Today's Content :-

Pairs with given Sum - 2

Pairs with given diff.

Subarrays with given Sum

Container with most water.

Today's Quote :-

The more you sweat in peace,
the less you bleed on war.

Ques :- Given a sorted integer array A & an integer k, find any pair (i,j) s.t, $A[i] + A[j] = k$ & $i != j$,

S.C -> O(1) .

$$
\begin{array}{cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
A = [& -5, & -2, & 1, & 8, & 10, & 12, & 15] \\
 & & & i & & J & &
\end{array}
\qquad k = 11 .
$$

↳ True .

1) Brute force :- Check all pairs.

T.C -> $O(n^2)$, S.C -> O(1) .

2) Binary Search . $A[i] + A[J] = k$.

$A[J] = k - A[i]$

∀ i, Binary Search, $k - A[i]$, s.t $i != J$,

T.C -> O(nlogn)

S.C -> O(1) .

$$
\begin{array}{cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
A = [& -5, & -2, & 1, & 8, & 10, & 12, & 15] \\
 & i & J & & & & i & J
\end{array}
\qquad k = 11 .
$$

arr[i] + arr[J]

$-5 + (-2)$

$-7 < 11$

i++ or J++ ✗

arr[i] + arr[J]

$27 > 11$

i-- or J-- ✗

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = & [-5, & -2, & 1, & 8, & 10, & 12, & 15] \\ & \cancel{i} & \cancel{i} & i & & j & \cancel{j} & \cancel{j} \end{array} \quad k = \underline{11}$$

A[i]	A[j]	k	
-5	15	10<k	i++
-2	15	13>k	
-2	12	10<k	
1	12	13>k	
1	10	$\underline{11 = k}$	

-5 + largest element < k
-5+ any element < k

15+ smallest avail elem > k

15+ A[i] > k

```
i=0,    j=n-1
while(i<j) {
    if (A[i]+A[j]==k) { return True }
    else if (A[i]+A[j]<k) { i++ }
    else { j-- }
}
```

T.C -> O(n)
S.C -> $\underline{O(1)}$.

Ques find all the pairs, in distinct array.

i j

1 2 3 4 5 6 8 k=10.

Same as prev,

when arr[i] + arr[j] == k

count++;

i++ & j--

duplicates

[2 3 3 10 10 10 15]

i j

[2 3 10 15] sum=13

freq 1 2 3 1

Create a distinct array & store freq of every element, whenever find a pair c̄ sum=k, in distinct array multiply the frequencies.

k= 14

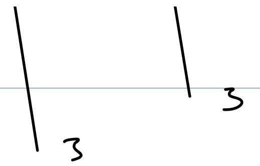
[2 4 4 4 5 5 7 10 10 10 15]

```
     ~~i~~     i     J     ~~J~~
     ii                    JJ
```

```
i = 0,   J = n-1
ans = 0;
while (i<J) {
    sum = A[i]+ A[J]
    if (sum < k) {
        i++
    }
    else if (sum > k) {
        J--
    }
    else {
        ii = i , JJ = J
        if (A[i] == A[J]) {
            cnt = J-i+1;
            ans += (cnt * (cnt-1))/2
            break;
        }
        else {
            while ( A[ii] == A[i]) {
                ii++
            }  → }
            cnt1 = ii-i
            i = ii
            while(A[JJ] == A[J]) {
                JJ--;
            }  → }
            cnt2 = J-JJ;
            J = JJ;
```

$\left[{}^{n}C_{2} \to \frac{n(n-1)}{2} \right]$

T.C → O(n)
S.C → O(1)



ans+= cnt1 * cnt2;

sum = 14

$$\begin{array}{ccccccccccccc}
 & i & & & & & i & & j & & & \cancel{j} & \\
[2 & 4 & 4 & 4 & 5 & 5 & 7 & 7 & 7 & 10 & 10 & 10 & 15]
\end{array}$$

Ques :- Given a sorted integer array A & an integer k, find any pair (i,j) s.t, A[j] - A[i] = k & i != j,

S.C -> O(1). k>0.

$$\begin{array}{cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
A = & [-5, & -2, & 1, & 8, & 10, & 12, & 15] \\
 & & & i & & & j &
\end{array}$$

k = 11.

Brute force :-

① check all pairs O(n^2).

② Binary search

arr[j] - arr[i] = k

↓

for other → arr[j] = k + arr[i]

no. do binary search.

idea :- Two Pointers .

0 1 2 3 4 5 6

A = [-5 -2, 1, 8, 10, 12, 15] k = 11 .

i under -5, J under 15

↓ arr[J] - arr[i] ↑ = 20 > k. X

↑x - y↓ < -

arr[J] - arr[i] = 11

→ This will work .

0 1 2 3 4 5 6

A = [-5, -2, 1, 8, 10, 12, 15] k = 11 .

i under 10, ~~i~~ under 12, J under 15

A[J] - A[i] = 15-12 = 3 < 11

↑b - a↓

largest element - A[i] < k

any element - A[i] < k

T.C → O(n)

S.C → O(1)

0 1 2 3 4 5 6

A = [-5, -2, 1, 8, 10, 12, 15] k = 11 .

i under 1, J under 12

A[J]	-	A[i]		k
-2	-	(-5)	=>	3 < 11
1	-	(-5)	=	6 < 11
8	-	(-5)	=	13 > 11
8	-	(-2)	=	10 < 11
10	-	(-2)	=	12 > 11
10	-	1	=	9 < 11

y - x < k

↑y

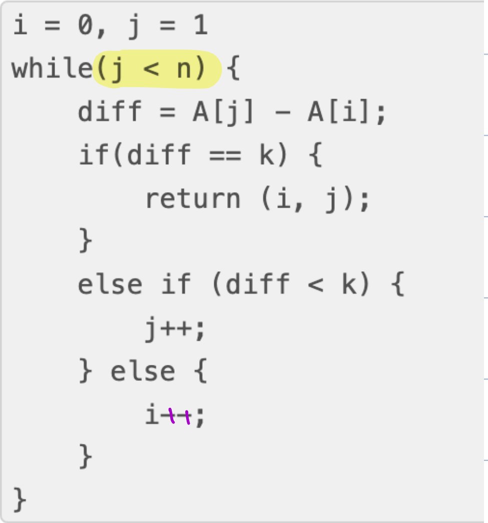
y - x > k

↑x

12 − 1 = 11 = 11 .

i j

i



```
i = 0, j = 1
while(j < n) {
    diff = A[j] - A[i];
    if(diff == k) {
        return (i, j);
    }
    else if (diff < k) {
        j++;
    } else {
        i++;
    }
}
```

Ques Given an integer array of +ve elements and an integer k.
Check if there exists a subarray with sum = k.

	0	1	2	3	4	5	6
A:	[1	3	15	10	20	3	23]

k = 33. True

k = 43, false.

subarrays.

$\frac{n(n+1)}{2}$.

Brute Force Check all subarray sums.
↳ carry forward
T.C → $O(n^2)$
S.C → $O(1)$

idea 2 :-

k = 33.

	0	1	2	3	4	5	6
A:	[1	3	15	10	20	3	23]
pf =	[1	4	19	29	49	52	75]

subarray

sum (i, j) — pf[j] − pf[i−1] . , i > 0
pf[j] i = 0

check pf[J]
↓
cover all subarrays
with i=0,

if P[J] − pf[i−1] = k,
& (i−J) = k
↓
Prev Problem
Pair diff = k

Todo, S.C → O(1), without changing i/p array.

ideas :-

k = 33

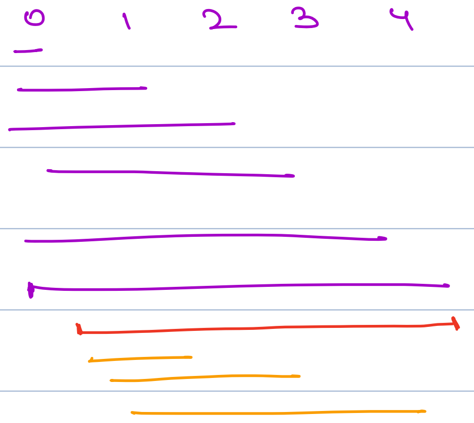
	0	1	2	3	4	5	6	7	8
A : [	1	3	15	10	20	3	23	33	43]
	~~i~~	~~i~~	~~i~~	i	~~J~~	J			

T.C → O(n)
S.C → O(1).

```
i=0, J=0, sum = A[0]
while (J<n) {
    if (sum == k) { return True }
    else if (sum < k) { J++;
                        if (J == n)
                          { break }
                        sum += A[J]
    } else {
        sum -= A[i];
        i++;
        if (i > J && i < n-1) {
            J++;
            sum += arr[J]
        }
    }
}
```

5 2

k=2



Sub arr

0 - 0 < k

0 - 1 < k

0 - 2 < k

0 - 3 < k

0 - 4 < k

0 - 5 > k

1 - 5

1 - 2

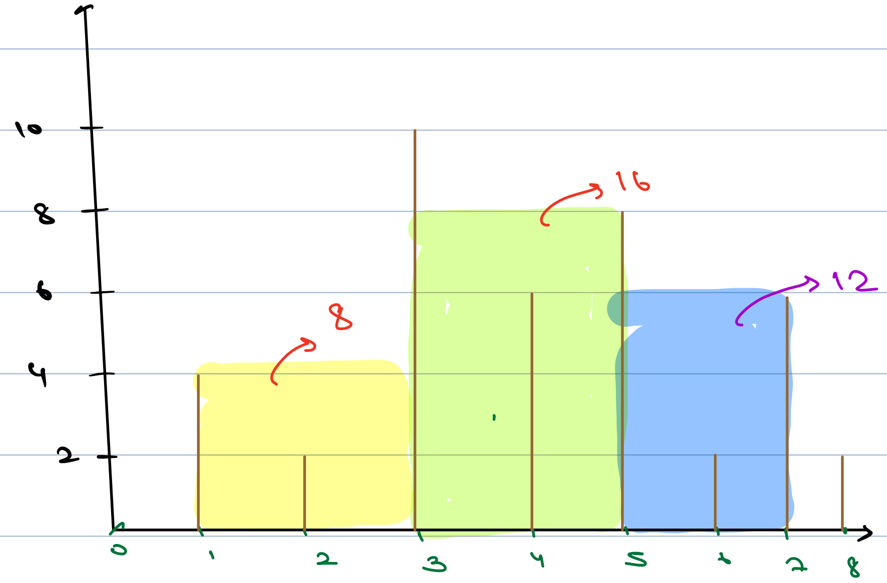
1 - 1

1 - 3

1 - 4

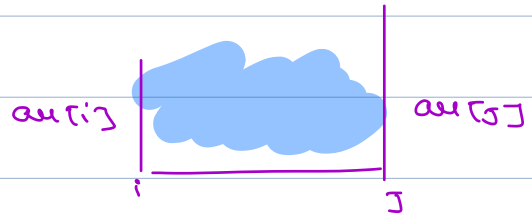
Containers with most water.

Ques) .



A [] → [4, 2 10 6 8 2 6 2]

find two walls that can form a container to store maxm water,



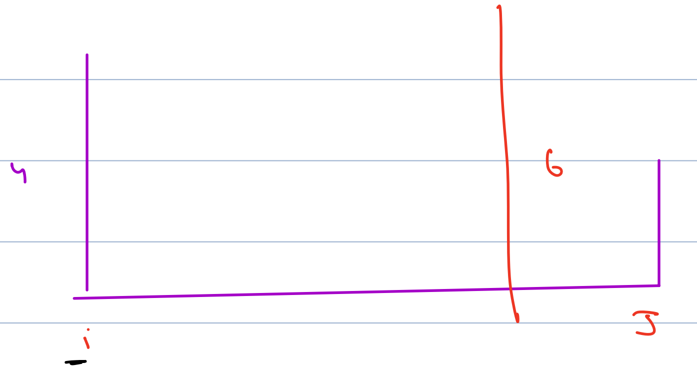
min(A[i], A[J]) * J-i

	0	1	2	3	4	5	6	7
A [] →	[4,	2	10	6	8	2	6	2]
			i	J				~~J~~

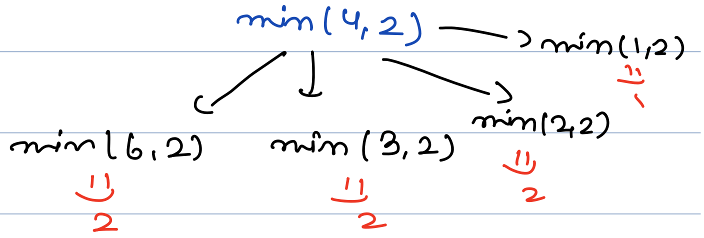
min(arr[i], arr[J]) * (J-i)

i	J	water stored
0	7	14
0	6	24
1	6	10
2	6	24
2	5	6

2	4	16
2	3	6



height * width
↓ ↓
decrease.



2 x 7 = 14
↓
4 x 6 = 24

$\boxed{2 \times 6 \text{ or } 1 \times 6}$

<u>idea</u> :- Always move the smaller <u>wall</u>.

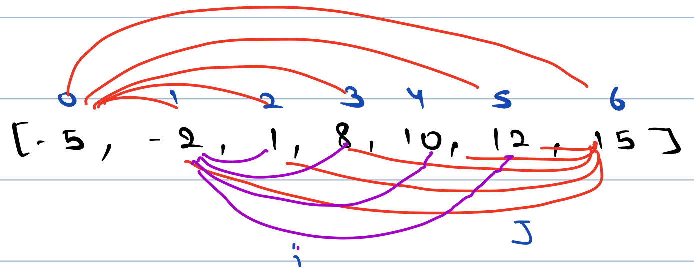
```
i=0, J=n-1
while (i<J) {
        area = Max(area, Min(a[i], a[J]) *
                                      J-i)

            if (A[i] < A[J]) {
                    i++;
            else if (A[i] > A[J]) {
                    J--
            }else {
```

```
    i++ en j--;
}
```

T.C → O(n)

S.C → O(1).



$k = \underline{11}$

$\underline{10 < 11}$

$-2+12 = 10$

$\underline{10 < 11}$