agenda:
1- Pair sum = k
2- Pair diff = k
3- subarr sum = k
4- Container with most water

P1 Given a sorted (distinct elements) integer array and an integer k, find any pair (i,j) s.t. a[i] + a[j] = k & i != j SC: O(1)

ret index i,j

ex a: {-5, -2, 1, 8, 10, 12, 15} (0 1 2 3 4 5 6) k = 11 (2,4) ← ans

idea1 Brute force, for i ... for j = i+1 ... $O(n^2)$ (i,j) TC

idea2 a[i] + a[j] = k ⇒ a[j] = k - a[i]

(i != j) ← ∀i, a[i] → binSearch(k - a[i]) TC: O(n log n) log(n)

idea3 two pointer

a: {-5, -2, 1, 8, 10, 12, 15} k = 11 (?, ?) a[i] a[j]

i → ← j

esp done
less i++
move j--

-5 + 15 = 10
a[0] + largest element in array = 10 < 11
i = 0 ⇒ a[0] + any element in array < 11
i = 1 a[6] + smalst element in array = 13 > 11 ⇒ a[6] + any available element > 13
j = 6 a[j] + smallest available element in arr > 11
j = 5 a[1] + a[5]
-2 + 12 = 10 < 11
a[2] + a[15] = 13 > 11
a[2] + a[4] = 1 + 10 = 11

1- when to intitialy put the two pointer indexes? how to move them?
2- when can I use two pointer what if my algo is short sighted and a counter example shows its wrong

```
i=0   j=n-1                 TC: O(n)
while(i<j){                 SC: O(1)
 if(a[i]+a[j]==k)
     ret (i,j) // for finding all pairs Continue the process
 else if(a[i]+a[j]<k)
    i++
 else // a[i]+a[j] > k
    j--
}
ret i,j
-1,-1 → not found
```

distinct elements

P2 Given a sorted integer array and an integer $k>0$, find any pair (i,j) s.t. $a[j]-a[i]=k$ $i \neq j$ SC: $O(1)$

ex $a = \{-5, -2, 1, 8, 10, 12, 15\}$ (indices 0 1 2 3 4 5 6), $k=11$

$a[j]-a[i]=k$ & $k>0$

$a[j] > a[i]$

idea1 Brute Force $O(n^2)$

idea2 $a[j]=k+a[i]$ $O(n \log n)$

bin search

idea3

$a = \{-5, -2, 1, 8, 10, 12, 15\}$, $k=11$

two pointers

i at index 0, j at index 6

i from → , j from ←

$a[j]-a[i]=15-(-5)=20>11$ ✓

less i++ move j--

doesn't work

---

j at index 6, i at index 5

$a[6]-a[i]=15-12=3<11$

Largest available number $- a[5] < 11$

$\Rightarrow$ any number $- a[5] < 11$ ✓

$a[6]-a[4]=15-10=5<11$

$a[6]-a[3]=15-8=7<11$

$a[6]-a[2]=15-1=14>11$

$a[6] \to 15$ $- a[2] > 11 \to j--$

$\Rightarrow$ ↳ Move to next largest element

$a[5] - a[2] = 11$ ← ans

$a[5]$ = 12, $a[2]$ = 1

$a = \{-5, -2, 1, 8, 10, 12, 15\}$, $k=11$

i at index 2, j at index 5

$a[j]-a[i]$ ① ②

l to r →

r to l ←

Psudo Code:

O(n):

```
i=0   j=1
while(j<n && i<n){
  if(a[j]-a[i]==k) ret (i,j)
  else if(a[j]-a[i]<k) j++;
  else i++                    // a[j]-a[i]>k
}
ret(-1,-1)
```

O(1):

```
i=n-2   j=n-1
while(i>=0 && j>=0){
  if(a[j]-a[i]==k) ret (i,j)
  else if(a[j]-a[i]<k) i--
  else j--
}
ret(-1,-1)
```

duplicates

HWs ↗

P3 Given an int array of positive elements ($a[i]>0$) & an int K

True / false ← check if there exist a subarray with sum K

ex: a: { 1, 3, 15, 10, 20, 3, 23 } K=33 → true

K=43 → false

idea1 Brute force $O(n^3)$ optimize using carryforward $O(n^2)$

idea2

ex: a: { 1, 3, 15, 10, 20, 3, 23 }

Prefix Sum ← PS: {1, 4, 19, 29, 49, 52, 75} → prev problem → P2

$$sum[i,j] = \begin{cases} PS[j] - PS[i-1] & i>0 \\ PS[j] & i==0 \end{cases}$$

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

idea3

two pointer ex: a: { 1, 3, 15, 10, 20, 3, 23 } K=33

TC:

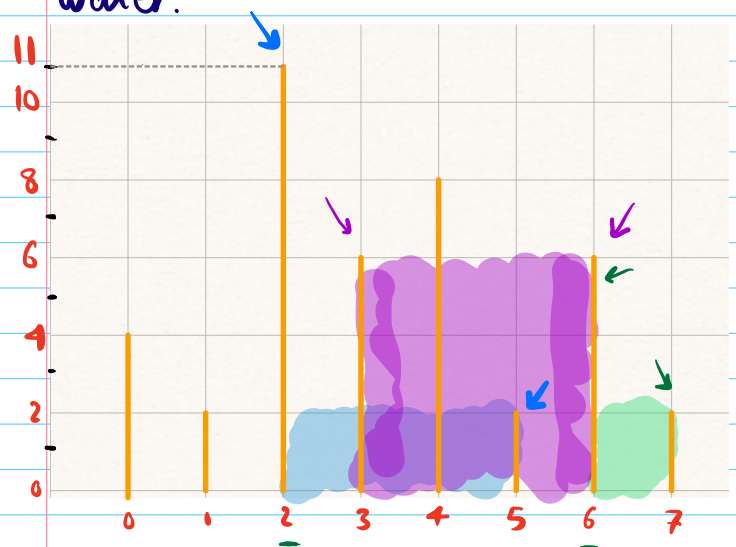
SC:

a: { 3, 3, 15, 10, 20, 3, 23 } k=33

```
while( j<n && i<n){
    if( sum == k) ret true
    if( sum<k){
            j++;
            if( j == n) break;
            sum += a[j]
    }
    else{
        sum -= a[i]
        i++
    }
} ret false;
```

i	J	sum
0	0	3
0	1	6
0	2	21
0	3	31
0	4	51
1	4	48
2	4	45
3	4	30
3	5	33 ✓

P4 Find two walls that can form a container to hold max water.



$(5-2)\times 1 = 3$

$(6-3)\times 3 = 9$

$(7-6)\times 1 = 1$

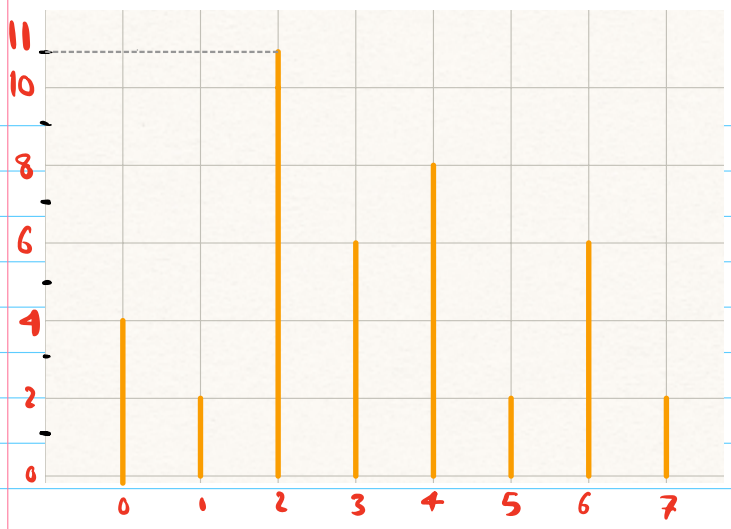
a: {4, 2, 11, 6, 8, 2, 6, 2}

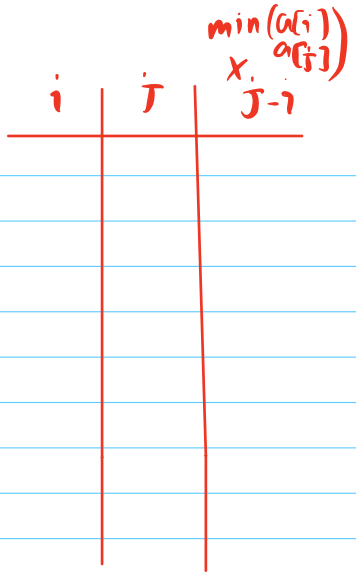
$O(n^2)$ brute force

```
for i 0 → n-1
  for j = i+1 → n-1
```

min(a[i], a[j]) × (j − i) = area water → max

```
i=0; j=n-1; area=0; max=0
while(i<j){
    Area = min(a[i], a[j]) × (j-i) ; max = Max(Area, max)
  if(a[i]<a[j]) i++;
  else if(a[i]>a[j]) j--;
  else if(a[i]==a[j]){ i++; j--;}
}
ret max;
```





a: {4, 2, 11, 6, 8, 2, 6, 2}

---

a: {1, 3, 5, 6, 9, 100}

b: {-2, 7, 11, 99, 101}

105
(6, 99)

8
(1, 7)

(a_i, b_j)

$(a_i, b_j) \xrightarrow{\text{sum}} a_i + b_j$

return the 5 smallest value pairs among all possible pairs

-1 8 13 102
(1, -2), (1, 7), (1, 11), ... (1, 101)

(3, -2), ... , (3, 101)

99
(100, -2), ... , (100, 101)