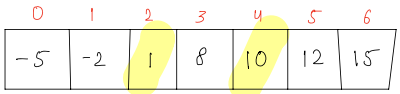
Lecture :- Two pointers

Agenda

- Pairs with given sum
- Count pair sum = k
- pair difference = k
- Subarray with sum = k
- Container with most water.

Qu:1 Given A[], sorted and integer K, find any pair (i, j) such that

A[i] + A[j] = K and i != j



K = 11

ans ⇒ true.

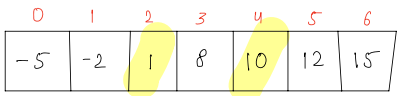
1	2	8	18	31

K = 4

ans ⇒ false

Brute force approach $O(n^2)$

Approach 2



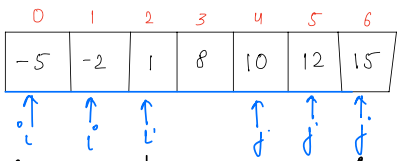
K = 11

a = -5
b = 16
a+b = 11.
search (16)
idx (1-6)

a = -2
b = 13
a+b = 11
search (13)
idx (2-6)

TC: O(n * log n). [Better approach than n^2]

Approach3



k = 11

i	j	Action
0	6	sum = A[0] + A[6] = 10 sum < k i++ ✓ j-- ✗
1	6	sum = A[1] + A[6] = 13 sum > k j--
1	5	sum = 10 sum<k i++
2	5	sum = 13 sum > k j--
2	4	sum = 11 sum == k return (i,j); return true.

Pseudocode

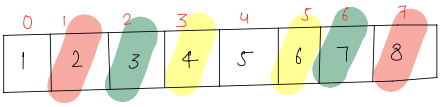
```
i = 0;
j = A.length - 1;
while ( i < j ) {
    sum = A[i] + A[j];
    if ( sum > K) {
        j--;
    }
    else if ( sum < K) {
        i++;
    }
    else {
        return true;
    }
}
```

Tc: O(n)

Sc: O(1)

Qu2 Count sum pair = k

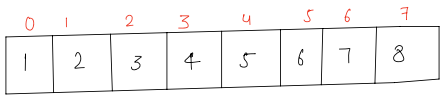
find all pairs in sorted array whose sum is k. [i ! = j]



k = 10

ans = 3

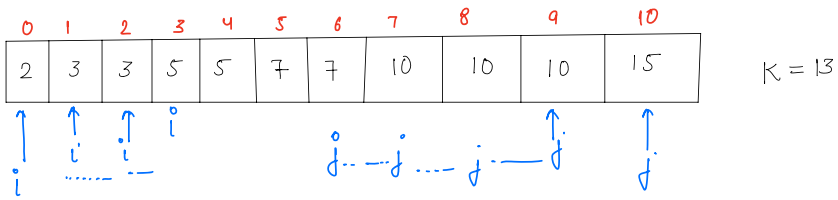
Case 1 When all elements are distinct



k = 10

```
i = 0;
j = A.length - 1;
count = 0;
while ( i < j ) {
    sum = A[i] + A[j];
    if ( sum > k) {
        j--;
    }
    else if ( sum < k) {
        i++;
    }
    else {
        count++;
    }   i++;
}       j--;
return count;
```

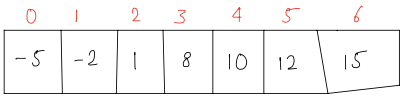
Case2 When elements are repeating



i	j	Action	count
0	10	sum = A[0] + A[10] = 17 sum > K j---	
0	9	sum = 12 sum < K 12<13 i++	
1	9	sum = 13 sum == K countI = 2 countJ = 3 count = 2*3 [countI * countJ]	6
3	6		

```
i = 0;
j = A.length - 1;
count = 0;
while ( i < j ) {
    sum = A[i] + A[j];
    if ( sum > K) {
        j--;
    }
    else if ( sum < K) {
        i++;
    }
    else {
      countI = 1; countJ = 1;
      while(i<j && A[i] == A[i+1]) {
            i++; countI++;
      }
      while(i<j && A[j] == A[j-1]) {
          j--; countJ++;
      }
      count += countI * countJ;
      i++;
      j--;
}   )
```

Qu3 Given A[n] (sorted) and integer K, find any pair (i,j) such that A[j] − A[i] = K and i != j and K > 0.



K = 11

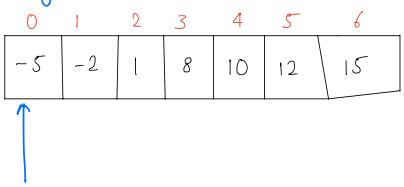
ans = true {2,5}
12 − 1 = 11.

Brute force approach

$O(n^2)$

$O(1)$

Approach2 Binary search.



K = 11

a = -5

b - a = K

b - (-5) = 11

b + 5 = 11

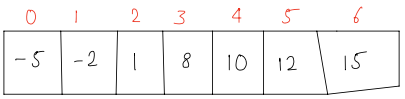
b = 6

search(1, 6)

TC: O(nlogn)

SC: O(1)

Approach3



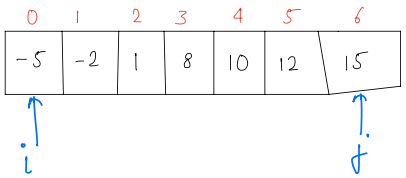
K = 11

Where should we keep our pointers ?

Option 1

$i = 0$

$j = n-1$



K = 11

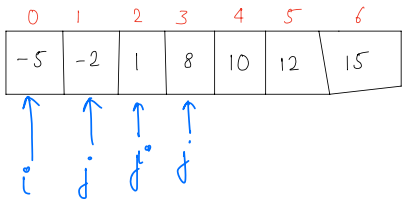
$i = 0$, $j = 6$

diff = A[j] - A[i]
= 15 - (-5) = 20

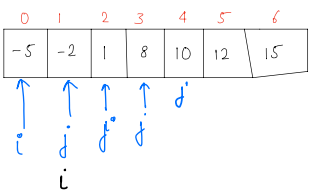
diff > K
20 > 11

- i++; ~~X~~
- j--; ~~X~~
- Cannot decide ✓ (cannot move both i & j)

Option 2



K = 11

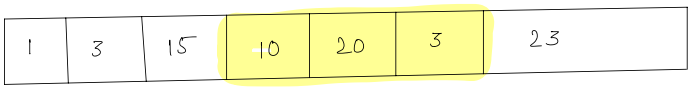
i	j	Action
0	1	diff = A[1] − A[0] = −2 − (−5) = 3 diff < K i++; — diff ↓ j++; ~ diff ↑
0	2	diff = A[2] − A[0] = 6 j++
0	3	diff = A[3] − A[0] = 13 diff > K. i++
		
1	3	diff = 10 j++
1	4	diff = 12 i++
2	4	diff = 9 j++
2	5	diff = 11 return true.

Pseudocode

```
i = 0;
j = A.length - 1;                    1
while ( i < j )                      {
                j < n. && i<n.
    diff = A[j] - A[i]
    if ( sum                         (k)
         diff
        j++;
    }
    else if ( sum                    >k) {
                diff
        i++;
    }
    else {
        return true;
    }
}
return false;
```

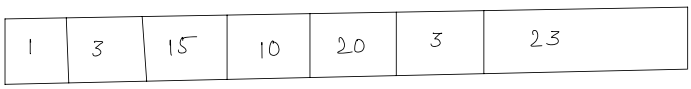
TC: O(n)
SC: O(1)

Qu Given A[n] and integer K, check if there exists a subarray with sum = K



K = 33

ans = true



K = 43

ans = no subarray

{ No of subarray = $\frac{n(n+1)}{2}$ }

Brute force approach

$O(n^2)$ { Go to all subarrays & calculate sum }

$O(1)$

Approach 2

	0	1	2	3	4	5	6
arr	1	3	15	10	20	3	23
pf	1	4	19	29	49	52	75

$k = 33$

→ 100% sorted [+ve integers]

pf[5] − pf[2] → subarray sum from 3 to 5 idx

Diff pair → prefix sum.

subarray [arr] ⟺ diff of pairs [pf]

TC: O(n)
SC: O(n)
O(1) ~ { changes in original array }

Break: 8:49 – 9:00

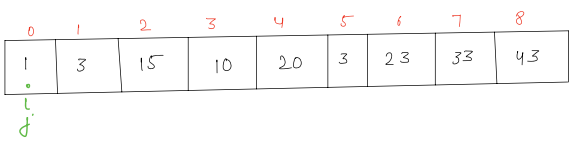
Approach3

sorted array.

0	1	2	3	4	5	6	7	8
1	3	15	10	20	3	23	33	43

i, j

K = 33

i	j	Action
0	0	sum = 1 1 < K Incr the length of subarray. j++
0	1	sum = 4 4 < K. j++
0	2	sum = 19 19 < K j++
0	3	sum = 29 29 < K j++
0	4	sum = 49 49 > K. Decr the length of subarray i++
1	4	sum = 48 48 > K i++
		
2	4	sum = 45 45 < K i++
3	4	sum = 30 30 < K. j++
3	5	sum = 33 return true

Pseudocode

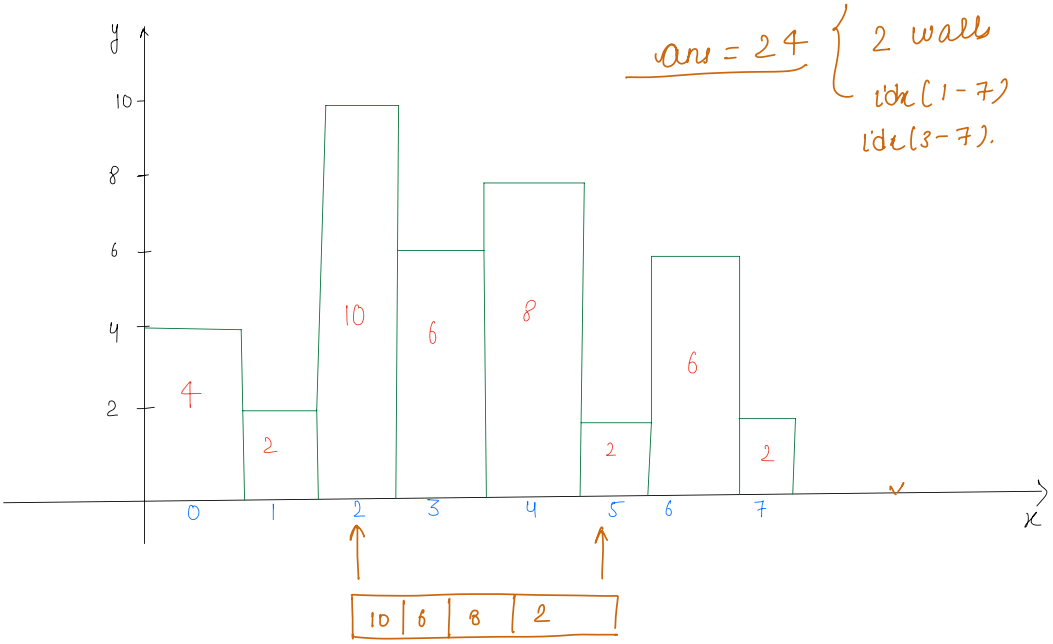
```
i=0 , j=0 . sum = A[0];
while( j<n) {
    if ( sum==K) {
        return true;
    }
    if ( sum<K) {
        j++;
        sum += A[j];
    } else {
        sum-= A[i];
        i++;
        if (i>j) {
            break;
        }
    }
}
return false;
```

// Ensure i never exceeds j

Qus Given A[n] where array elements represents height of the wall. find any two walls that can form a container to store the maximum amount of water.

h | w

0	1	2	3	4	5	6	7
4	2	10	6	8	2	6	2



ans = 24 { 2 walls idx (1-7) idx (3-7).

Brute force' $O(n^2)$

Approach2

0	1	2	3	4	5	6	7
4	2	10	6	8	2	6	2

i	j	Action

Pseudocode