

Today's Content

1. Continous sum ✓

2. Water logging

7th Query

Given $arr[N]=0$ & 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 find arr[].

Ex1:

$N=7$ $Q=3$

$arr[] =$ 0 1 2 3 4 5 6

$Q[3][2]$ 0 0 0 0 0 0 0

s v

1 3 3 3 3 3 3 3

4 -2 -2 -2 -2

3 1 1 1 1 1

0 3 3 4 2 2 2

Ex2:

$N=7$ $Q=5$

$Q[5][2]$ 0 1 2 3 4 5 6

s v

2 6 ✓ 6 6 6 6 6

0 -1 ✓ -1 -1 -1 -1 -1 -1

3 2 ✓ 2 2 2 2

5 4 ✓ 4 4

3 3 3 3 3 3

-1 -1 5 10 10 14 14

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

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

↳ We are returning $arr[]$

idea2: Step1: for every query (s, v) : $ar[s] += v$;
 Step2: Apply pf[] on array
 TC: $O(Q + N)$ SC: $O(1)$

ex1:

$N = 7$ $Q = 3$

$ar[] =$	0	1	2	3	4	5	6
$Q[3][2]$	0	1	0	1	1	0	0
$s \quad v$	0	+3	0	1	-2	0	0
1 3	0	3	3	4	2	2	2
4 -2							
3 1							

ex2:

$N = 7$ $Q = 5$

	0	1	2	3	4	5	6
$Q[5][2]$	0	0	0	0	0	0	0
$s \quad v$	-1	0	+6	5	0	4	0
2 6 ✓	-1	-1	5	10	10	14	14
0 -1 ✓							
3 2 ✓							
5 4 ✓							
3 3 ✓							

7th Query

Given $arr[N]=0$ & 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 $arr[]$.

Ex1 $N=7$ $Q=4$

$arr[] =$

0	1	2	3	4	5	6
0	0	0	0	0	0	0

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

Ex2 $N=7$ $Q=4$

$arr[] =$

0	1	2	3	4	5	6
0	0	0	0	0	0	0

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

Idea: For every query (s, e, v) :
Iterate from $[s \dots e]$ & add v .

Tc: $O(Q * N)$ Sc: $O(1)$

Hint:	0	1	2	...	s	s+1				e-1	e	e+1	...	N-1
ar[N]:						v	v	v	v	v	v	v	v	v

-v -v -v -v -v

Query: start Previous Query:

s e v : Add v from s.. N-1 : ar[s] += v;

Add -v from e+1.. N-1 : ar[e+1] -= v;

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

Dry Run:

N=7

Q[4][3]	ar[] =	0	1	2	3	4	5	6
		0	0	0	0	0	0	0

s	e	v	
2	5	7	2 8 2 -2 -1 -7

2 5 7 ✓ ar[2] += 7 ar[6] += (-7)

1 3 2 ✓ ar[1] += 2 ar[4] += (-2)

2 4 1 ✓ ar[2] += 1 ar[5] += (-1)

3 6 2 ✓ ar[3] += 2 ar[7] += (-2) # skip it

ar[] =	0	1	2	3	4	5	6
	0	0	0	0	0	0	0

0	2	8	2	-2	-1	-7
---	---	---	---	----	----	----

0	2	10	12	10	9	2
---	---	----	----	----	---	---

Idea 2:

for every query (s, e, v):

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

ar[s] += v

if (e+1 < N)

ar[e+1] -= v;

}

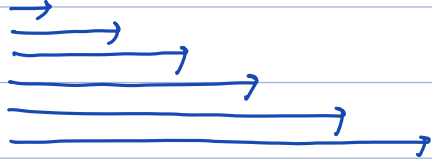
Apply prefix sum on ar[]

38 Given an $arr[N]$, create Prefin Man array.

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

Ex:

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



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

#Idea: For every element:

We iterate n left to calculate man.

We can optimize above using carry forward

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

$man = -\infty$

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

Steps:

Carry forward man from $L \rightarrow R$

1. Update man

2. Store in array.

`vector<int> pfman(vector<int> arr) { TC: $O(N)$ 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.

$sf[i] = \text{Max of all elements from } [i..n-1]$

Ex:

arr = { 3 10 6 7 0 2 }

sf = { 10 10 7 7 2 2 }

#Idea: For every element:

We iterate n right to calculate max.

We can optimize above using carry forward

arr = { 3 10 6 7 0 2 }

sf = { 10 10 7 7 2 2 }

Steps:

Carry forward max from R to L

1. update max

2. store in array.

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

int N = arr.size();

vector<int> sf(N, 0);

int max = $-\infty$;

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

if (arr[i] > max) {

max = arr[i];

sf[i] = max;

}

return sf;





