

# Todays Content

1. Continuous ✓
2. Water logging

## Two Queries

Given  $\text{arr}[n]=0$  & Q queries

for each query

Given  $(s, v)$ : Add  $v$  to all index elements from index  $\{s..n-1\}$

Once all queries are done return final arr[].

Ex1:

$$N=7 \quad Q=3$$

$$\begin{array}{c} \text{arr[]} = \underline{0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6} \\ Q[3][2] \quad \underline{0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0} \end{array}$$

$$\begin{array}{cc}
 s & v \\
 \hline
 1 & 3 & 3 & 3 & 3 & 3 & 3 \\
 4 & -2 & & -2 & -2 & -2 \\
 8 & 1 & & 1 & 1 & 1 & 1 \\
 \hline
 & \underline{0 \quad 3 \quad 3 \quad 4 \quad 2 \quad 2 \quad 2}
 \end{array}$$

Ex2:

$$N=7 \quad Q=5$$

$$\begin{array}{c} \text{arr[]} = \underline{0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6} \\ Q[5][2] \quad \underline{0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0} \end{array}$$

$$\begin{array}{cc}
 s & v \\
 \hline
 2 & 6 \checkmark & 6 & 6 & 6 & 6 & 6 \\
 0 & -1 \checkmark & -1 & -1 & -1 & -1 & -1 & -1 \\
 3 & 2 \checkmark & 2 & 2 & 2 & 2 \\
 5 & 4 \checkmark & & 4 & 4 \\
 3 & 3 & 3 & 3 & 3 & 3 \\
 \hline
 & \underline{-1 \quad -1 \quad 5 \quad 10 \quad 10 \quad 14 \quad 14}
 \end{array}$$

Idea: For every query  $(s, v)$  TC:  $O(Q \cdot N) = O(Q \cdot N)$  SC:  $O(1)$

Iterate q and v from  $[s..n-1]$

↳ We are returning arr[]

Idea 2: Step 1: for every query  $(s, v)$ :  $\text{arr}[s] += v$

Step 2: Apply pf[] in array

TC:  $O(8 + N)$  SC:  $O(1)$

fn1:

$N = 7$   $Q = 3$

		0	1	2	3	4	5	6
		0	X	0	X	X	0	0
s	v	0 + 3	0	1	-2	0	0	
1	3	0	3	3	4	2	2	2
4	-2							
3	1							

fn2:

$N = 7$   $Q = 5$

		0	1	2	3	4	5	6
		0	0	0	0	0	0	0
s	v	-1	0 + 6	5	0	4	0	
2	6 ✓	-1	-1	5	10	10	14	14
0	-1 ✓							
3	2 ✓							
5	4 ✓							
3	3 ✓							

Two Queries

Given  $\text{arr}[n] = 0$  & Q queries

for each query

Given  $(s, e, v)$ : Add v to all index elements from index  $\{s \dots e\}$

Once all queries are done return find  $\text{arr}[]$ .

Ex1  $N=7$   $Q=4$

$$\text{arr}[] = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

s	e	v					
1	4	3	3	3	3	3	
0	5	-1	-1	-1	-1	-1	-1
2	2	4	4				
4	6	3		3	3	3	
			-1	2	6	2	5
						2	3

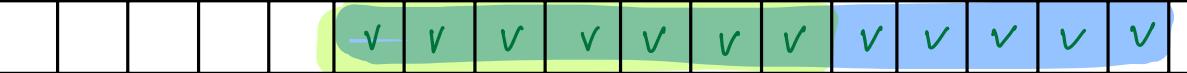
Ex2  $N=7$   $Q=4$

$$\text{arr}[] = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

s	e	v					
2	5	7	7	7	7	7	7
1	3	2	2	2	2	2	
2	4	1	1	1	1	1	
3	6	2	2	2	2	2	2
		0	2	10	12	10	9
							2

Idea: For every query  $(s, e, v)$ :  
Iterate from  $[s..e]$  & add v.  
TC:  $O(Q \times N)$  SC:  $O(1)$

think: 0 1 2 ... s s+1 e-1 e e+1 ... n-1

arr[N]: 

-v -v -v -v -v

Query:

start

Previous Query:

$s \leftarrow v$ : Add  $v$  from  $s..N-1$ :  $\text{arr}[s] += v$ ;

Add  $-v$  from  $e..N-1$ :  $\text{arr}[e+1] -= v$

$$Q(s, e, v) = Q(s, v), Q(e+1, -v)$$

Dry Run:

$N=7$

$$Q[4][3] \quad \text{arr}[] = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 \end{array}$$

$$\begin{array}{|c|c|c|c|c|c|c|c|} \hline s & e & v & 2 & 8 & 2 & -2 & -1 & -7 \\ \hline \end{array}$$

$$2 \quad 5 \quad 7 \quad \checkmark \quad \text{arr}[2] += 7 \quad \text{arr}[6] += (-7)$$

$$1 \quad 3 \quad 2 \quad \checkmark \quad \text{arr}[1] += 2 \quad \text{arr}[4] += (-2)$$

$$2 \quad 4 \quad 1 \quad \checkmark \quad \text{arr}[2] += 1 \quad \text{arr}[5] += (-1)$$

$$3 \quad 6 \quad 2 \quad \checkmark \quad \text{arr}[2] += 2 \quad \text{arr}[7] += (-2) \quad \# \text{skip it}$$

$$\text{arr}[] = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 \end{array}$$

$$\begin{array}{ccccccc} 0 & 2 & 8 & 2 & -2 & -1 & -7 \\ \hline \end{array}$$

$$\begin{array}{ccccccc} 0 & 2 & 10 & 12 & 10 & 9 & 2 \\ \hline \end{array}$$

# Idea:

For every query  $(s, e, v)$ :

TC:  $O(Q + N)$  SC:  $O(1)$

$$\text{arr}[s] += v$$

if  $(e+1 \leq N)$  {

$$\text{arr}[e+1] -= v$$

}

Apply prefix sum on arr[]

38 Given an  $arr[N]$ , create Prefix Max array.

$Pf[i] = \text{Max of all elements from } [0..i]$

Eg:

$$arr[] = \{ 1 \ 6 \ 3 \ 3 \ 8 \ 7 \}$$



$$Pf[6] = \{ 1 \ 1 \ 3 \ 3 \ 8 \ 8 \}$$

Idea: For every element:

We iterate on left to calculate max.

We can optimize above using carry forward

$$arr[] = \{ 1 \ 6 \ 3 \ 3 \ 8 \ 7 \}$$

Steps:

$$\text{man} = -\infty \rightarrow 1 \rightarrow 1 \rightarrow 3 \rightarrow 3 \rightarrow 8 \rightarrow 8$$

Carry forward man from L  $\rightarrow$  R

$$Pf[6] = \{ 1 \ 1 \ 3 \ 3 \ 8 \ 8 \}$$

1. update man
2. store in array.

$\text{vector<int>} Pf \text{ man}(\text{vector<int>} &arr) \{ \text{TC: } O(N) \text{ SC: } O(1)$

```
int N = arr.size();
```

```
vector<int> Pf(N, 0);
```

```
int man = -\infty;
```

```
for (int i = 0; i < N; i++) {
```

```
    if (arr[i] > man) {
```

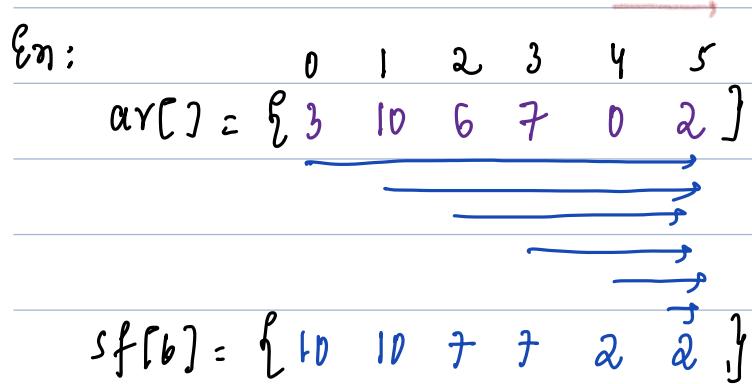
```
        man = arr[i];
```

```
        Pf[i] = man;
```

```
}
```

```
return Pf;
```

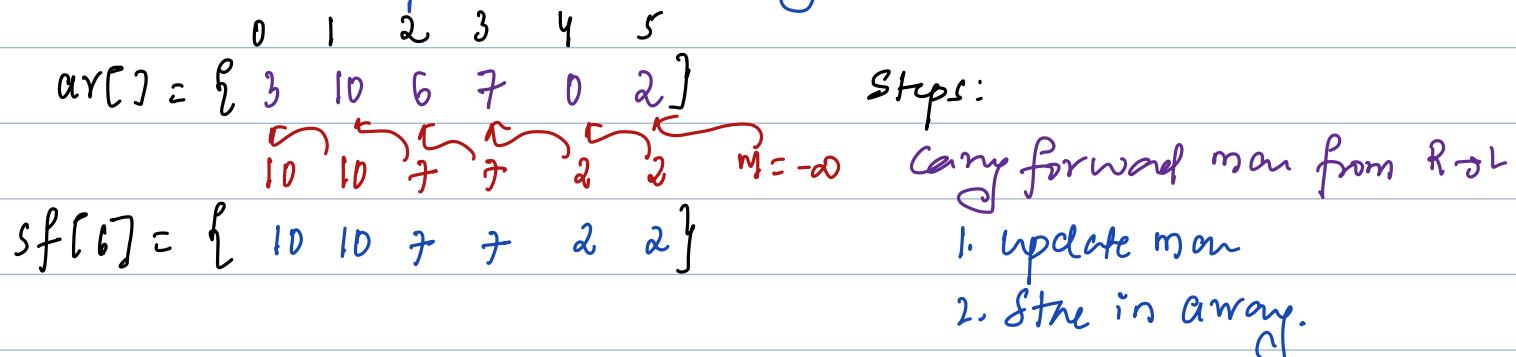
38 Given an arr[N], create suffix max array.  
 $sf[i] = \max$  of all elements from  $[i..N-1]$



Idea: For every element:

We iterate in right to calculate max.

We can optimize above using carry forward



`vector<int> sfmax(vector<int> &arr) { TC: O(N) SC: O(1)`

`int N = arr.size();`

`vector<int> sf(N, 0);`

`int man = -∞;`

`for (int i = N-1; i >= 0; i--) {`

`if (arr[i] > man) {`

`man = arr[i];`

`sf[i] = man;`

`}`

`} return sf;`

