**Stock buy and sell**

class Solution{

public:

vector<vector<int> > stockBuySell(vector<int> A, int n){

vector<vector<int>> ans;

int buy = 0;

for (int i = 1; i < n; ++i) {

if (A[i] < A[i - 1]) {

if (buy < i - 1) {

ans.push\_back({buy, i - 1});

}

buy = i;

}

}

if (buy < n - 1) {

ans.push\_back({buy, n - 1});

}

return ans;

}

};  
  
**Coin Change**

class Solution {

public:

int count(vector<int>& coins, int sum) {

vector<int> dp(sum + 1, 0);

dp[0] = 1;

for (int c : coins) {

for (int j = c; j <= sum; ++j) {

dp[j] += dp[j - c];

}

}

return dp[sum];

}

};

**First and Last Occurrences**class Solution {

public:

vector<int> find(vector<int>& arr, int x) {

int n = arr.size();

int first = -1, last = -1;

int low = 0, high = n - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == x) {

first = mid;

high = mid - 1;

} else if (arr[mid] < x) {

low = mid + 1;

} else {

high = mid - 1;

}

}

low = 0, high = n - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == x) {

last = mid;

low = mid + 1;

} else if (arr[mid] < x) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return {first, last};

}

};

**Find Transition Point**

class Solution {

public:

int transitionPoint(vector<int>& arr) {

int n=arr.size();

if (arr[0] == 1) return 0;

if (arr[n - 1] == 0) return -1;

int low = 0, high = n - 1, res = -1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == 1) {

res = mid;

high = mid - 1;

} else {

low = mid + 1;

}

}

return res;

}

};  
  
**First Repeating Element**

class Solution {

public:

int firstRepeated(vector<int> &arr) {

// code here

unordered\_map<int, int> elementInd;

int n = arr.size();

int minind = INT\_MAX;

for (int i = 0; i < n; i++) {

if (elementInd.find(arr[i]) != elementInd.end()) {

minind = min(minind, elementInd[arr[i]]);

} else {

elementInd[arr[i]] = i;

}

}

if (minind== INT\_MAX) return -1;

return minind + 1;

}

};  
  
**Remove Duplicates Sorted Array**

class Solution {

public:

int remove\_duplicate(vector<int> &arr) {

// code here

int n = arr.size();

if (n == 0) return 0;

int ind = 0;

for (int i = 1; i < n; i++) {

if (arr[i] != arr[ind]) {

ind++;

arr[ind] = arr[i];

}

}

return ind + 1;

}  
};  
  
**Maximum Index**

class Solution {

public:

// arr[]: input array

// Function to find the maximum index difference.

int maxIndexDiff(vector<int>& arr) {

int n = arr.size();

vector<int> leftMin(n), rightMax(n);

leftMin[0] = arr[0];

for(int i = 1 ; i < n ; i++){

leftMin[i] = min(leftMin[i-1],arr[i]);

}

rightMax[n-1] = arr[n-1];

for(int i = n-2 ; i >= 0 ; i--){

rightMax[i] = max(rightMax[i+1],arr[i]);

}

int i = 0, j = 0;

int maxDiff = 0;

while(i < n and j < n){

if(leftMin[i] <= rightMax[j]){

maxDiff = max(maxDiff, j - i);

j++;

}else{

i++;

}

}

return maxDiff;

}

};  
  
**Wave Array**  
class Solution {

Public:

void convertToWave(vector<int>& arr) {

int n = arr.size();

for (int i = 0; i + 1 < n; i += 2) {

swap(arr[i], arr[i+1]);

}

}

};