n)$

$O(n-1) \equiv O(n-2) \equiv O(c*n) \not\equiv O(n*n)$

↳ constant

$\equiv O(1n*n)$

H.W: Array কে reverse করানোর কোড

Q: time complex কমাও [target O(1)]

2. Leetcode Q: 1 no probl m.
3. Time complexity/MC related video

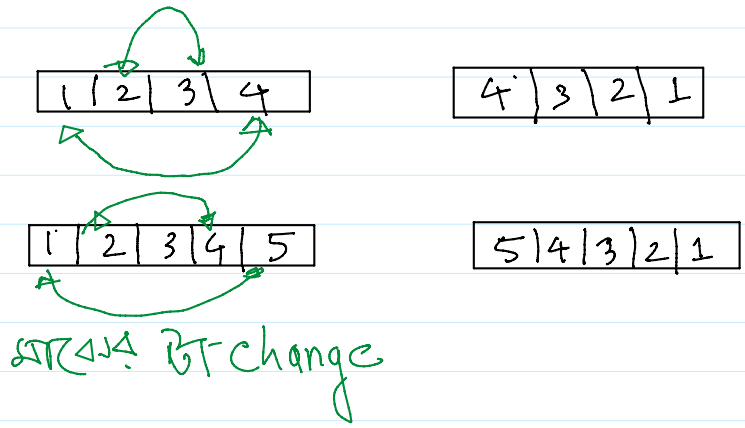
PART-2: 12.20 - 1.20 pm

Q: a=[1,2,3,4] এ array reverse

Q: a = [1, 2, 3, 4] ← array reverse

T: O(n)
M: O(n)
→ reduce → M: O(1)

Ans:



সবসময় 2টা change

```
array = [1,2,3,4]
start = 0;
end = len(a) - 1;
while (start < end):
    temp = a[start];
    a[start] = a[end];
    a[end] = temp;
    start ++;
    end --
```

```
a = [1,2,3,4,5,6]
lo=0;hi=len(a)-1;
while lo<hi:
   tmp = a[lo]
   a[lo]=a[hi]
   a[hi]=tmp
   lo+=1
   hi-=1
print(a)
```

Data structure → কিভাবে data কে সাজিয়ে কিভাবে সংরক্ষণ করা যায়?

Algorithm → Way of program writing.

T.C O(1)

s = 0 ← O(1) → MC

O(1) e = len(a) − 1 ← O(1) a[s], a[e] = a[e], a[s]

while s < e:

O(n/2)
- temp = a[s] ← O(1)
- a[s] = a[e]
- a[e] = temp
- s + 1 = 1
- e − = 1

(s বাড়ছে, e কমছে)

return a ← O(1)

Here, we don't need extra memory, so Memory complexity is: O(1)

But n time iteration so T.C = O(n)

Memory Complexity = $O(1) + O(1) + O(1) = O(1)$

Time complexity = $O(1) + O(1) + O(n/2) + O(1)$

$= O(n)$