Next Fri
↓
17 Nov

Pairs with given sum - 2
Pairs with given difference
Subarray with given sum
Container with most water

Ass/HW
↓
Notes
↓
Recording

Reattempt date

Contest 3 Nov

Advance DSA : Contest 2

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

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

0	1	2	3	4	5	6
-5	-2	1	8	10	12	15

k = 11

i = 2, j = 4, k = 11

A[2] + A[4] = 11

pair with sum = k

BF : Consider all pairs (i, j)

(upper or lower Δ)

```
for (i=0 ; i<n ; i++) {
    for (j = i+1 ; j<n ; j++) {
        if (A[i] + A[j] == k)
            return i, j
```

TC : O(N²)

SC : O(1)

0	1	2	3	4	5	6
-5	-2	1	8	10	12	15

i

k = 11

-2 + ___ = 11

i	j
0	1 → 6
1	2 → 6

A[i] + A[j] = k

-5 + A[j] = 11

A[j] = 11 - (-5)

= 16

i	j	
0	1 → 6	search for 16 [11-(-5)]
1	2 → 6	search for 13 [ ]

A[i] + A[j] = K

-2 + A[j] 11

A[j] = 11-(-2)

2 3 → 6 search for 10

A[j] = K - A[i]

A[i] + A[j] = K

```
for (i=0 ; i<n ; i++) {
    // search for K-A[i]
       in range [i+1, n-1]
}
```

TC : O(n log n)

SC : O(1)

Approach 3 : 2 pointers

Directly eliminate some pairs

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

i j

A[i] + A[j]

A[0] + A[1]

-5 + -2

-7 < 11

x+y < K

↑x or ↑y

X i++ or j++

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

0 1 2 3 4 5 6

i j

↓A[i] or ↓A[j]

i-- or j--

12 + 15 = 27 > 11

x + y > k

↓x or ↓y

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

i j

x + y < k

↑x or ↑y

A[i]	A[j]	sum	
-5	15	10 < 11	i++
-2	15	13 > 11	j--
-2	12	10 < 11	i++
1	12	13 > 11	j--
1	10	11 = K	✓

-5 + biggest de < k

-5 + any de x

> x + y > k

↓x or ↓y

i = 0, j = N-1

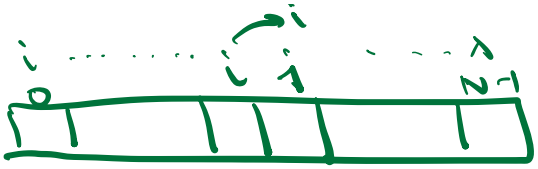
```
while ( i < j ) {
    if (A[i] + A[j] == k)
        return (i, j)
    else if (A[i] + A[j] < k)
        i++
    else
        j--
}
```

curr smallest de + 15 > 11

+15

TC: O(N)

SC: O(1)



Find all pairs in distinct array.

1	2	3	4	5	6	8	K = 10
				i			4,6
				j			2,8

cnt = 0 // 2

```
if (arr[i] + arr[j] == k) {
    i++
    j--
    cnt++
}
```

i = 0, j = N-1 cnt = 0

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

→ SC: O(1)

For duplicates TC: O(N) SC: O(N)

① Freq array

1 3 3 10 10 10 15 K = 13

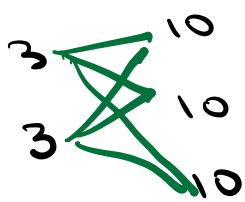
3 + 10 = 13

	i	j
Ar →	1 3	10 15
freq	1 2	3 1

elem: freq 6 pairs

1:1
3:2
10:3
15:1

No. of pairs
= freq[A[i]] * freq[A[j]]

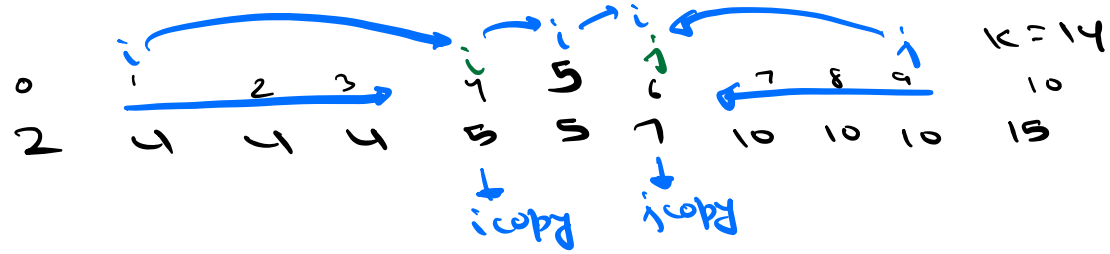


6 pairs

Create a distinct array and store freq of every element. Whenever we find a pair where sum = k, cnt += freq[A[i]] * freq[A[j]]

i++ j--

② without freq array



cnt = 0 9

count ith ele

```
int icopy = i
while (A[icopy] == A[i]) {
    icopy++
}
ele = icopy - i
```

```
cnt = 0    i = 0    j = n-1
while (i < j) {
    if (A[i] + A[j] == k) {
        if (A[i] == A[j]) {
            freq = j - i + 1
            cnt += (freq * (freq - 1)) / 2
            break
        }
        icopy = i        jcopy = j
        while (A[icopy] == A[i]) {
            icopy++
        }
        iele = icopy - i
        while (A[jcopy] == A[j]) {
            jcopy--
        }
        jele = j - jcopy
        count += iele * jele
```

```
        i = icopy
        j = jcopy
    }
    else if (A[i] + A[j] < k)
        i++
    else
        j--
}
```

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

$^4C_2$

$\frac{n \times (n-1)}{2}$

k = 2

6 pairs

1 1 1 1

↓ ↓ ↓ ↓

3 2 1 0

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

k > 0

```
          0   1   2   3   4   5   6
A = [-5, -2,  1,  8, 10, 12, 15]      k = 11
```

```
   i   j
   2   5
ek 1   12   12-1=11
```

pair with diff = k

A[j] - A[i] = k > 0

A[j] - A[i] > 0

A[j] > A[i]

j is always on right of i

BF : check all idx pairs

i → 0 to N-1

j → i+1 to N-1

TC : $O(N^2)$

SC : $O(1)$

```
          0   1   2   3   4   5   6
A = [-5, -2,  1,  8, 10, 12, 15]      k = 11
```

```
i   j
0   1 to 6
    search for 6

1   2 to 6
    search for 9
```

A[j] - A[i] = k

A[j] - (-5) = 11

A[j] = 11 - 5

= 6

A[j] = k + A[i]

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

Approach 3: 2 pointers

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

A[j] - A[i]

15 - (-5) = 20 > 11

y - x > K

↓y or ↑x

X j-- or i++

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

15 - 12 = 3 < 11

y - x < K

largest - A[i] < K

etc

any de - A[i] < K

↑y or ↓x

↓

i--

A[j] - A[i] = K

↓

j++

↑

A[j]

X

↓

- A[i] < K

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

K = 11

A[j]	A[i]	Diff	
-2	-5	-2-(-5) = 3 < 11	j++
1	-5	1-(-5) = 6 < 11	j++
8	-5	8-(-5) = 13 > 11	i++
8	-2	8-(-2) = 10 < 11	j++
10	-2	10-(-2) = 12 > 11	i++
10	1	10-1 = 9 < 11	j++
12	1	12-1 = (11) ✓	

i=0 j=1

-2 - smallest < k

while (j < N) {

-2 - 1

diff = A[j] - A[i]

if (diff == k)

return (i, j)

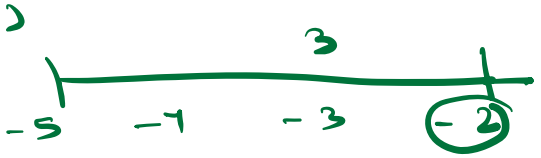
else if (diff < k)

j++

else

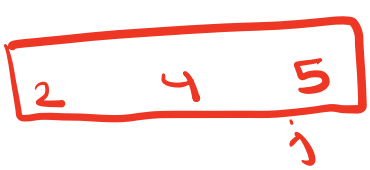
i++

}



TC : O(N)

SC : O(1)



K = 1

10:45

break

3. 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]
```

N = 7

$\frac{7*8}{2} = 28$

s = 3, e = 5

k = 33 → True

k = 43 → False

BF: Check all subarray sums

→ cnt → $\frac{N(N+1)}{2}$

Using carry forward

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

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

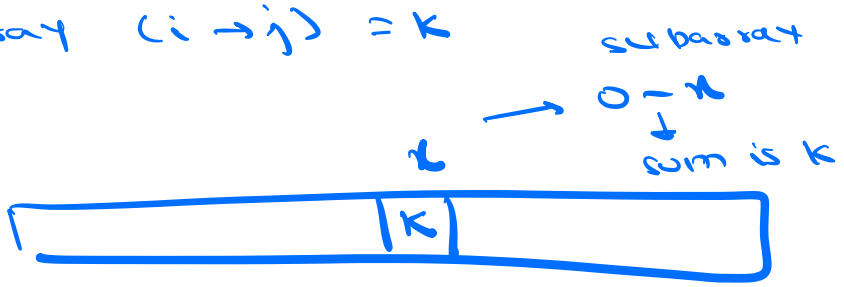
pf = [ ]

Prefix sum

sum [i → j] = pf [j] − pf [i−1] i != 0

sum [0 → j] = pf [j] = k i = 0

sum of subarray (i → j) = k



sum [i → j] = pf[j] − pf[i−1] = k

① ∀ i → check pf[i] = k if it is k

subarray (0 − i) → sum = k

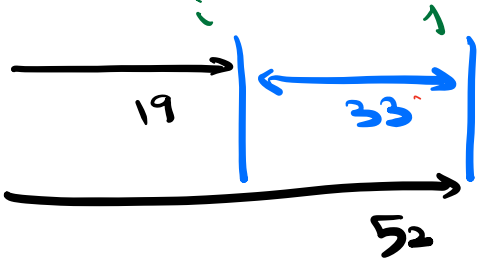
② sum [i → j] = pf[j] − pf[i−1] = k

Find a pair in pf[] whose diff = k

A = [1, 3, 15, 10, 20, 3, 23] K = 33

pf = [1 4 19 29 49 52 75]

Prefix array



TC: O(N)

SC: O(N)

↓

O(1)

modify same arr to store pf[]

Approach 3 : Dynamic sliding window Approach

A = [1, 3, 15, 10, 20, 3, 23] K = 33

0 1 2 3 4 5 6

i j

i	j			
0	0	1		j++
0	1	4 < 33	1-1	j++
0	2	19 < 33	1-2	j++
0	3	29 < 33	1-3	j++
0	4	49 > 33		i++

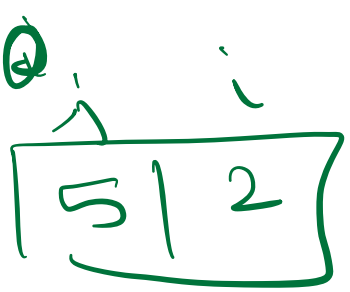
sum = X X 19 29 ~~49~~ ~~45~~ ~~45~~ 30 33

1	4	48 > 33	i++
2	4	45 > 33	i++
3	4	30 < 33	j++
3	5	33 = 33	✓

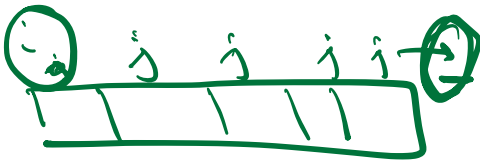
i → start
j → end

K = 2

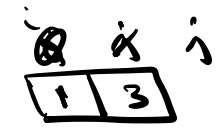
sum = ~~5~~0



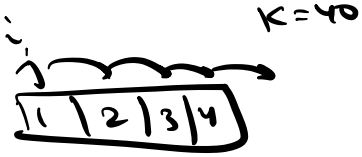
i=0 j=0 sum = A[0]

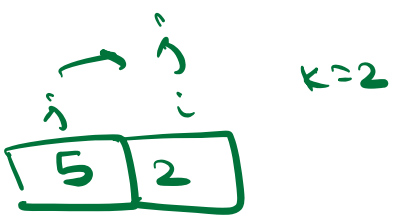


```
while ( j < N ) {
    if (sum == k)
        return true
    else if (sum < k) {
        j++
        if (j == n) break
        sum += A[j]
    }
    else if (sum > k) {
        sum -= A[i]
        i++
        if (i > j && i <= n-1) {
            j++
            sum += A[j]
        }
    }
}
```



sum=4

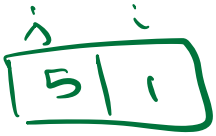




TC: O(N)

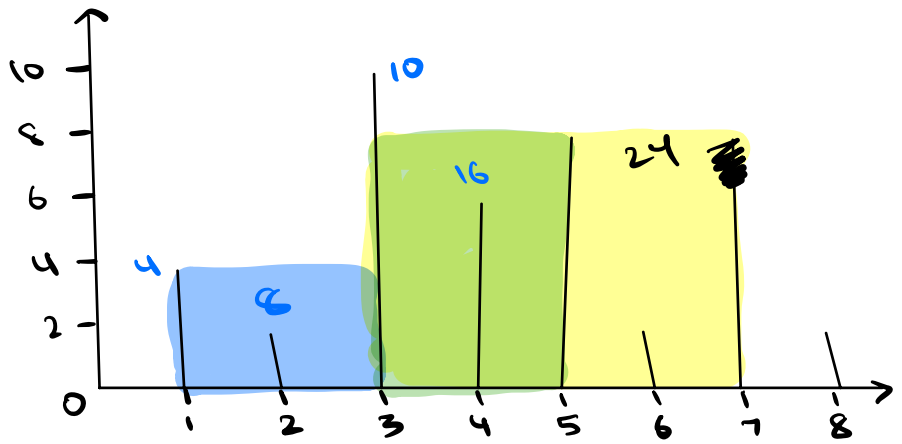
SC: O(1)

sum=5 2



sum=5 0

4. Return max water that can be stored b/w any 2 walls



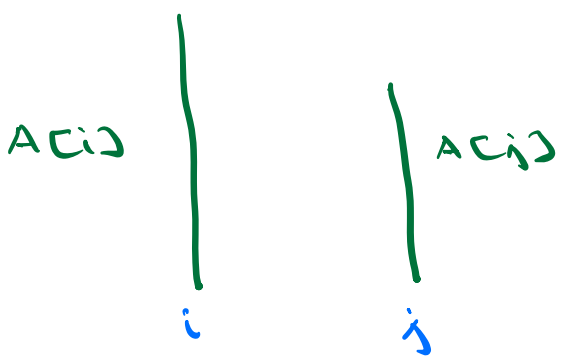
A[] = [4 2 10 6 8 2 6 2]

4 * 2 = 8 units

8 * 2 = 16 units

idx 3 & 7

ans = 24

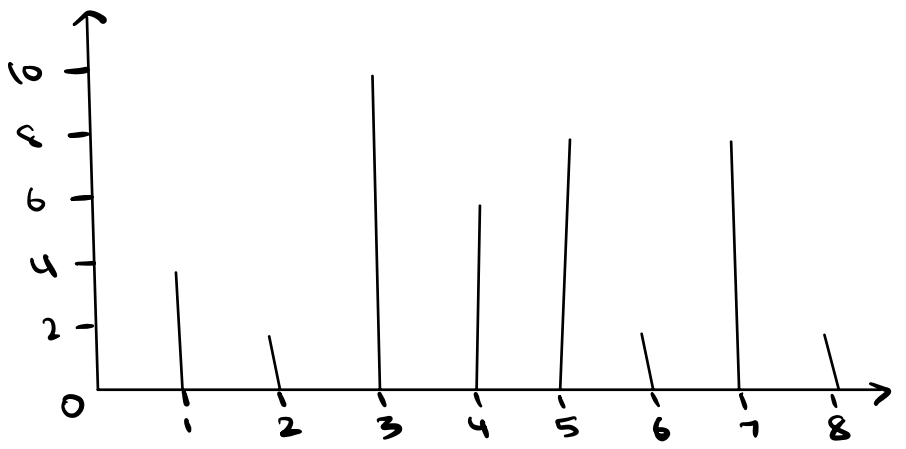


min (A[i], A[j]) x j - i

Water stored b/w A[i] & A[j]

= min (A[i], A[j]) * (j - i)

. Return max water that can be stored b/w any 2 walls



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

i j

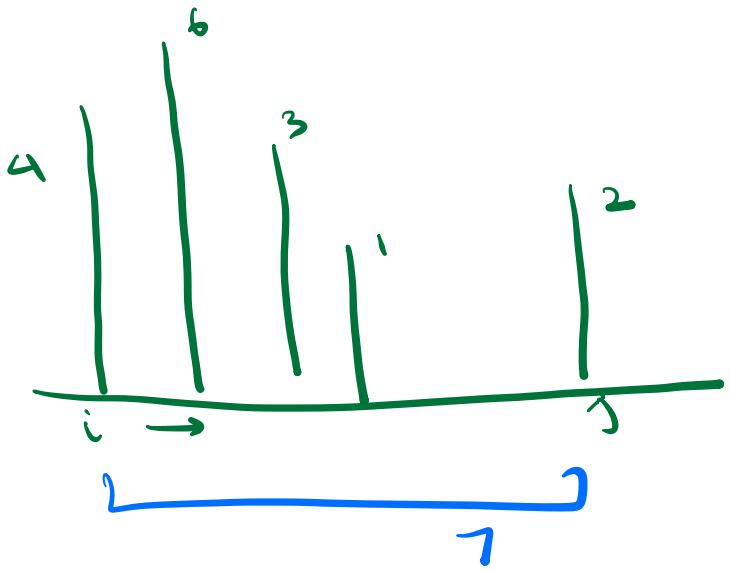
min(4,2) = 2

water = 2 * 7 = 14 units

0 1 2 3 4 5 6 7

[4 2 10 6 8 2 6 2]

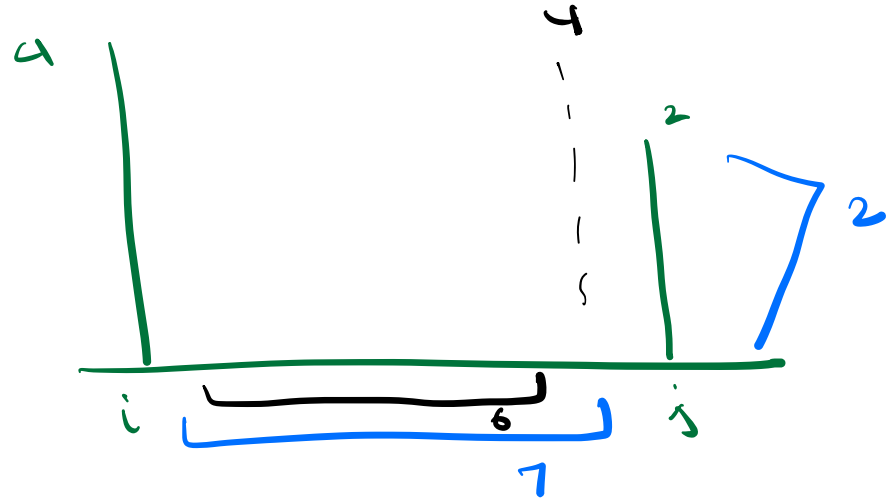
i j

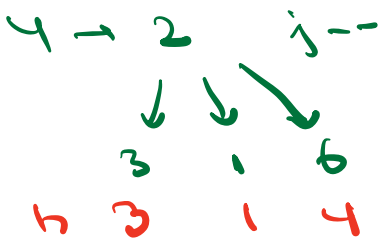


2 → 4 i++

6 3 1

h 2 2 1





Always move smaller wall

0	1	2	3	4	5	6	7
4	2	10	6	4	2	6	2
		i					
		j					

	i	j	water	Max =D
j--	0	7	14	14
i++	0	6	24	24
i++	1	6	10	24
j--	2	6	24	24
j--	2	5	6	24
j--	2	4	16	24

j-- 2 3 6 24

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

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

2 ptr → pairs
↳ starting pt of i and j

ele pair with sum = k
↳ sorted arr

eliminate pairs and move in a specific direction