Two Pointers

Abhishek Sharma
Akansh Nirmal
amit khandelwal
Bhaveshkumar
Gagan Kumar S
Gowtham
Nikhil Pandey
Pankaj
Prajwal Khobragade
Rajat Sharma
Rajendra
Sanket Giri
Saurabh Ruikar
sharath r
Shradha Srivastava
Sneha L
Subhashini
Subhranil Kundu
Sumit Adwani
Vasanth
Venkata Sribhavana Nandiraju
Vimal Kumar
Vishal Mosa

GREAT JOB!

AGENDA:

→ Pair sum = k
→ Count of pair with sum = k
→ Pair diff = k
→ Subarray sum = k
→ Container with most water.

Current PSP

64 → 70%

Contest Next week Friday

Hashing, sorting, search

Pair with sum K

Given a sorted array A and an int K.
Find any pair (i,j) such that

SC : O(1)

$$A_i + A_j == K \quad \&\& \quad i \neq j$$

A = -5 , -2 , 1 , 8 , 10 , 12 , 15 K = 11

ans = true

A = -3 0 1 3 6 8 11 14 18 25 K = 12

ans = true

Brute force $\forall i, j$ $A_i + A_j == K$ & $i \,!= j$ TC : $O(N^2)$

return true SC: O(1)

Idea 2 ∵ A is sorted Binary search

$$A_i + A_j = K$$

Given $A_i \longrightarrow A_j = K - A_i$

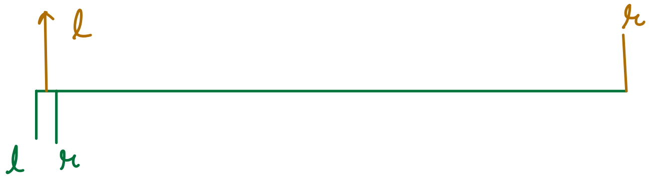
-3 0 1 3 6 8 11 14 18 25 K= 12

↑ ↑ ↑

A_i	A_j	
-3	15	BS for 15 in array ⟶ No
0	12	BS for 12 ⟶ No ans
1	11	BS for 11 there is 11

TC : O(N logN) SC : O(1)

Two pointers



l and r can be initialized either at the

① same corner

② diff corner

-5 , -2 , 1 , 8 , 10 , 12 , 15 $\qquad K = 11$

l $\qquad$ r

A[l]	A[r]	total
-5	15	10

⟶ No ans can be found via

-5 ⟶ l++

A[l]	A[r]	total
-2	15	13

⟶ No ans can be found via

15 ⟶ r--

A[l]	A[r]	total	
-2	12	10	l++
1	12	13	r--
1	10	11	return true

$A[l] + A[r] < K$

↓ smallest $\qquad$ ↓ largest

smallest + any value

Opposite of above

Pseudocode

```
bool pairSumK ( A[] , K ) {
    l = 0
    r = N-1

    while ( l < r ) {
        total = A[l] + A[r]
        if ( total == k )   return true
        if ( total < k )   l++
        else   r--
    }
    return false
}
```

TC : O(N)

SC : O(1)

Same Q as above array is sorted
Find count of pairs with sum K in a distinct array.

1 2 3 4 5 6 8   K = 10

~~l~~ (under 1)   l r (under 5)

ans = ~~0~~ ~~1~~ 2

total = ~~9~~ ~~10~~ 9

```
l = 0
r = N-1
ans = 0
while ( l < r ) {
        total = A[l] + A[r]
        if( total == k )  ans++   l++ r--
    else if ( total < k )   l++
        else  r--
}
return ans
```

what if there are duplicates ?

$$[\ 2\ \underbrace{3\ \ 3}_{2}\ \ 5\ \ \underbrace{10\ \ 10\ \ 10}_{3}\ \ 15\ ] \qquad K = 13$$

r l (under 5)

A_l	A_r	total
2	15	17
2	10	12
3	10	

r--
l++ } same cond^n as above

3 10 → count 3s on left
count 10s on right *

ans = 6

K=14

0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	4	4	4	5	5	7	7	7	7	10	10	10	15
~~l~~	~~l~~			~~l~~	~~l~~	l			r			~~r~~	~~r~~

A_l	A_r	total	
2	15	17	r--
2	10	12	l++
4	10	14	3 4_s * 3 10_s = 9
5	7	12	l++
5	7	12	l++
7	7	14	A[l] == A[r]

count = r - l + 1 = 4

$$\frac{count * count - 1}{2} = \frac{4*3}{2} = 6$$

ans = 9 + 6 = 15

6	7	8	9
7	7	7	7

6 7   7 8   8 9

6 8   7 9

6 9

6

```
int countSumK ( A[] , K ) {
    l = 0
    r = N - 1
    ans = 0
    while ( l < r ) {
        total = A[l] + A[r]
        if ( total < k )      l++
        else if ( total > k )     r--
        else {   total == k
            // A[l] == A[r]
            if ( A[l] == A[r] ) {
                count = r - l + 1
                ans += count * (count - 1)/2
                return ans
            }
            // A[l] != A[r]
            cnt l = 0
            lval = A[l]
            while ( A[l] == lval ) {
                cntl ++
                l++
            }
            cntr = 0
            rval = A[r]
            while ( A[r] == rval ) {
                cntr++
                r--
            }
            ans += cntl * cntr
        }
    }
    return ans
}
```

TC : O(N)

SC : O(1)

Q) Given a sorted integer array A and an integer k, find any pair {i, j} such that

$A_j - A_i == k$ & $i \neq j$ $K > 0$

SC : O(1)

$A = \{ -5 \quad -2 \quad 1 \quad 8 \quad 10 \quad 12 \quad 15 \}$ $k = 11$

ans = true

Quiz

$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [5 & 4 & 2 & 12 & 1 & 6] \end{array}$$ $k = 10$

Brute force $\forall i\ j$ check $A_j - A_i == k$ & $i \neq j$

TC : $O(N^2)$ SC : O(1)

Idea 2 Binary search

A_i is given

$A_j = k + A_i$

$\forall i$ search $k + A_i$ in the sorted array

TC : O(N log N)

SC : O(1)

Idea 3

Initialising l & r at opp corners doesn't help

A = {	-5	-2	1	8	10	12	15 }	k= 11
	~~l~~	~~l~~	~~l~~	l		r	~~r~~	

A_l	A_r	diff {$A_r - A_l$}	
-5	15	20	l++
-2	15	17	l++
1	15	14	l++
8	15	7	
8	12	4	

no ans

		~~r~~	~~r~~	~~r~~	~~r~~	r		
A = {	-5	-2	1	8	10	12	15 }	k= 11
	~~l~~	~~l~~	l					

A_l	A_r	diff $A_r - A_l$	
-5	-2	3	r++ { increment diff }
-5	1	6	—||—
-5	8	13	l++ { decrement diff }
-2	8	10	r++ { increment diff }
-2	10	12	l++ { decrement diff }
1	10	9	
1	12	11	

Break — 22:39

Pseudo code

HW

```
bool   pairDiffK ( A[] , K ) {
```

Q) Given an A[] of +ve elements and an int K
check if there exists a subarray with sum K

$K = 33$ true

$A = \{ 1 \quad 3 \quad 15 \quad 10 \quad 20 \quad 3 \quad 23 \}$

$K = 43$ → false

Bruteforce

for each subarray check sum == K

TC : $O(N^3)$ $\xrightarrow[\text{carry forward}]{}$ TC : $O(N^2)$

Idea 2 Two pointers Dynamic sliding window

1 3 15 10 20 3 23 $K = 33$

l r

A_l	A_r	sum $(l \to r)$
1	23	75

If we do l++ $\Longrightarrow$ sum decreases

If we do r-- $\Longrightarrow$ sum decreases

~~r~~ ~~r~~ ~~r~~ ~~r~~ ~~r~~ r

1 3 15 10 20 3 23 K = 33

~~l~~ ~~l~~ ~~l~~ l

A_l	A_r	sum (l – r)	
1	1	1	inc r → r++
1	3	4	inc r → r++
1	15	19	—〃—
1	10	29	—〃—
1	20	49	l++ to dec sum
3	20	48	—〃—
15	20	45	—〃—
10	20	30	inc r → r++
10	3	33	—〃—

Pseudocode

TC : O(N)

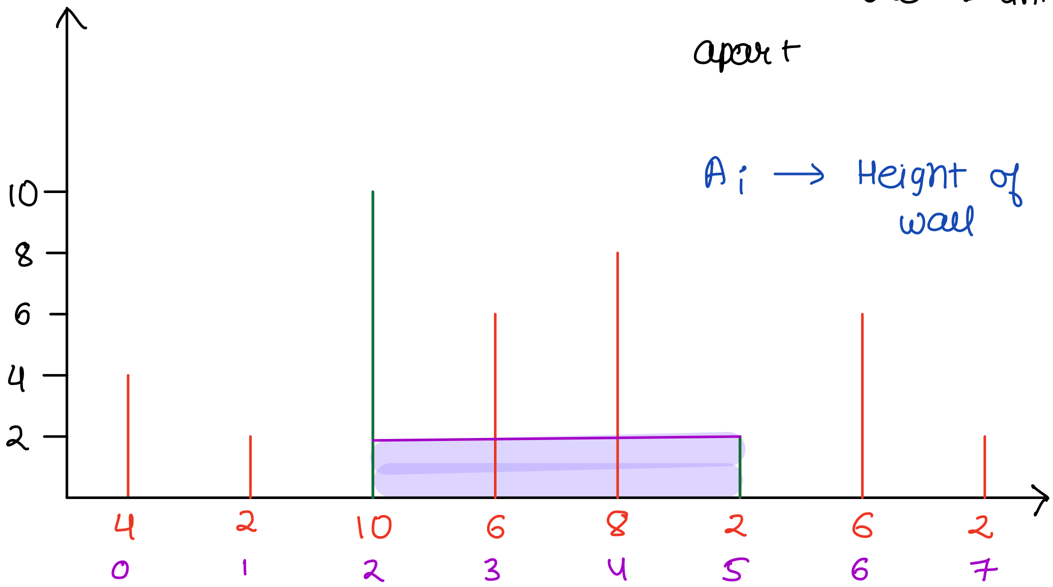
SC : O(1)

```
l = 0
r = 0
total = A[0]
while ( l<N && r < n ) {
        if ( total == k )  return true
        if ( total < k ) {
                r++
                if (r<N)    total += A[r]
        }
        else {
                total -= A[l]
                l++
        }
}
return false
```

Ques Container with most water.

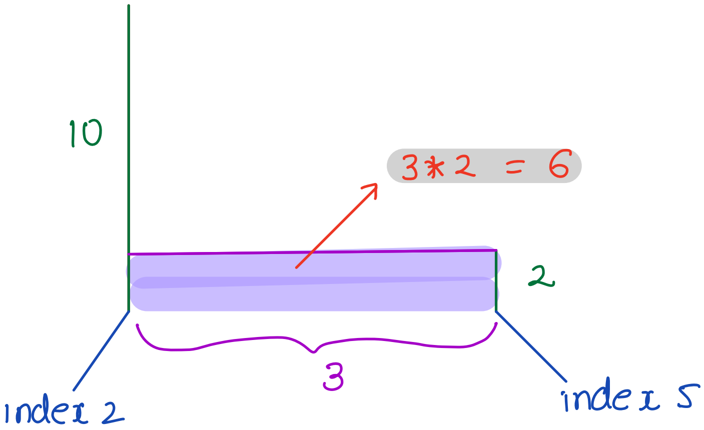
Find two walls that can form a container to store the maximum water.

All walls are 1 unit apart



A_i → Height of wall

A = 4 2 10 6 8 2 6 2

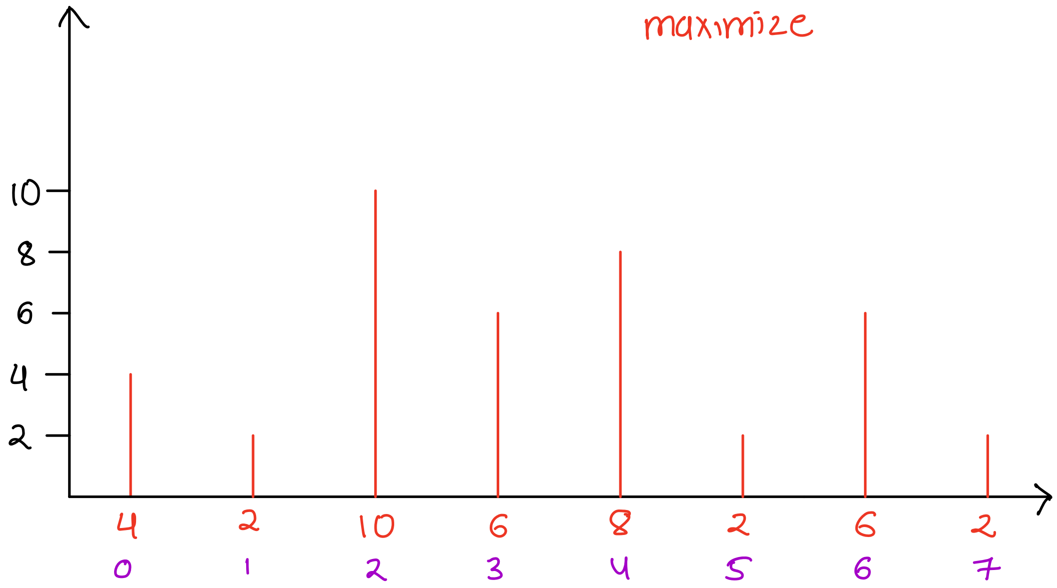


Height of water b/w to walls A_i & A_j = $\min(A_i, A_j)$

Total water b/w index L & R = $(R-L)$ width

$* \min(A_l, A_r)$ Height

$$\underbrace{(R-L) \; * \; \min(A_l , A_r)}_{\text{maximize}}$$



index	0	1	2	3	4	5	6	7
A	4	2	10	6	8	2	6	2
	~~l~~	~~l~~	l r	~~r~~	~~r~~	~~r~~	~~r~~	~~r~~

A_l	A_r	width	height	area
4	2	7-0	min(4,2)=2	7*2
4	6	6 - 0	min(4,6)=4	4*6
2	6	5	min(6,2)=2	2* 5
10	6	4	min(10,6)=6	6* 4
10	2	3	min(10,2) = 2	2*3
10	8	2	min(10,8) = 8	8 *2
10	6	1	min(10,6) =6	6 * 1

to inc
min height
↑
r--

Pseudocode

```
l = 0
r = N - 1
marea = 0
while (l < r) {
        height = min( A[l], A[r])
        width = r - l
        area = height * width
        marea = max (marea , area)
        if ( A[l] < A[r])    l++
        else  r--
}
print (marea)
```

Contest on next Friday