# Two Pointers

## Agenda

- lot of questions

## Question 1

Given a sorted array A & integer K. Find any pair (i, j) s.t. A[i] + A[j] = K & i != j.

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

ans = (2, 4)

1. Bruteforce → TC = $O(N^2)$  SC = $O(1)$

2. Hashing → $A[j] = K - A[i]$

   ∀i, check if $K - A[i]$ is present.

   TC = $O(N)$  SC = $O(N)$ → $O(1)$

3. Binary Search (sorted)

   ∀i, check if $K - A[i]$ is present at index != i

   TC = $O(N \log N)$  SC = $O(1)$

## 4. Two Pointers

→ where to place 2 pointers
  - → same corner
  - → different corner

→ how to update

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

$K = 11$

[same corner]

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

$-5 + (-2) = -7 < K$ → increase $i$ or $j$
not defined

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

$K = 11$

[different corner]

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

$-5 + 15 = 10 < K$ ⇒ increase (only option is $i$++)
defined

$-2 + 15 = 13 > K$

$-2 + 12 = 10 < K$

$1 + 12 = 13 > K$

$1 + 10 = 11 = K$

$$\boxed{\begin{array}{l} -5 + \text{largest } A[i] < K \\ \Rightarrow -5 + \text{any } A[i] < K \end{array}}$$

```
i=0 , j=n-1

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

i != j holds

TC = O(N)

SC = O(1)

## Question 2

Count the number of pairs (i,j) s.t. A[i] + A[j] = K & i != j (sorted)

2 cases :
1. Distinct Array
2. Duplicates allowed

1. Distinct A[i]

$$\begin{array}{ccccccccc} & & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = & [ & 1 & 2 & 3 & 4 & 5 & 6 & 8 ] \\ & & \cancel{i} & \cancel{i} & \cancel{i} & \cancel{i} & i,j & \cancel{j} & \cancel{j} \end{array}$$

K = 10

1+8 < 10 ⇒ i++

2+8 = 10 ⇒ cnt++ , i++ , j--

3+6 < 10 ⇒ i++

4+6 = 10 ⇒ cnt++ , i++, j--

ans = 2

```
i=0 , j=n-1 , cnt=0

while (i<j) {
    if (A[i] + A[j] == K) {
        cnt++ , i++ , j--
    }
    if (A[i] + A[j] < K)   i++
    else   j--
}
return cnt
```

TC = O(N)

SC = O(1)

2. Duplicate A[i]

$A = [\underset{0}{1}\ \underset{1}{2}\ \underset{2}{2}\ \underset{3}{4}\ \underset{4}{8}\ \underset{5}{8}\ \underset{6}{8}]$ K = 10

1+8 < 10 ⇒ i++

2+8 = 10 ⇒ cnt += 6

2, 3 ⇒ # pairs = 2×3 = 6

$K=10$

$$A = [\overset{0}{1}\quad \overset{1}{2}\quad \overset{2}{5}\quad \overset{3}{5}\quad \overset{4}{5}\quad \overset{5}{5}\quad \overset{6}{8}]$$

$(2,3)\ (2,4)\ (2,5)$

$(3,4)\ (3,5)$

$(4,5)$

$$\#\text{pairs} = {}^{4}C_{2} = \frac{4\times 3}{2} = 6$$

```
cnt = 0
i = 0 , j = n-1

while (i<j) {
    if (A[i] + A[j] < K)        i++
    else if (A[i] + A[j] > K)   j--

    else {   // A[i] + A[j] = K
        if (A[i] == A[j]) {
            c = j-i+1
            cnt += c*(c-1)/2
            return cnt
        }
        else {   // A[i] < A[j]
            c1 = 1
            while (A[i] == A[i+1]) {
                c1++ , i++
            }
```

$TC = O(N)$

$SC = O(1)$

```
c2 = 1
while (A[j] == A[j-1]) {
    c2++ , j--
}
cnt += c1 * c2
i++, j--
}
}
}
```

return cnt

K=10

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 3 | 5 | 5 | 7 | 8 | 8 | 8 |
| ~~i~~ | ~~i~~ | ~~i~~ | ~~i~~ | ~~i~~ | i | j | ~~j~~ | ~~j~~ | ~~j~~ | ~~j~~ |

$1+8 < 10 \Rightarrow i{+}{+}$

$2+8 = 10$ , $2 \ne 8$

$c_1 = \cancel{1}\ 2$ $\quad c_2 = \cancel{1}\ \cancel{2}\ 3$

$\Rightarrow cnt += 2\times 3 = 6$

$3+7 = 10$ $\quad 3 \ne 7$

$c_1 = \cancel{1}\ 2$ $\quad c_2 = 1$

$\Rightarrow cnt += 2\times 1 = 2$

$5+5 = 10$ $\quad 5 = 5$

$c = 6-5+1 = 2$

$cnt += \frac{2\times 1}{2} = 1$

$\boxed{cnt = 6+2+1 = 9}$

## Question 3

Given a sorted integer array A & integer K.
find any pair (i,j) s.t. A[j] - A[i] = K & i != j
& K > 0.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|

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

A[5] - A[2] = 12 - 1 = 11   ans = (2,5)

Brute force   Hashing   Binary search ✓

Two pointers

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|

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

~~i~~ ~~j~~ ~~j~~ ~~j~~ ~~j~~ j

~~i~~ i

A[j] - A[i]

-2 - (-5) = 3 < K ⇒ increase

1 - (-5) = 6 < K

8 - (-5) = 13 > K ⇒ either j-- or i++

not making sense

8 - (-2) = 10 < K

A[j] - A[i]

| increase | decrease |
|---|---|
| j++ | i-- |

$10 - (-2) = 12 > K$

$10 - 1 = 9 < K$

$12 - 1 = 11 = K$

$\boxed{\begin{array}{l} A[j] - \text{smallest } A[i] < K \\ \Rightarrow A[j] - \text{any } A[i] < K \end{array}}$

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

TC = O(N)
SC = O(1)

## Question 5

Given an array A & integer K

$\forall i \quad A[i] > 0$

check if there exist a subarray with sum = k.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| A = | 1 | 3 | 15 | 10 | 20 | 3 | 23 |

K = 33

ans = true

Bruteforce → ∀ subarray check sum = K

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

Prefix Sum

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| A = | 1 | 3 | 15 | 10 | 20 | 3 | 23 |

K = 33

pf = [ 1 4 19 29 49 52 75 ] K = 33

↳ always increasing ⇒ sorted

$pf[j] - pf[i] = K$ or $pf[j] = K$

↳ like Question 3

$TC = O(N)$ $SC = O(N)$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| A = | 1 | 3 | 15 | 10 | 20 | 3 | 23 |
| | i | j | j | j | j | j | |
| | j | i | i | i | | | |

K = 33

Carry forward

sum = 1 < K ⇒ add more elements → ≥0

= 4 < K

19 < K

29 < K

49 > K ⇒ remove elements

48 > K

45 > K

30 < K

33 = K

```
i=0, j=0 , sum = A[0]

while ( j<n) {
    if (sum == K ) {
        return true
    }
    if (sum < K ) {
        j++        // j = n
        if (j==n)  return false
        sum += A[j]
                  → index out of bound
    }
    else {
        sum -= A[i]
        i++
    }
}
return false
```

A= [ 2  6  9 ]    K = 3

~~i~~ ~~j~~ ~~i~~ i

~~j~~ ~~i~~ ~~j~~ j → return false

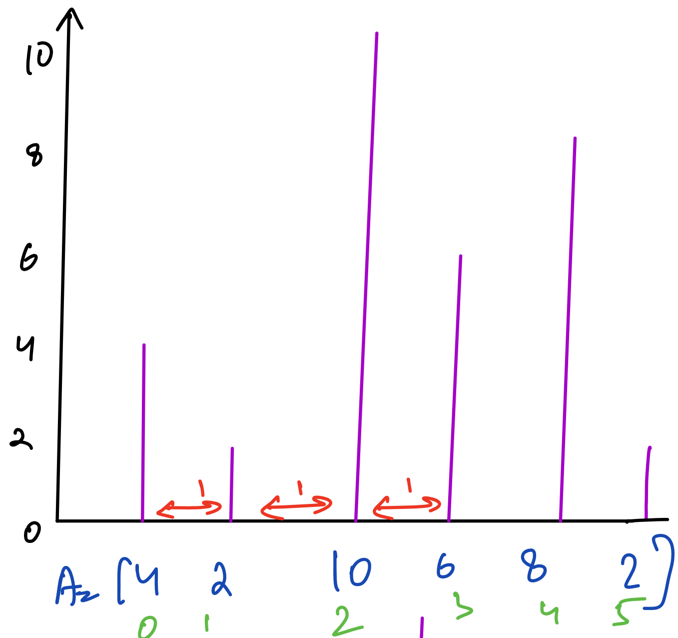
sum = ~~2~~ ~~8~~ ~~6~~ 0 ~~9~~ 0

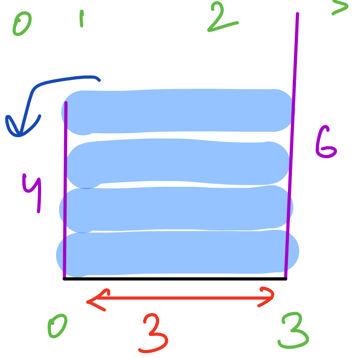
## Question 5

Given an array where every a[i] represents height of $i^{th}$ wall. Find any 2 walls that can form a container to store the maximum water.

area    A[i] → height

every adj. wall is 1 unit away




$H = \min(4, 6) = 4$

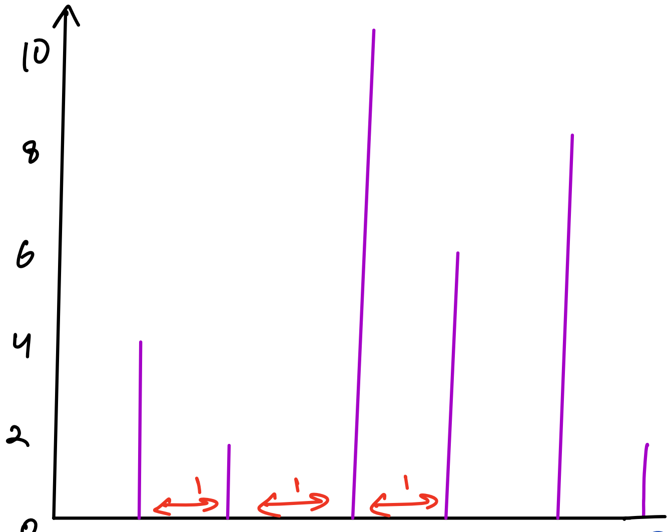
$\text{area} = 3 \times 4 = 12$

Bruteforce → $\forall i, j$ s.t. $i < j$

calculate area & take max

$(j - i) * \min(A[i], A[j])$

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




A = [4 2 10 6 8 2]

0 1 2 3 4 5

~~i~~ ~~i~~ i j ~~j~~ ~~j~~

area | ans
--- | ---
(5-0) * min(4,2) = 5×2 = 10 | 10
(4-0) * min(4,8) = 4×4 = 16 | 16
(4-1) * min(2,8) = 3×2 = 6 |
(4-2) * min(10,8) = 2×8 = 16 |
(3-2) * min(10,6) = 1×6 = 6 |

(i++ or j--) ⇒ (j-i) will decrease by 1

A[i] > A[j] ⇒ j--

A[i] < A[j] ⇒ i++

A[i] == A[j] ⇒ i++, j--

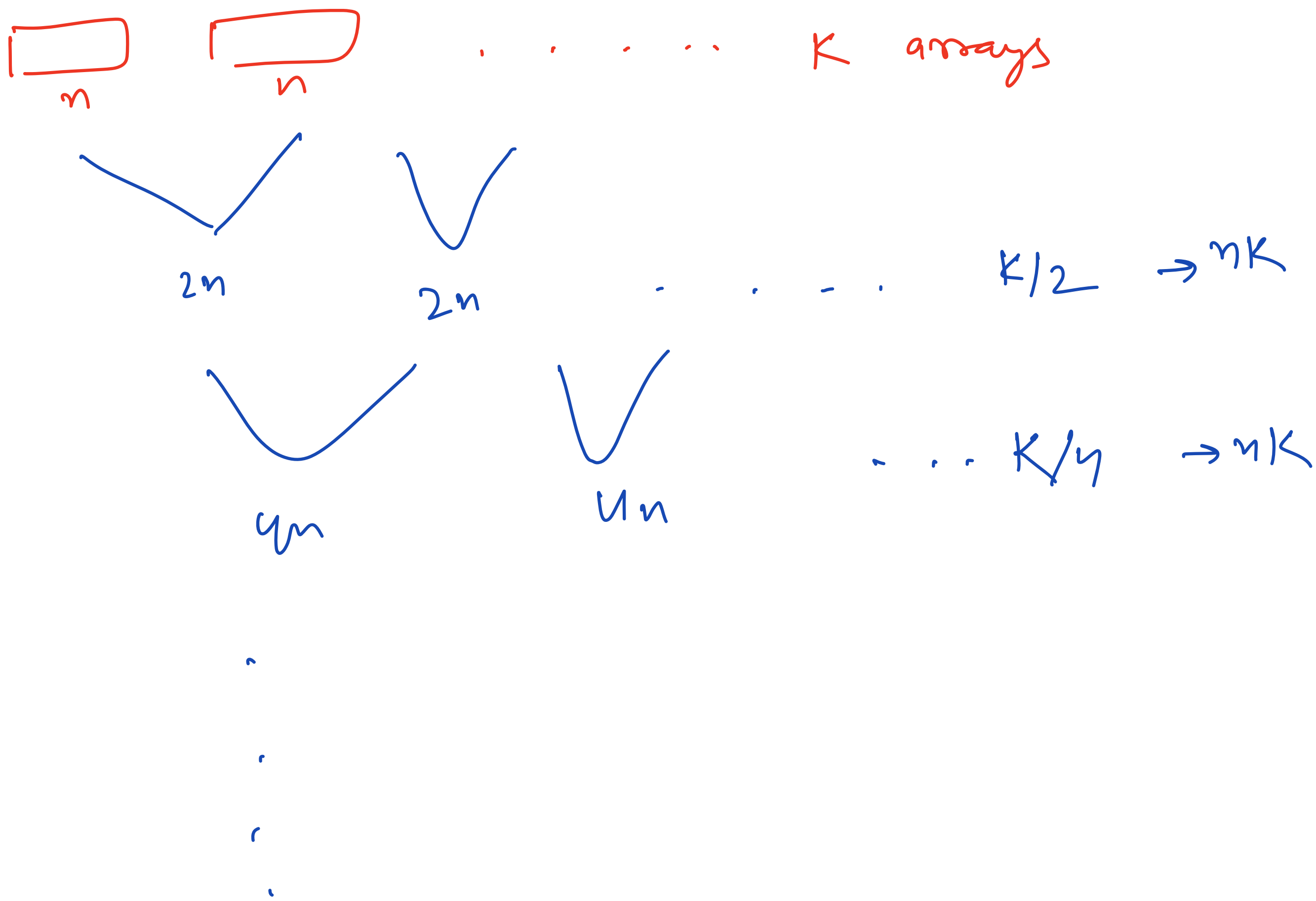
```
i=0 , j=n-1 , ans=0
while (i<j) {
    area = (j-i) * min( A[i], A[j])
    ans = max( ans , area)
    if (A[i] < A[j])  i++
    else if ( A[i] > A[j])  j--
    else {  i++, j--  }
}
```

TC = O(N)

SC = O(1)

return ans

## Doubt




$TC = O(nk \log k)$

$SC = O(nk)$