Agenda

1) Pair sum
2) Pair difference
3) container with most water

Q.1 Pair sum

Given a sorted array, check if there exists a pair (i,j) such that A[i] + A[j] = k (i != j)

A = [3 7 8 12 19] k = 19 ans = true

A = [2 5 8 9 10] k = 9 ans = false

i) go on all the pairs, TC: $O(n^2)$ SC: $O(n)$

ii) Hashset, TC: $O(n)$ SC: $O(n)$

iii) Binary search, TC: $O(n \log n)$ SC: $O(1)$

(iv) Two pointers

optimal approach

$$A = \begin{array}{cccccccccc} [-3 & 0 & 1 & 3 & 6 & 8 & 11 & 14 & 18 & 25] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ & & & i & & & & j & & \end{array} \qquad K = 17$$

A[i]	A[j]	A[i] + A[j]		
-3	25	-3 + 25	22 > K	j--
-3	18	-3 + 18	15 < K	i++
0	18	0 + 18	18 > K	j--
0	14	0 + 14	14 < K	i++
1	14	1 + 14	15 < K	i++
3	14	3 + 14	17 == K	

$A[i] + A[j] == K$ => return true

$\underbrace{A[i] + A[j]}_{\downarrow} > K$ => j--

$\underbrace{A[i] + A[j]}_{\uparrow} < K$ => i++

A = [-3 0 1 3 6 8 11 14 18 25]   K = 17

0 1 2 3 4 5 6 7 8 9

(0,9)
~~(1,9)~~
~~(2,9)~~
~~(3,9)~~
~~(4,9)~~
~~(5,9)~~
~~(6,9)~~
~~(7,9)~~
~~(8,9)~~

(0,8)
~~(0,7)~~
~~(0,6)~~
~~(0,5)~~
~~(0,4)~~
~~(0,3)~~
~~(0,2)~~
~~(0,1)~~

-3 + 25 = 22   22 > K   j-- ✓

-3 + 18 = 15   15 < K   i++ ✓

```
boolean Pairsum ( int [] A, int K ) {

    int  i = 0 , j = A.length - 1;

    while ( i < j ) {

        if (A[i] + A[j] == K ) {
                return true;
        }
        else if (A[i] + A[j] > K ) {
                j--;

        }
        else if (A[i] + A[j] < K ) {
                i++;
        }
    }
    return false;
}
```

TC: O(n)

SC: O(1)

Dry run

```
boolean Pairsum ( int [] A, int K ) {
    int i = 0 , j = A.length-1;
    while ( i < j ) {
        if (A[i] + A[j] == K ) {
            return true;
        }
        else if (A[i] + A[j] > K ) {
            j--;
        }
        else if (A[i] + A[j] < K) {
            i++;
        }
    }
    return false;
}
```

K = 9

A = [-3 0 1 3 6 8 11 14 18]

-3	0	1	3	6	8	11	14	18
0	1	2	3	4	5	6	7	8
		i			j			

A[i]	A[j]	A[i] + A[j]	
-3	18	15 > 9	j--
-3	14	11 > 9	j--
-3	11	8 < 9	i++
0	11	11 > 9	j--
0	8	8 < 9	i++
1	8	9 == 9	

Q.2 Pair difference

Given a sorted array, check if there exists a pair (i, j) such that A[j] - A[i] = k and k > 0 (i != j)

A = [-3 0 1 3 6 8 11 14 18 25] k = 5 ans = true

-3	0	1	3	6	8	11	14	18	25
0	1	2	3	4	5	6	7	8	9

i) Brute force : going on all pairs TC → $O(n^2)$, SC → O(n)

ii) Hashset TC → O(n) SC → O(n)

iii) can you discard some invalid pairs by two pointer strategy

A = [-3 0 1 3 6 8 11 14 18 25] k = 5

-3	0	1	3	6	8	11	14	18	25
0	1	2	3	4	5	6	7	8	9
i									j

A[j] - A[i] = 25 - (-3) = 28 ✗

j-- : diff will decrease

i++ : diff will decrease

A =	[-3	0	1	3	6	8	11	14	18	25]	K = 5
	0	1	2	3	4	5	6	7	8	9	
			i		j						

A[i]	A[j]	A[j] - A[i]		
-3	0	3	3 < K	j++
-3	1	4	4 < K	j++
-3	3	6	6 > K	i++
0	3	3	3 < K	j++
0	6	6	6 > K	i++
1	6	5	5 == 5	

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

$\underbrace{A[j] - A[i]}_{\downarrow} > K \Rightarrow$ i++

$\underbrace{A[j] - A[i]}_{\uparrow} < K \Rightarrow$ j++

A = [-3 0 1 3 6 8 11 14 18 25]   k = 4

0 1 2 3 4 5 6 7 8 9

(2, 4)

A[4] - A[2] = 6 - 1 = 5 , 5 > k, i++

~~(2, 5)~~

~~(2, 6)~~

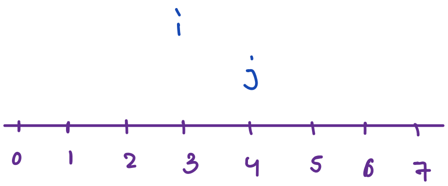
~~(2, 7)~~

~~(2, 8)~~

~~(2, 9)~~

```
boolean PairDiff (int [] A, int k) {
    int i=0, j=1;
    while (j < A.length) {
        if (A[j] - A[i] == k) {
            return true;
        }
        else if (A[j] - A[i] > k) {
            i++;
        }
        else if (A[j] - A[i] < k) {
            j++;
        }
    }
    return false;
}
```

i is always <= j



k > 0

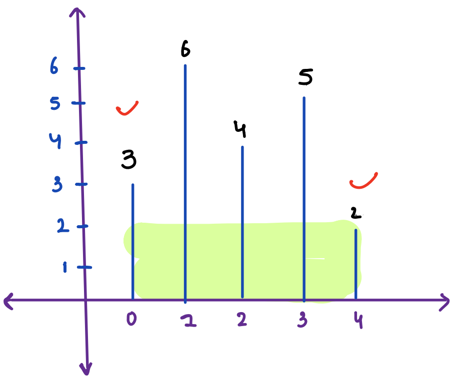
TC : O(n)

SC : O(1)

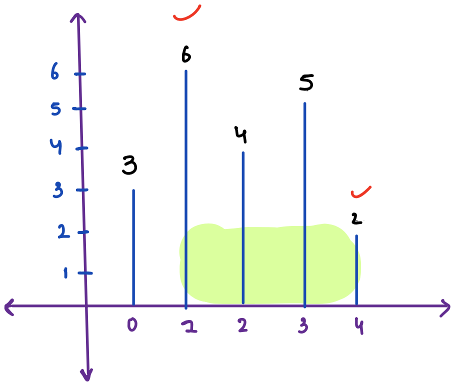
Q.3 Container with most water

Given an array, where A[i] represents height of each wall. Pick any 2 walls such that max water is accumulated b/w them.

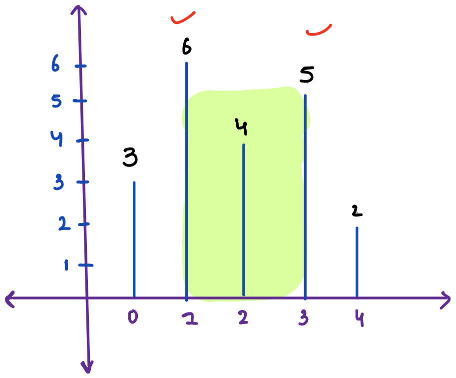
A = [3 6 4 5 2]



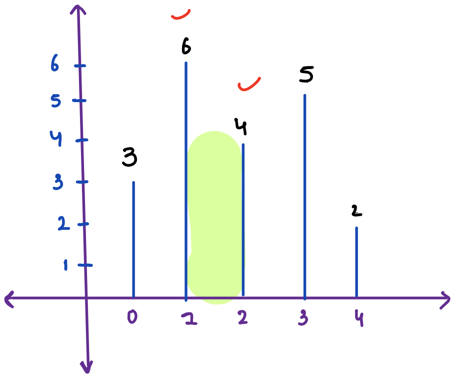
$2 * 4 = 8$



$2 * 3 = 6$



$2 * 5 = 10$



$1 * 4 = 4$

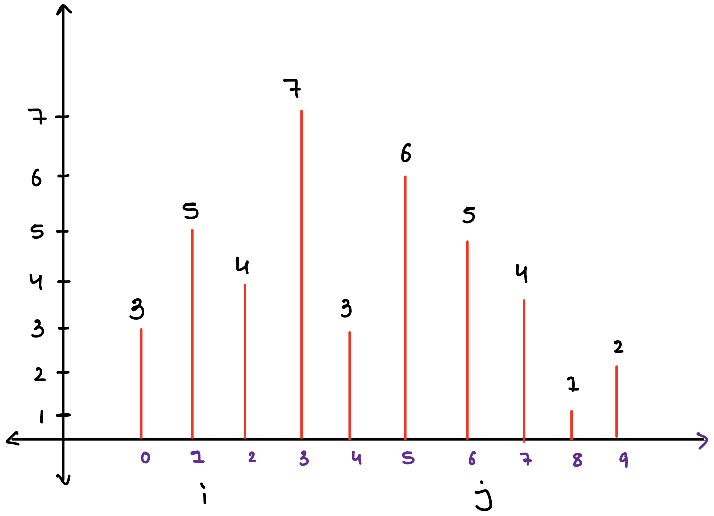
i) go on all pairs, TC: $o(n^2)$ SC: $o(1)$

(i, j)

amount of water = $\underbrace{\min(A[i], A[j])}_{h} * \underbrace{(j-i)}_{w}$

optimal solution

A = [3 5 4 7 3 6 5 4 1 2]

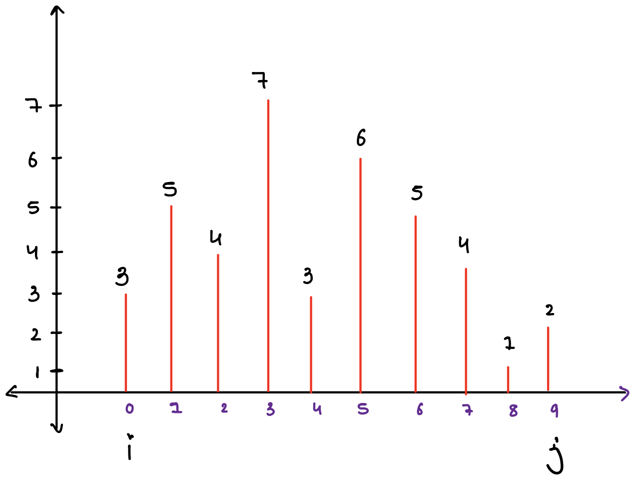


ans= ~~0~~ ~~18~~
~~24~~ ~~21~~
25

A[i]	A[j]	h	w	temp	action
3	2	2	9	18	j--
3	1	1	8	8	j--
3	4	3	7	21	i++
5	4	4	6	24	j--
5	5	5	5	25	......

(discard lower ht)

A = [3 5 4 7 3 6 5 4 1 2]



(0,9)

~~(1,9)~~

~~(2,9)~~

~~(3,9)~~

~~(4,9)~~

~~(5,9)~~

~~(6,9)~~

~~(7,9)~~

~~(8,9)~~

Why we discard lower ht

$$temp = \underbrace{\min(A[i], A[j])}_{\text{either same or less}} * \underbrace{(j-i)}_{\text{always reducing}}$$

```
int  cwmw ( int [] A) {

   int  i=0 , j= A.length-1 ;
   int  ans=0 ;

   while (i < j) {

        int  temp =  min (A[i], A[j]) * (j-i);
        if ( temp > ans) {
             ans= temp;
        }

        // discard lower height
        if (A[i] < A[j]) {
             i++ ;
        }
        else {
             j-- ;
        }
   }

   return ans;
}
```

TC : O(n)

SC : O(1)

```
while (i < j) {
    int temp = min (A[i], A[j]) * (j-i);
    if ( temp > ans) {
        ans = temp;
    }

    // discard lower height
    if (A[i] < A[j]) {
        i++;
    }
    else {
        j--;
    }
}
```

A = [3 6 4 5 2]

0 1 2 3 4

i

j

ans = ~~0~~ ~~8~~ ~~9~~

10

A[i]	A[j]	h	w	temp	action
3	2	2	4	8	j--
3	5	3	3	9	i++
6	5	5	2	10	j--
6	4	4	1	4	j--

A^{th} magical no.

magical no. → a no. div. by either B or C

A = 5   B = 2   C = 3

1	2	3	4	5	6	7	8
X	1	2	3	X	4	X	5

Easy problem

find total no. of magical no. <= X (a particular value)

B = 3   C = 5   X = 20

3   5   6   9   10   12   15   18   20

count of multiples of B (<= X) => $\frac{X}{B}$

count of multiples of C (<= X) => $\frac{X}{C}$

count of common multiples => $\frac{X}{B*C}$

$$\text{count} = \frac{X}{B} + \frac{X}{C} - \frac{X}{\text{LCM}(B,C)}$$

$$\frac{20}{3} + \frac{20}{5} - \frac{20}{15}$$

$$\Rightarrow 6 + 4 - 1$$

$$= 9$$

$B = 2$ $C = 4$ $X = 10$

2 4 6 8 10

$$\text{count} = \frac{10}{2} + \frac{10}{4} - \frac{10}{4}$$

$$= 5 + 2 - 2 = 5$$

$B = 3$ $C = 6$ $X = 20$

3 6 9 12 15 18

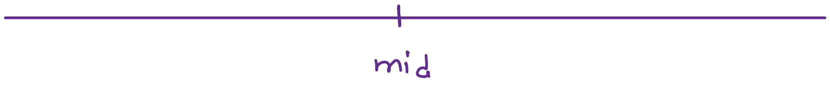
$$\text{count} = \frac{20}{3} + \frac{20}{6} - \frac{20}{6}$$

$$= 6 + 3 - 3 = 6$$

Idea of Binary search

Ath magical no.

5th magical no.



=> 30

count magical no. <= 30

↳ 9

B = 3   C = 5   10th magical no.

lo = min(B, C)   hi = max(B, C) * A

(1st magical no.)   (at max of Ath magical no.)

ans => lo to hi