Introduction to Arrays

Monday, 22 February 2021 9:08 PM

Arrays

· Linear DS.

int A[5]; 0 1 2 3 4

The element at index i in the array A → A[i] → TC: O(1)

int A[N]; A 0 — N-1

⊙ print an array!

N

print(A[0]);
—(A[1]);
⋮
—(A[N-1]);

for(i: 0; i<N; i++){ print(A[i]); }

Q Given N elements, count no. of element having atleast 1 element greater than itself? [No extra] [L6]

A[7] = [-3, -2, 6, 8, 4, 8, 5] → 5

A[8] = [2, 3, 10, 7, 3, 2, 10, 8] → 6

2 2 3 3 7 8 10 10 → X

Ans → N − cnt of Max element!

⊙ If I give you an array & x find the count of x...

Obs
1) find the MAX element ✓
2) find the count of it ✓
3) Ans = N − count ✓

A, N

```
maxi = INT_MIN / A[0]
for(i: 0; i<N; i++){
    if(A[i] > maxi){
        maxi = A[i];
    }
}
```

i: [0, N-1] O(N)

```
cnt = 0;
for(i: 0; i<N; i++){
    if(A[i] == maxi){
        cnt++;
    }
}
print(N - cnt);
```

i: [0, N-1] O(N)

: N+N = 2N

TC: O(N)

A: 2, 3, 10, 7, 3, 2, 10, 8

pMax : 10

cnt : 2

```
pMax = INT_MIN;
cnt = 0
for(i: 0; i<N; i++){
    if(A[i] > pMax){
        pMax = A[i];
        cnt = 1;
    }
    else if(A[i] == pMax){
        cnt++;
    }
}
N - cnt;
```

i: [0, N-1] TC: O(N)

SC: O(1)

1 <= N <= 10^5

TL: 1 sec

10^8 op/sec

N <= 10^3 O(N^2) ✓ $(10^3)^2 = 10^6$ → 1ms

TL: 1 sec

+, − ; *, / ; %

100 N^2 + 10 N log N

100 · 10^6 + 10^3 · 13

10^8 + ...

Q Given N array element, check if there exists a pair i, j such that A[i] + A[j] == K && i != j

Given: A[], N, K Output: Bool

A: [3, -2, 1, 4, 3, 6, 8] (0 1 2 3 4 5 6)

K=10 A[3] + A[5] = 10

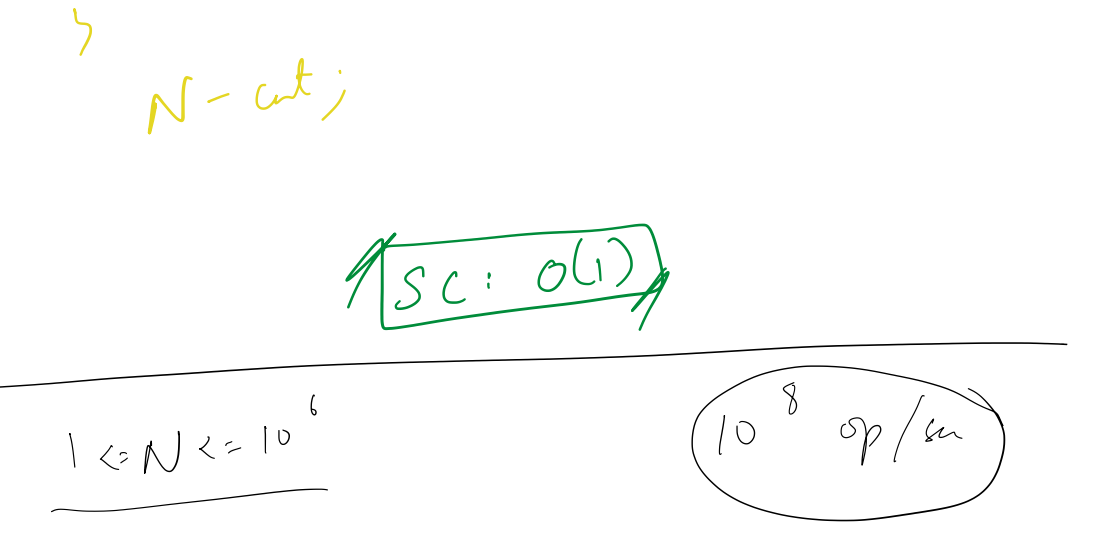
o/p → True

A: [2, 4, -3, 7] (0 1 2 3)

K: 8 A[i] + A[j] = 8

o/p → false

i: 0 [1, 3]
i: 1 [2, 3]
i: 2 [3, 3]



A[4]: 0 1 2 3

(0,0) (0,1) (0,2) (0,3)
(1,0) (1,1) (1,2) (1,3)
(2,0) (2,1) (2,2) (2,3)
(3,0) (3,1) (3,2) (3,3)

i → j: [i+1, N-1]

(0, 3) → A[0] + A[3] == K ?
(3, 0) → A[3] + A[0] == K ?

```
for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        if(i != j){
            if(A[i]+A[j]==K){
                ret True;
            }
        }
    }
}
ret False;
```

TC: O(N^2)

i	j: [0, N-1]	#:
0	(0, N-1)	N
1	(0, N-1)	N
⋮	⋮	N
N-1		

N^2

```
for(i=0; i<N-1; i++){
    for(j: i+1; j<N; j++){
        if(A[i]+A[j]==K){
            ret True;
        }
    }
}
ret False;
```

i	j [i+1, N-1]	#it
0	(1, N-1)	N-1
1	(2, N-1)	N-2
⋮		⋮
N-2	(N-2, N-1)	2

Avg = $\frac{(N-1)N}{2}$ → $\frac{N^2}{2} - \frac{N}{2}$ → O(N^2)

N: 10^5 N^2 → 10^{10} → 100 sec

$\frac{N^2}{2}$ → 5 · 10^9 → 50 sec

TL

doubt

```
for(i: 0 → N-2){
    for(j: i+1 → N-1){
        if(Ai + Aj == K)
            ret true;
    }
}
ret F;
```

A: [0 1 2 ...]

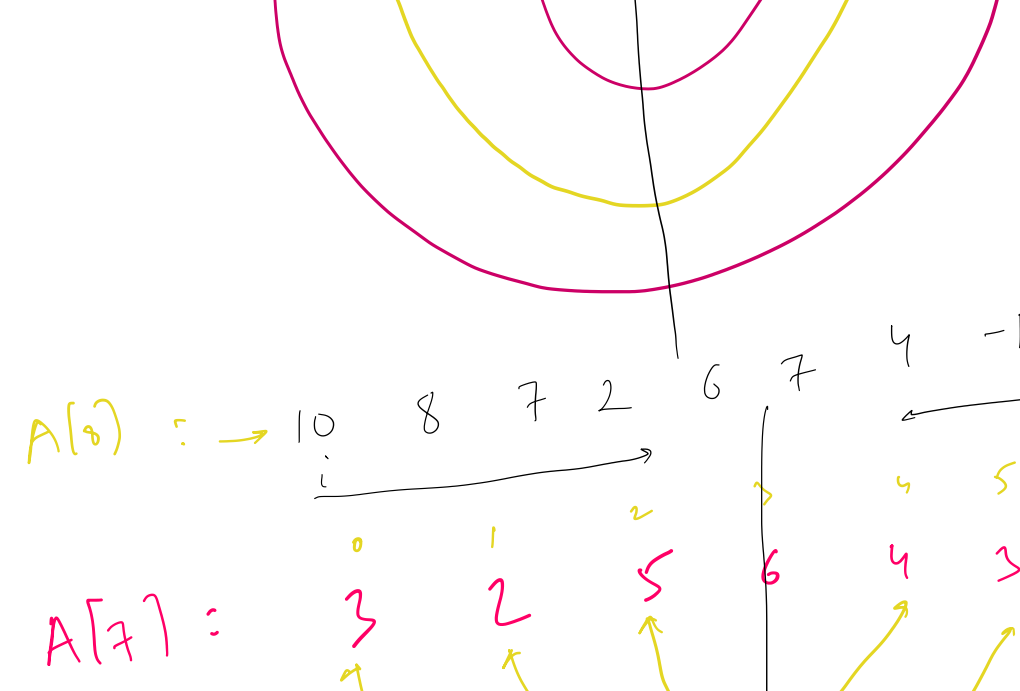
i: 0 j: 1 — 2 (0,1) (0,2)

i: 1 j: 2 (1, 2)

(1,2)

Q Given an array. Reverse the entire array. INPLACE Expected SC: O(1)

A[8]: -1 4 7 6 2 7 8 10 (0 1 2 3 4 5 6 7)



A[8] → 10 8 7 2 6 7 4 -1

A[7]: 3 2 5 6 4 3 2

```
while(i < j){
    swap(A[i], A[j]);
    i++;
    j--;
}
```

N/2

TC: O(N)

A 0 1 2 3 ... N-2 N-1

i → j
0 → N-1
1 → N-2
i → (N-1-i)

```
for(i: 0 → N/2){
    swap(A[i], A[N-1-i]);
}
```

TL: 1 sec

N: 10^6 → N, N log N
N: 10^5 → N, N log N, N√N
N: 10^3 → N^2
N: 20 → 2^N
N: 10 → N!

O(N) N <= 10^8

Q Given N elements... [L, R] reverse array from [L, R]. L <= R

(L, R) (3, 7)

A: [-1 3 2 3 7 5 9 2 8 10]

```
reversePart(A[], L, R){
    i = L, j = R;
    while(i < j){
        swap(A[i], A[j]);
        i++;
        j--;
    }
}
```

TC: R−L+1 → O(R−L)

0 — N-1

R-L ⇒ N

O(N)!

Q Given an array of N elements. Rotate array from last to first K times. INPLACE Expected SC: O(1)

	0	1	2	3	4	5	6
A:	3	-2	1	4	6	9	8
K=1:	8	3	-2	1	4	6	9
K=2:	9	8	3	-2	1	4	6
K=3:	6	9	8	3	-2	1	4

Idea: 1) Do as said!

temp

op → 1 rotation: O(N)

#op → K rot: O(KN) → 10^{12}

1 <= N, K <= 10^6

TL: 1 sec

Idea: K=4. N-4 4

Last 4 elem → 1st 4 elem
first N-4 → Last N-4 elem

A: -2 3 1 4 6 2 8 7 9 3 (0 1 2 3 4 5 6 7 8 9)

Rotate by K: 3 : 7 9 3 -2 3 1 4 6 2 8

A: a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8

K=3 WANT: a_6 a_7 a_8 a_0 a_1 a_2 a_3 a_4 a_5

rev A: a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 (K-1, K; rev, rev)

a_6 a_7 a_8 a_0 a_1 a_2 a_3 a_4 a_5

Steps:
1) Rev the entire array rev(0, N-1) → O(1)
2) Rev the 1st K elem rev(0, K-1) → O(1)
3) Rev the last N-K elem rev(K, N-1) → O(1)

SC ✓

SC: O(1)

TC: O(N)

⊙ 1 <= N <= 10^6 1 <= K <= 10^9

K=0	a_0	a_1	a_2	a_3	K=4	K=8	12	16
K=1	a_3	a_0	a_1	a_2	K=5	9	13	17
K=2	a_2	a_3	a_0	a_1	K=6	10	14	18
K=3	a_1	a_2	a_3	a_0	K=7	11	15	19 ...

→ K%N==0
→ K%N==1
→ K%N==2

10%4 → 2

N → #rev = N

K → K = K % N → [0, N-1]

⊙ Dynamic Array

int A[5];

Inc/dec their size as per demand!

C++	JAVA	Python	C#	JS
Vector	ArrayList	list	ArrayList	Array

vector<int> v;
v.size() → ? 0
v.push_back(8)
v.size() → ? 1
v.push_back(10)
? 2.

// add ability at back → TC: O(1)

add N → O(N) time

Amortized!

// Access ith element in O(1)

v[i] → O(1)

A[N]

```
vector<int> v;
for(i: 0 → N-1){
    v.push_back(A[i]);
}
```

→ O(N)