Today's agenda

- Two Sum
- Container with most water
- Sort 0, 1, 2 → dutch national flag

Q) Two Sum

↳ Given N distinct sorted ele, check if there exists a pair(i,j) such that A[i] + A[j] == K [i != j]

Eg: A[] → {3 7 8 12 19}, K=15 → true

A[] → {2 5 8 11 15}, K=16 → true

A[] → {1 3 9 20 27}, K=14 → false

Idea1

↳ check all pairs.

T.C: $O(N^2)$ S.C: $O(1)$

Idea2

↳ using HashMap

T.C: $O(N)$ S.C: $O(N)$

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

A[i] + A[j] > K → j--

A[i] + A[j] < K → i++

//idea 3

K = 17

i ↓ j ↓

A[] = { -3 0 1 3 6 8 11 14 18 25 }

A[i]	A[j]	Sum	*
-3	25	22	j--
-3	18	15	i++
0	18	18	j--
0	14	14	i++
1	14	15	i++
3	14	17	→ return true

while i < j

O(N²) Pairs ⟶ O(N) Pairs

100 Pairs → 10 Pairs

K=17

i

j

A[] = { -3 0 1 3 6 8 11 14 18 25 }
0 1 2 3 4 5 6 7 8 9

(0,9) ??

(0,8) ??

A[0]+A[8]: 15 < K

↓A[0] + A[7] = 15↓ < K

A[0] + A[9] : 22 > K → j--

↑A[1] + A[9] = ↑22 > K

↑A[2] + A[9] = ↑22 > K

// Pseudo Code

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

T.C: O(N)

S.C: O(1)

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

A[i] + A[j] > K → j--

A[i] + A[j] < K → i++

* Things to focus while applying 2 Pointers.

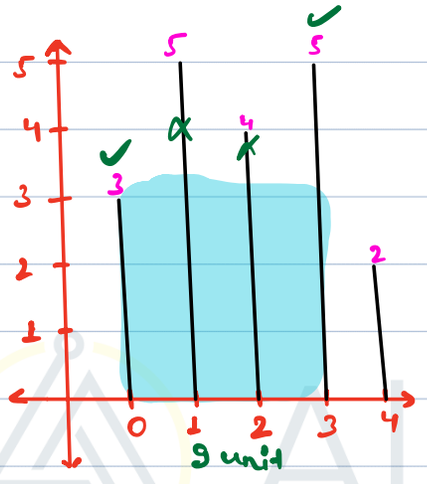
↳ if you are checking all the Pairs, try optimizing it with two Pointers.

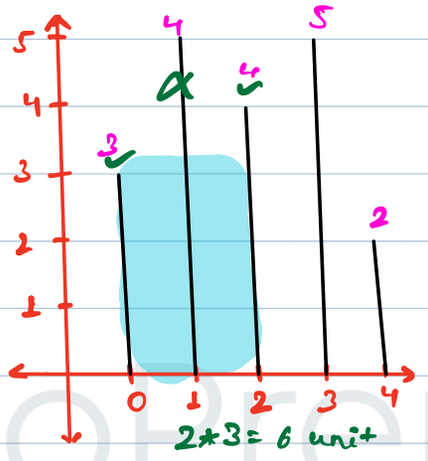
1) where to initialize two Pointers.
2) update your Pointers.
3) when to stop.

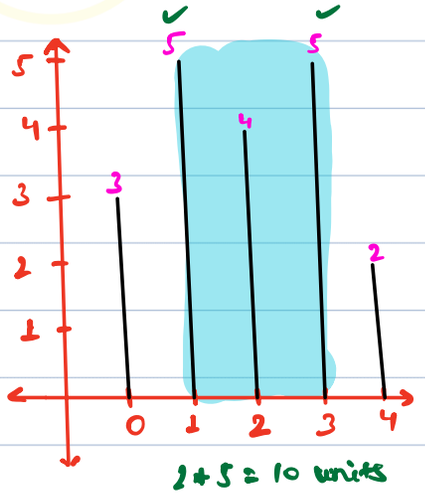
Q) Container with most water

↳ Given an arr[N], where A[i] represents height of each wall. Pick any 2 walls s.t max water is accumulated between them.

Ex:







Idea 1

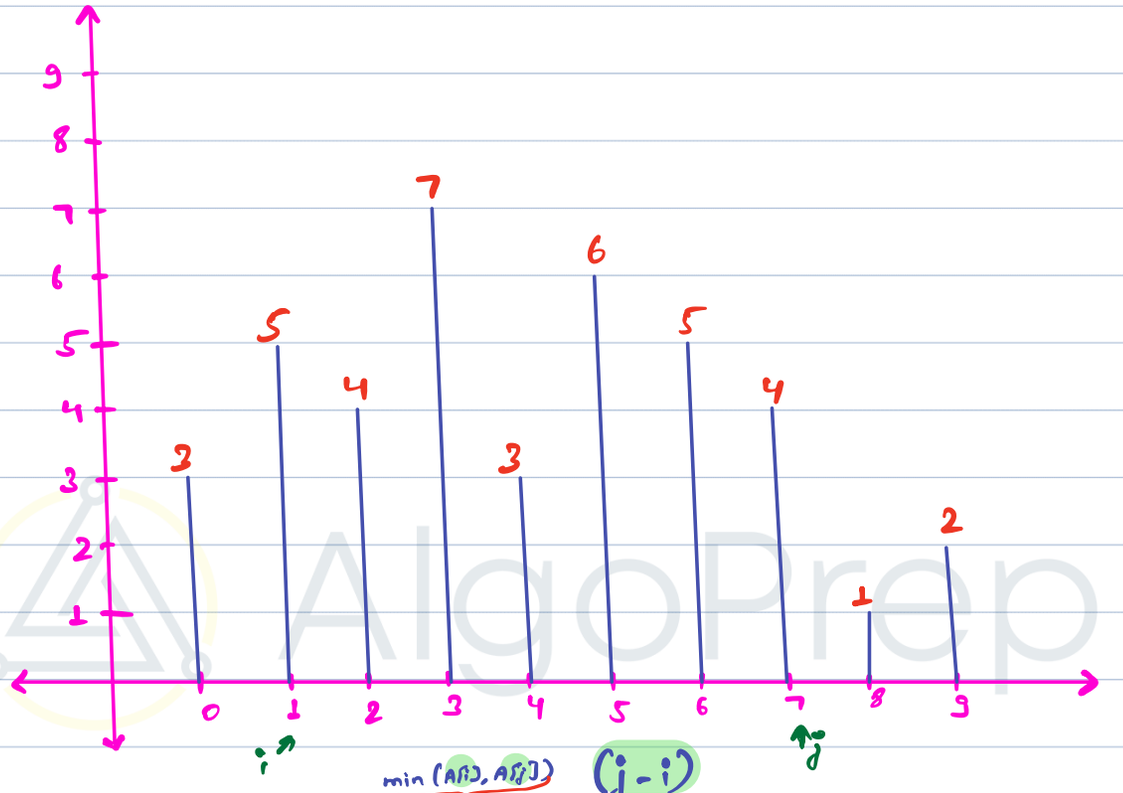
↳ Check all Pairs

T.C: $O(N^2)$ S.C: $O(1)$

* Amount of water (i,j) = $(j-i)$ * $\min(A[i], A[j])$

$(j-i)$: width; $\min(A[i], A[j])$: height

A[]: {3 5 4 7 3 6 5 4 1 2}

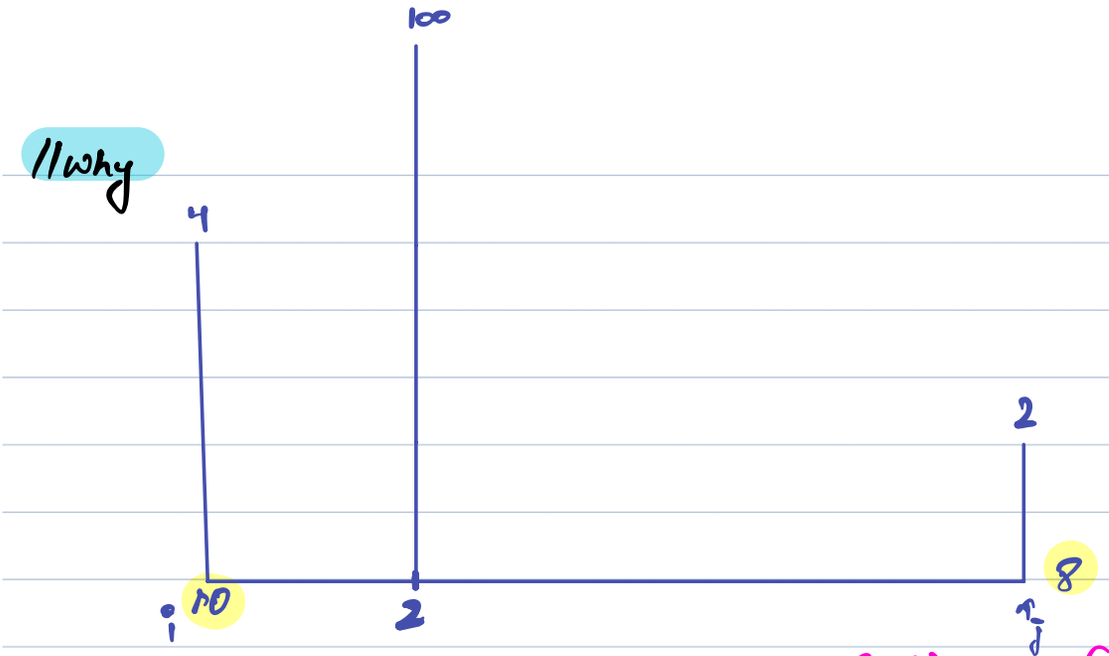


A[i]	A[j]	height	width	water	
3	2	2	9	18	→ j--
3	1	1	8	8	→ j--
3	4	3	7	21	→ i++
5	4	4	6	24	→ j--

$O(N^2) \rightarrow O(N)$

100 10

(i < j)



//why

(j-i)=8 (j-i)↓

(0,8) (2,8)

(0,8) !! → ~~x~~

(1,8)

(2,8)

(3,8)

(4,8)

(5,8)

(6,8)

(7,8)

Case ①: A[2] < A[0]

Goverall smaller

Case ②: A[2] == A[0]

(j-i) (0,8) > (j-i) (2,8) ↓

Height are same

Goverall smaller

Case ③: A[2] > A[0]

width → (0,8) > (2,8) ↓

Height → height will remain same. ✓

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

height

//Pseudo code

```
int maxWater (int arr[N]){
    int i=0;
    int j=0;
    int ans = -∞;

    while (i<j) {
        int w = (j-i) * min(arr[i], arr[j]);

        ans = max(ans, w);

        if (arr[i] <= arr[j]){
            i++;
        }
        else {
            j--;
        }
    }

    return ans;
}
```

T.C: O(N)

S.C: O(1)

Q) Sort 0, 1 and 2

↳ Given an arr[N] containing only 0, 1 and 2. Sort the array in ascending order.

Ex: arr[10]: { 0 1 2 1 2 1 0 0 1 0 }

(indices: 0 1 2 3 4 5 6 7 8 9)

↳ 0 0 0 0 1 1 1 1 2 2

//idea1

↳ Sort the entire array.

T.C: O(N log N) S.C: O(1)

//idea2

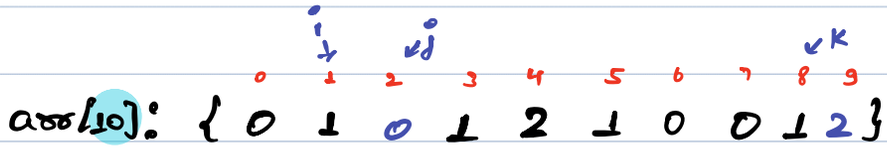
↳ Count sort

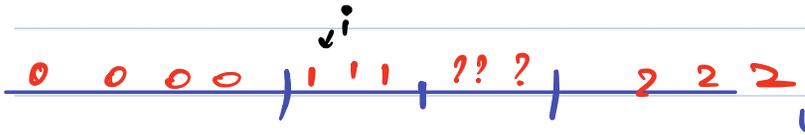
T.C: $O(N + \text{max}) = O(N)$ S.C: O(1)

//idea2

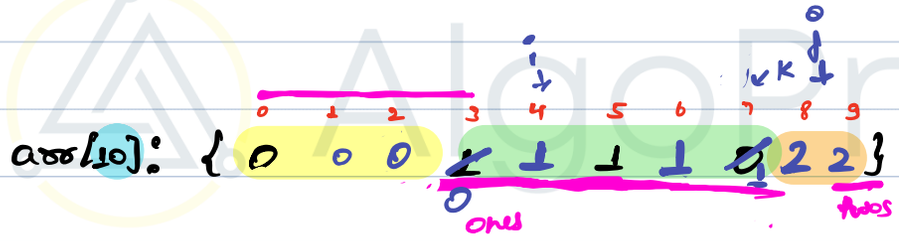
$(0, i-1) \rightarrow 0\ 0\ 0$
$(K+1, N-1) \rightarrow 2\ 2\ 2$
$(i, K) \rightarrow 1\ 1\ 1$

$(0, i-1) \rightarrow 0\ 0\ 0$
$(i, j-1) \rightarrow 1\ 1\ 1$
$(j, K) \rightarrow$ to be solved
$(K+1, N-1) \rightarrow 2\ 2\ 2 \ldots$





→ arr[j] == 0 → swap(i, j) → i++, j++
↳ arr[j] == 1 → j++
↳ arr[j] == 2 → swap(j, K) → K--



$(0, i-1) \rightarrow 0\ 0\ 0$
$(i, j-1) \rightarrow 1\ 1\ 1$
$(j, K) \rightarrow$ to be solved
$(K+1, N-1) \rightarrow 2\ 2\ 2 \ldots$

→ arr[j] == 0 → swap(i,j) → i++, j++

↳ arr[j] == 1 → j++

↳ arr[j] == 2 → swap(j,k) → k--

//Pseudo code

```
void sort012 (int arr[N]){
    int i=0;
    int j=0;
    int k=N-1;

    while(j<=k)
        if (arr[j]==0){
            swap(i,j);
            i++; j++;
        }
        else if (arr[j]==1){
            j++;
        }
        else{
            swap(j,k);
            k--;
        }
}
```

T.C: O(N)

S.C: O(1)

↳ Dutch national flag algo.

Two Sum

Java Code:

```
class Solution {
    public class pair implements Comparable<pair>{
        int num;
        int idx;
        pair(int num , int idx){
            this.num = num;
            this.idx = idx;
        }
        public int compareTo(pair O){
            if(this.num != O.num) return this.num-O.num;
            else return this.idx-O.idx;
        }

    }
    public int[] twoSum(int[] nums, int target) {
        int n = nums.length;
        pair[] arr=  new pair[n];
        for(int i = 0; i<n; i++){
            arr[i] = new pair(nums[i] ,i);
        }
        Arrays.sort(arr);

        int i = 0;
        int j = n-1;
        while(i<j){
            int left = arr[i].num;
            int right = arr[j].num;
            int sum = left+right;
         if(sum == target) return new int[]{arr[i].idx , arr[j].idx};
         else if(sum<target) i++;
            else j--;
        }
        return new int[2];
    }
}
```

C++ Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

class Solution {
public:
    struct Pair {
        int num;
        int idx;

        Pair(int num, int idx) : num(num), idx(idx) {}

        bool operator<(const Pair& other) const {
            if (num != other.num) return num < other.num;
            else return idx < other.idx;
        }
    };

    vector<int> twoSum(vector<int>& nums, int target) {
        int n = nums.size();
        vector<Pair> arr(n);

        for (int i = 0; i < n; i++) {
            arr[i] = Pair(nums[i], i);
        }

        sort(arr.begin(), arr.end());

        int i = 0;
        int j = n - 1;

        while (i < j) {
            int left = arr[i].num;
            int right = arr[j].num;
            int sum = left + right;

            if (sum == target) {
                return {arr[i].idx, arr[j].idx};
            } else if (sum < target) {
                i++;
            } else {
                j--;
            }
        }
```

```
        return {0, 0};
    }
};

int main() {
    Solution solution;
    vector<int> nums = { /* your input array */ };
    int target = /* your target value */;

    vector<int> result = solution.twoSum(nums, target);

    // Print the result
    for (int num : result) {
        cout << num << " ";
    }

    return 0;
}
```

Python Code:

```
class Solution:
    class Pair:
        def __init__(self, num, idx):
            self.num = num
            self.idx = idx

        def __lt__(self, other):
            if self.num != other.num:
                return self.num < other.num
            else:
                return self.idx < other.idx

    def two_sum(self, nums, target):
        n = len(nums)
        arr = [self.Pair(nums[i], i) for i in range(n)]
        arr.sort()

        i = 0
        j = n - 1

        while i < j:
            left = arr[i].num
            right = arr[j].num
```

```
            sum_val = left + right

            if sum_val == target:
                return [arr[i].idx, arr[j].idx]
            elif sum_val < target:
                i += 1
            else:
                j -= 1

        return [0, 0]

# Example usage:
solution = Solution()
nums = [ /* your input array */ ]
target = /* your target value */
result = solution.two_sum(nums, target)
print(result)
```

Container With Most Water

Java Code:

```
class Solution {
    public int maxArea(int[] height) {

        int i = 0,j = height.length-1;
        int ans = Integer.MIN_VALUE;
        while(i<j){
            int w = (j-i)*Math.min(height[i],height[j]);
            ans = Math.max(ans,w);
            if(height[i] <= height[j]) i++;
            else j--;
        }
       return ans;
    }
}
```

C++ Code:

```
#include <iostream>
#include <vector>
#include <algorithm>

class Solution {
public:
    int maxArea(std::vector<int>& height) {
        int i = 0, j = height.size() - 1;
        int ans = INT_MIN;

        while (i < j) {
            int w = (j - i) * std::min(height[i], height[j]);
            ans = std::max(ans, w);

            if (height[i] <= height[j])
                i++;
            else
                j--;
        }

        return ans;
    }
};

int main() {
    std::vector<int> height = {1, 8, 6, 2, 5, 4, 8, 3, 7};
    Solution solution;
    int result = solution.maxArea(height);
    std::cout << "Maximum area: " << result << std::endl;

    return 0;
}
```

Python Code:

```
class Solution:
    def maxArea(self, height):
        i, j = 0, len(height) - 1
        ans = float('-inf')

        while i < j:
            w = (j - i) * min(height[i], height[j])
            ans = max(ans, w)
```

```
            if height[i] <= height[j]:
                i += 1
            else:
                j -= 1

        return ans


# Example usage
height = [1, 8, 6, 2, 5, 4, 8, 3, 7]
solution = Solution()
result = solution.maxArea(height)
print("Maximum area:", result)
```

Sort 0, 1 and 2

Java Code:

```
class Solution
{
    public static void sort012(int arr[], int n)
    {
       int i = 0, j = 0, k = n-1;
        while(j<=k){
            if(arr[j] == 0){
                 swap(arr,i,j);
                j++;
                i++;
            } else if(arr[j] == 1){
                j++;
            } else {
                 swap(arr,j,k);
                 k--;
            }
        }
    }
    public static void swap(int[] arr, int i,int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
```

```
}
```

C++ Code:

```
#include <iostream>
#include <vector>

class Solution {
public:
    static void sort012(int arr[], int n) {
        int i = 0, j = 0, k = n - 1;

        while (j <= k) {
            if (arr[j] == 0) {
                swap(arr, i, j);
                j++;
                i++;
            } else if (arr[j] == 1) {
                j++;
            } else {
                swap(arr, j, k);
                k--;
            }
        }
    }

    static void swap(int arr[], int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
};

int main() {
    int arr[] = {0, 1, 2, 0, 1, 2};
    int n = sizeof(arr) / sizeof(arr[0]);

    Solution::sort012(arr, n);

    std::cout << "Sorted array: ";
    for (int i = 0; i < n; i++) {
        std::cout << arr[i] << " ";
    }

    return 0;
}
```

Python Code:

```
class Solution:
    @staticmethod
    def sort_012(arr, n):
        i, j, k = 0, 0, n - 1

        while j <= k:
            if arr[j] == 0:
                Solution.swap(arr, i, j)
                j += 1
                i += 1
            elif arr[j] == 1:
                j += 1
            else:
                Solution.swap(arr, j, k)
                k -= 1

    @staticmethod
    def swap(arr, i, j):
        arr[i], arr[j] = arr[j], arr[i]


# Example usage
arr = [0, 1, 2, 0, 1, 2]
n = len(arr)

Solution.sort_012(arr, n)

print("Sorted array:", arr)
```

Sort colors

Java Code:

```
class Solution {
    public void sortColors(int[] nums) {
        int n = nums.length;
        int i = 0, j = 0, k = n-1;
        while(j<=k){
            if(nums[j] == 0){
```

```
                swap(nums,i,j);
                j++;
                i++;
            } else if(nums[j] == 1){
                j++;
            } else {
                swap(nums,j,k);
                k--;
            }
        }
    }
    public static void swap(int[] arr, int i,int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

}
```

C++ Code:

```
#include <iostream>
#include <vector>
using namespace std;

class Solution {
public:
    void sortColors(vector<int>& nums) {
        int n = nums.size();
        int i = 0, j = 0, k = n - 1;

        while (j <= k) {
            if (nums[j] == 0) {
                swap(nums[i], nums[j]);
                j++;
                i++;
            } else if (nums[j] == 1) {
                j++;
            } else {
                swap(nums[j], nums[k]);
                k--;
            }
        }
    }
};
```

```
int main() {
    Solution solution;
    vector<int> nums = { /* your input array */ };

    solution.sortColors(nums);

    // Print the result
    for (int num : nums) {
        cout << num << " ";
    }

    return 0;
}
```

Python Code:

```
class Solution:
    def sortColors(self, nums):
        n = len(nums)
        i, j, k = 0, 0, n - 1

        while j <= k:
            if nums[j] == 0:
                nums[i], nums[j] = nums[j], nums[i]
                j += 1
                i += 1
            elif nums[j] == 1:
                j += 1
            else:
                nums[j], nums[k] = nums[k], nums[j]
                k -= 1

# Example usage:
solution = Solution()
nums = [ /* your input array */ ]
solution.sortColors(nums)

# Print the result
print(nums)
```

Pair Difference

Java Code:

```
class Solution
{
    public boolean findPair(int arr[], int size, int n)
    {
       Arrays.sort(arr);
        int i = 0;
        int j = 1;
        while(i < size && j < size){
            if(i != j && arr[j] - arr[i] == n){
                return true;
            }else if(arr[j]- arr[i] > n){
                i++;
            }else{
                j++;
            }
        }
        return false;
    }
}
```

C++ Code:

```
#include <iostream>
#include <algorithm>
using namespace std;

class Solution {
public:
    bool findPair(int arr[], int size, int n) {
        sort(arr, arr + size);
        int i = 0;
        int j = 1;

        while (i < size && j < size) {
            if (i != j && arr[j] - arr[i] == n) {
                return true;
            } else if (arr[j] - arr[i] > n) {
                i++;
            } else {
                j++;
```

```
            }
        }

        return false;
    }
};

int main() {
    Solution solution;
    int size = /* size of your array */;
    int* arr = new int[size];

    // Initialize your array with values

    int n = /* your target difference */;
    bool result = solution.findPair(arr, size, n);

    cout << (result ? "true" : "false") << endl;

    delete[] arr;
    return 0;
}
```

Python Code:

```
class Solution:
    def findPair(self, arr, size, n):
        arr.sort()
        i, j = 0, 1

        while i < size and j < size:
            if i != j and arr[j] - arr[i] == n:
                return True
            elif arr[j] - arr[i] > n:
                i += 1
            else:
                j += 1

        return False

# Example usage:
solution = Solution()
arr = [/* your array */]
size = len(arr)
```

```
n = /* your target difference */
result = solution.findPair(arr, size, n)
print(result)
```

Today's agenda

- max subarray sum
- minimum swaps

Q) Given N elements, return max subarray sum of len : k.

k=3

ex: arr[]: 2 4 3 7 9 8 5 4 3 (indices 0 1 2 3 4 5 6 7 8)

9 (2+4+3)
14 (4+3+7)
19 (3+7+9)
24 (7+9+8)
22 (9+8+5)
17 (8+5+4)
12 (5+4+3)

= 24

Idea 1

↳ nested loop

T.C: O(N*k) S.C: O(1)

Idea 2

↳ using Prefix sum.

k=3

	0	1	2	3	4	5	6	7	8
arr[]	2	4	3	7	9	8	5	4	3
pf[]	2	6	9	16	25	33	38	42	45

sp	ep	Sum	ans :-∞
0	2	9	9
1	3	14	14
2	4	19	19
.	.	.	.
.	.	.	.

T.C: O(N) S.C: O(N)

Idea 3

k=3

	0	1	2	3	4	5	6	7	8
arr[i]:	2	4	3	7	9	8	5	4	3

s	e	Sum	ans :-∞
0	2	9	9
1	3	9+7-2:14	14
2	4	14+9-4=19	19
3	5	19+8-3:24	24

//Pseudo code

```
int subsumk (int arr[N], int k){
    int sum=0;
    for(int i=0; i<k; i++){        //O(k-1)
        sum= sum+ arr[i]
    }
    int ans= sum;
    int s=1;
    int e=k;
    while (e<N ){
        sum= sum+ arr[e] - arr[s-1]
        ans= max(ans,sum);
        s++;
        e++;
    }

    return ans;
}
```

T.C: O(N)

S.C: O(1)

```
int subsumk (int arr[N], int k){
    int sum=0;
    for(int i=0; i<k; i++){        //o(k-1)
        sum= sum+ arr[i]
    }
    int ans= sum;
    int s=1;
    int e= k;
    while (e<N ){
        sum= sum + arr[e] - arr[s-1]
        ans= max(ans,sum);
        s++;
        e++;
    }

    return ans;
}
```

k=3

	0	1	2	3	4	5	6	7	8
arr[9]:	2	4	3	7	9	8	5	4	3

Sum = 9

ans = ~~9~~ 14 19

s	e	Sum
1	3	9+7-2 = 14
2	4	14+9-4 = 19
.		

Q) minimum swaps

↳ Given an array of integers and an integer B, find and return the minimum number of swaps required to bring all the numbers less than or equal to B together.

eg: arr[7]: {1 12 10 3 14 10 5} B:6

{1 3 5 12 14 10 10} → 2

eg: arr[7]: {1 12 3 10 14 10 5} B:6

↳ 1

//idea1

arr[7]: {1 12 10 3 14 10 5} B=6

0 1 2 3 4 5 6

2

2

2

2

2

ans: 2

B=7

	0	1	2	3	4	5	6	7	8	9
arr[10]:	1	12	6	3	8	13	15	12	4	5

window length : 5

S	e	Count <= B	reqSwap	ans
0	4	3	5-3 = 2	2
1	5	3+0-1 = 2	5-2 = 3	2
2	6	2+0-0 = 2	5-2 = 3	2
3	7	2+0-1 = 1	5-1 = 4	2

:

(

IIPSueds code

```
int minSwaps (int arr[N], int B){
    int K=0;
    for(int i=0; i<N; i++){
        if(arr[i]<=B){K++;}
    }
```

T.C: O(N)

S.C: O(1)

```
    int c=0;
    for (int i=0; i<K; i++){
        if (arr[i]<=B){c++;}
    }
    int ans = K-c;
    int s=1;
    int e=K;

    while (e<N){
        if(arr[e]<=B){c++;}

        if(arr[s-1]<=B){c--;}
        ans= min(ans, K-c);

        s++;
        e++;
    }

    return ans;
}
```

Max Sum Subarray of size K

Java Code:

```
class Solution{
    static long maximumSumSubarray(int K, ArrayList<Integer> Arr,int N){
        long sum = 0;
        for(int i = 0; i<K; i++){
            sum = sum + Arr.get(i);
        }
        long ans = sum;
        int s = 1;
        int e = K;
        while(e<N){
            sum = sum + Arr.get(e) - Arr.get(s-1);
            ans = Math.max(ans , sum);
            s++;
            e++;
        }
        return ans;
    }
}
```

C++ Code:

```
#include <vector>
#include <algorithm>
using namespace std;

class Solution {
public:
    static long maximumSumSubarray(int K, vector<int>& Arr, int N) {
        long sum = 0;
        for (int i = 0; i < K; i++) {
            sum += Arr[i];
        }
        long ans = sum;
        int s = 1;
        int e = K;
        while (e < N) {
            sum = sum + Arr[e] - Arr[s - 1];
            ans = max(ans, sum);
            s++;
            e++;
```

```
        }
        return ans;
    }
};
```

Python Code:

```
class Solution:
    @staticmethod
    def maximumSumSubarray(K, Arr, N):
        sum_val = sum(Arr[:K])
        ans = sum_val
        s = 1
        e = K
        while e < N:
            sum_val = sum_val + Arr[e] - Arr[s - 1]
            ans = max(ans, sum_val)
            s += 1
            e += 1
        return ans
```

Minimum swaps and K together

Java Code:

```
class Complete{
    // Function for finding maximum and value pair
    public static int minSwap (int arr[], int n, int B) {
        //Complete the function
        int k = 0;
        for(int i = 0; i<n; i++){
            if(arr[i] <= B) k++;
        }

        int c = 0;
         for(int i = 0; i<k; i++){
            if(arr[i] <= B) c++;
        }

        int ans = k - c;
        int s = 1;
        int e = k;
```

```
        while(e<n){
            if(arr[e] <= B) c++;
            if(arr[s-1] <= B) c--;

            ans = Math.min(ans , k-c);
            s++;
            e++;

        }
        return ans;
    }
}
```

C++ Code:

```
#include <vector>
#include <algorithm>
using namespace std;

class Complete {
public:
    static int minSwap(vector<int>& arr, int n, int B) {
        int k = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] <= B) k++;
        }

        int c = 0;
        for (int i = 0; i < k; i++) {
            if (arr[i] <= B) c++;
        }

        int ans = k - c;
        int s = 1;
        int e = k;
        while (e < n) {
            if (arr[e] <= B) c++;
            if (arr[s - 1] <= B) c--;

            ans = min(ans, k - c);
            s++;
            e++;
        }
        return ans;
    }
};
```

Python Code:

```
class Complete:
    @staticmethod
    def minSwap(arr, n, B):
        k = 0
        for i in range(n):
            if arr[i] <= B:
                k += 1

        c = 0
        for i in range(k):
            if arr[i] <= B:
                c += 1

        ans = k - c
        s = 1
        e = k
        while e < n:
            if arr[e] <= B:
                c += 1
            if arr[s - 1] <= B:
                c -= 1

            ans = min(ans, k - c)
            s += 1
            e += 1

        return ans
```