Agenda

1. Pair with given sum - 2
2. Pair with given difference
3. Subarray with sum K
4. Container with most water

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

1. Given an integer sorted array A and an integer k, find any pair (i, j) such that A[i] + A[j] = k and i != j

$$A = [\overset{0}{-5} \quad \overset{1}{-2} \quad \overset{2}{1} \quad \overset{3}{8} \quad \overset{4}{10} \quad \overset{5}{12} \quad \overset{6}{15}] \qquad k = 11$$

Ans	i	j	A[i] + A[j]
True	2	4	1 + 10 = 11

$$A = [\overset{0}{-3} \quad \overset{1}{0} \quad \overset{2}{1} \quad \overset{3}{3} \quad \overset{4}{6} \quad \overset{5}{8} \quad \overset{6}{11} \quad \overset{7}{14} \quad \overset{8}{18} \quad \overset{9}{25}]$$

k = 12

Ans	i	j	A[i] + A[j]
True	2	6	1 + 11 = 12

Approach 1 : Brute Force

Go to all pairs and check for sum = k

```
for (i=0 ; i<n ; i++) {
    for (j= i+1 ; j<n ; j++) {
        if (a[i] + a[j] == k)
            return (i, j) TRUE
    }
}
return (-1, -1) FALSE
```

TC: O(N²)
SC: O(1)

Approach 2 : For every A[i], look for k - A[i] on the right side (i+1 → N-1)

$A[i] + A[j] = K$

$\Rightarrow A[j] = K - A[i]$

```
for (i=0 ; i<n ; i++){
    // partner = K - A[i]
    int f = binary search (partner, i+1, N-1)
    if (f != -1)   (i, f)
}
return  (-1, -1)
```

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

Approach 3 : HashMap / HashSet

TC: O(N)
SC: O(N)

1. Put all elements of array in hashmap <elem, freq>
2. Go to every element A[i] in array, search for K - A[i] in hm

If A[i] == partner then freq / HM[A[i]] > 1

Approach 4 : 2 pointer approach

TC: O(N)

	i	j						
	0	1	2	3	4	5	6	
A =	[-5	-2	1	8	10	12	15]	K = 11

$A[i] + A[j] = -5 + -2 = -7 < 11$

Increase j^{th} ele or i^{th} ele or both

$$
\begin{array}{ccccccccc}
 & & 0 & 1 & 2 & 3 & 4 & \overset{i}{5} & \overset{j}{6} \\
A & = & [-5 & -2 & 1 & 8 & 10 & 12 & 15]
\end{array}
\qquad K=11
$$

$A[i] + A[j] = 12 + 15 = 27 > 11$

Decrease i^{th} ele or j^{th} ele or both

$$
\begin{array}{ccccccccc}
 & & 0 & 1 & \overset{i}{2} & 3 & \overset{j}{4} & 5 & 6 \\
A & = & [-5 & -2 & 1 & 8 & 10 & 12 & 15]
\end{array}
\qquad K=11
$$

i	j	A[i] + A[j]	sum	
0	6	$-5 + 15$	$10 < 11$	Increase sum, i++
			(bcoz j can't be ↑)	
1	6	$-2 + 15$	$13 > 11$	Decrease sum j--

already eliminated ← (bcoz i can't be ↓)

i	j	A[i] + A[j]	sum	
1	5	$-2 + 12$	$10 < 11$	Increase sum i++
2	5	$1 + 12$	$13 > 11$	Decrease sum j--

2 4 1 + 10 11 == 11 break

```
// A[], N

int i=0, j=N-1
while (i<j) {
    if (A[i] + A[j] == k) {
        return true / (i,j)
    }
    else if (A[i] + A[j] < k) {
        i++
    }
    else {
        j--
    }
}
return false / (-1,-1)
```

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

---

2. Count all pairs in a sorted array whose sum is k.

A = [1 2 3 4 5 6 8] k = 10

ans = 2 (2,8) (4,6)

2 pointer Approach

Case 1 : when elements are distinct

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

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

Case 2 : when elements are duplicate

A = [2 3 3 5 7 10 10 10 15]   K=13

ans=6

i
j

A = [2 3 3 5 7 10 10 10 15]   K=13

① A[i] + A[j] = K   (A[i] != A[j])

x=3   y=10

cnt_x=2   cnt_y=3

$\boxed{\text{ans += cnt\_x * cnt\_y}}$

A = [2 3 6 6 6 6 20]   k=12

ans = 6

(2) A[i] + A[j] = k   (A[i] != A[j])

$\boxed{x}$ x x x $\boxed{x}$

cnt_x = j - i + 1

ans += ((cnt_x - 1) * (cnt_x))/2

x x x x   4 x

3 2 1 = 3+2+1 = 6

k = 2x

x x x ...... x .. N x

N-1 N-2 N-3 0

$$\text{sum} = \frac{N-1\,(N)}{2}$$

```
// A[], n

int ans = 0, i = 0, j = N-1
while (i < j) {
    if (A[i] + A[j] == k) {
        if (A[i] != A[j]) {
            int x = A[i], cnt_x = 0
            while (A[i] == x) {
                cnt_x++    i++
            }
            int y = A[j], cnt_y = 0
            while (A[j] == y) {
                cnt_y++    j--
            }
            ans += cnt_x * cnt_y
        }
        else {
            cnt_x = j - i + 1
            ans += (cnt_x - 1) * (cnt_x) / 2
            break
        }
    }
    else if (A[i] + A[j] < k)
        i++
    }
```

else {
  i++, j--
}

return ans

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

---

3. Given an integer sorted array A and an integer k, find any pair (i, j) such that A[j] - A[i] = k, i != j and k > 0.

$$\begin{array}{cccccccccc} & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \\ A & = & [-5 & -2 & 1 & 8 & 10 & 12 & 15] & k = 11 \end{array}$$

ans
true

i	j	A[j] - A[i]
2	5	12 - 1 = 11

$k > 0 \Rightarrow A[j] - A[i] > 0$
$\Rightarrow A[j] > A[i]$

$j > i$

Approach 1: Brute Force

Go to all pairs and check for diff = k

```
for (i=0 ; i<n ; i++) {
    for (j= i+1 ; j<n ; j++) {
        if (a[j] - a[i] == k)
            return (i, j) / true
    }
}
return (-1, -1) / false
```

TC : $O(N^2)$

SC : $O(1)$

$$\begin{array}{ccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \\ A = [& -5 & -2 & 1 & 8 & 10 & 12 & 15 &] \end{array} \quad K = 11$$

$A[j] - A[i] = K$

$\Rightarrow A[j] = K + A[i]$

A[i]	A[j]
-5	6
-2	9

Approach 3 : 2 pointers

$$\begin{array}{ccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & \\ A = [& 1 & 2 & 4 & 5 & 6 & 12 &] \end{array} \quad K = 10$$

ans
true

$(i, j) = 1, 5$

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

$A[j] - A[i] = 15 - (-5)$

$= 20 > 11$

Decrease diff

$\downarrow A[j]$ or $A[i] \uparrow$

$j--$ or $i++$ or both

$b - a$

$\downarrow 8 - 2 \uparrow = 6 \downarrow$

① $5 - 2 = 3$

② $8 - 4 = 4$

$3 < 11$

			i			j		
		0	1	2	3	4	5	6
$A =$	[	-5	-2	1	8	10	12	15]

$k = 11$

i	j	diff $(A[j] - A[i])$	$b - a = \text{diff}$
0	1	$-2 - (-5) = 3 < 11$	$\uparrow b$ or $\downarrow a$ Increase diff $j++$

$\underline{8} - 2 = 6 \uparrow$

increase $A[j]$ ① $10 - 2 = 8$

decrease $A[i]$ ② $8 - 0 = 8$

i	j	diff	
0	2	$1 - (-5) = 6 < 11$	Increase diff $j++$
0	3	$8 - (-5) = 13 > 11$	Decrease diff $b - a = \text{diff} \downarrow$ $i++$

$\downarrow b$ or $\uparrow a$

$\downarrow j$ or $\uparrow i$

1	3	$8-(-2)=10<11$	Increase diff $j{+}{+}$
1	4	$10-(-2)=12>11$	Decrease diff b-a = diff ↓ $i{+}{+}$
2	4	$10-1=9<11$	Increase diff $j{+}{+}$
2	5	$12-1==11$	Break

```
// A[], N
int i=0, j=1
while (j < N){
    if (A[j] - A[i] == k){
        return (i, j) / true
    }
    else if(A[j] - A[i] < k){
        j++
    }
    else {
        i++
    }
}
return (-1,-1) / false
```

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

-8	3	4
	i	j

k=4

all +ve

4. Given an integer array A and an integer k, check if there exists a subarray with sum k.

A = [1 3 15 10 20 3 23] k=33

k = 33 Ans → true [10, 20, 3]

k = 44 Ans → false

Approach 1 : use pf [ ]

k=33

A = [1 3 15 10 20 3 23]

pf = [1 4 19 29 49 52 75]

$\underbrace{19 \quad 29 \quad 49 \quad 52}_{52-19=33}$

① sum (i → j) = k

pf[j] - pf[i-1] = k

Array of all +ve

↓

pf → sorted array

In pf[ ] array, find a pair (i, j) whose difference = k

② sum (0 → j) = k ⇒ pf[j] = k

Check every value in pf[ ] with k

TC : O(N) SC : O(N)

↓

O(1) modify original arr

K = 33

A = [1 3 15 $\boxed{\overset{i}{10} \quad 20 \quad \overset{j}{3}}$ 23]

sum = 1 → 4 → 19 → 29 → 49

48 → 45 → 30 → 33

sum > K
i++

sum < K
j++

```
// A[], N, K
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++
    }
}
```

K=7

$\begin{array}{cc} i & j \\ \boxed{2} & \boxed{4} \end{array}$

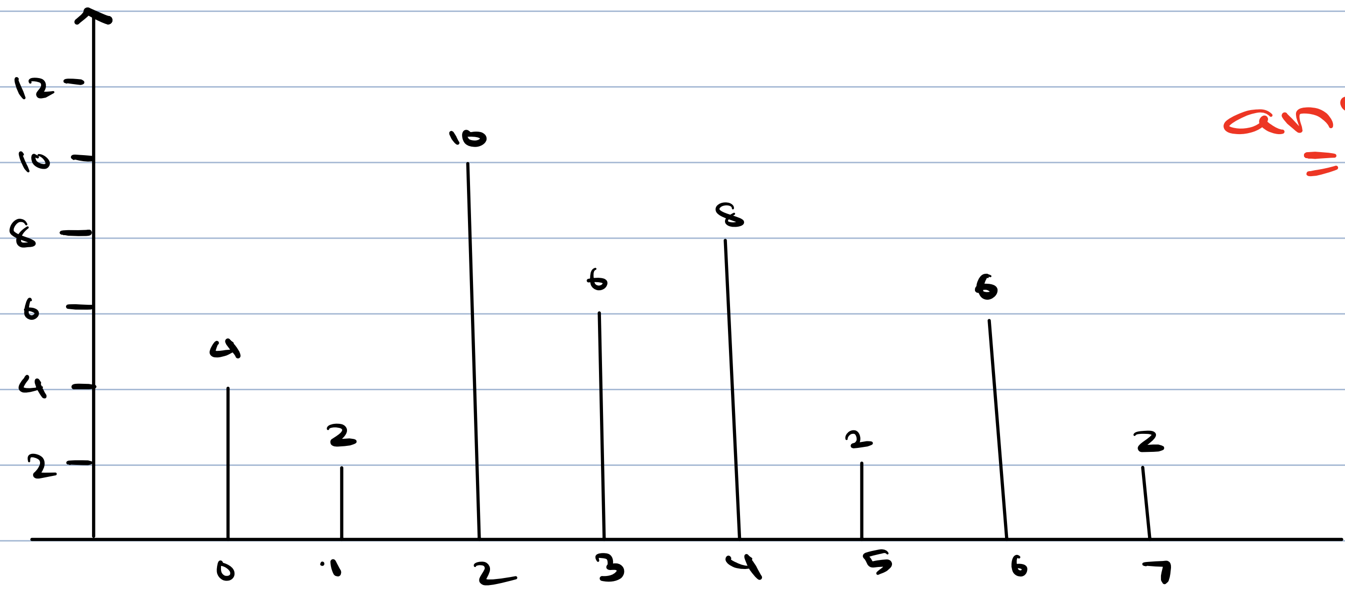
K= 20

$\begin{array}{ccc} j\ i & & \\ \boxed{30} & \boxed{\ } & \boxed{\ } \end{array}$

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

5. Given an integer array A where array element represents height of the wall. Find two walls that can form a container to store maximum water.

A = [4 2 10 6 8 2 6 2]



ans = 24

① water b/w wall 1 and 4

width = 3
height = min (2, 8) = 2
area = 2 × 3 = 6

② water b/w wall 4 and 6

width = 2
height = min (8, 6) = 6
area = 16

③ water b/w wall 2 and 6

width = 4

height = min (10, 6) = 6

area = 4 × 6 = 24

water b/w walls L and R

area = (R − L) × min (A[L], A[R])

		i j					
0	1	2	3	4	5	6	7
A = [4	2	10	6	8	2	6	2]

ans = ~~0~~ ~~14~~ 24

i	j	h	w	water	
0	7	2	7	14	A[i] > A[j] j--
0	6	4	6	24	A[i] < A[j] i++
1	6	2	5	10	A[i] < A[j] i++
2	6	6	4	24	A[i] > A[j] j--
2	5	2	3	6	A[i] > A[j] j--
2	4	8	2	16	A[i] > A[j]

j--

2 3 6 1 6 A[i] > A[j]

j--

2 2 i == j STOP

// A[], N

```
int i=0, j=N-1
int ans=0
while (i<j) {
    water = min (A[i], A[j]) * (j-i)
    ans = max (ans, water)
    if (A[i] < A[j])
        i++
    else
        j--
}
```

TC: O(N)

SC: O(1)

Doubts

ans = ~~0~~ ~~14~~ 24

$$\begin{array}{cccccccccc} & & & i & & & & & j & \\ & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A & = & [4 & 2 & 10 & 6 & 8 & 2 & 6 & 2] \end{array}$$

i	j	h	w	water	move ?
0	7	2	7	14	A[i] > A[j] j--
0	6	4	6	24	A[i] < A[j] i++

k = 19

$$\begin{array}{ccccccccc} & & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A & = & [1 & 3 & 15 & 10 & 20 & 3 & 23] \end{array}$$

pf = 1 4 19 29 49 52 75

sum(0 → 2) = pf[2]

sum(0 → x) = pf[x]

sum(2 → 5) = pf[5] − pf[1]

sum(i → j) = pf[j] − pf[i−1]

pf = [1 4 19 29 49 52 75]