Assignment-2

26 May 2023 16:13

Question 5

Given an integer array nums, find three numbers whose product is maximum and return the maximum product.

Example 1:

Input: nums = [1,2,3]

Output: 6

```
class Solution {
public:
    int maximumProduct(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        int n=nums.size();

        return max(nums[n-1]*nums[n-2]*nums[n-3],nums[0]*nums[1]*nums[n-1]);
    }
};
```

Question 6

Given an array of integers nums which is sorted in ascending order, and an integer target,

write a function to search target in nums. If target exists, then return its index. Otherwise,

return -1.

You must write an algorithm with O(log n) runtime complexity.

Input: nums = [-1,0,3,5,9,12], target = 9

Output: 4

Explanation: 9 exists in nums and its index is 4

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int n=nums.size();
        int low=0;
        int high=n-1;

        while(low<=high){
            int mid=(low+high)>>1;

            if(nums[mid]==target)
                return mid;

            else if(nums[mid]>target)
                high=mid-1;
            else
                low=mid+1;
        }
        return -1;
    }
};
```

Question 7

An array is monotonic if it is either monotone increasing or monotone decreasing.

An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j]. An array nums is

monotone decreasing if for all i <= j, nums[i] >= nums[j].

Given an integer array nums, return true if the given array is monotonic, or false otherwise.

Example 1:

Input: nums = [1,2,2,3]

Output: true

```
class Solution {
public:
    bool isMonotonic(vector<int>& nums) {
        int n=nums.size();
        if(n==1 || n==2)
            return true;

        bool inc=true;
```

```
        bool dec=true;
        for(int i=0;i<nums.size()-1;i++){
            if(inc){
                if(nums[i]<=nums[i+1])
                    inc=true;
                else
                    inc=false;
            }
            if(dec){
                if(nums[i]>=nums[i+1])
                    dec=true;
                else
                    dec=false;
            }

        }
        return (inc||dec);
    }
};
```

Question 4

You have a long flowerbed in which some of the plots are planted, and some are not.

However, flowers cannot be planted in adjacent plots.

Given an integer array flowerbed containing 0's and 1's, where 0 means empty and 1 means

not empty, and an integer n, return true if n new flowers can be planted in the flowerbed

without violating the no-adjacent-flowers rule and false otherwise.

Example 1:

Input: flowerbed = [1,0,0,0,1], n = 1

Output: true

```
class Solution {
public:
    bool canPlaceFlowers(vector<int>& flowerbed, int n) {

        int total=0;

        for(int i=0;i<flowerbed.size() && total<n;i++){
            if(flowerbed[i]==0){
                int next=(i==flowerbed.size()-1) ? 0 :flowerbed[i+1];
                int prev=(i==0) ? 0 : flowerbed[i-1];

                if(prev==0 && next==0){
                    flowerbed[i]=1;
                    total++;
                }
            }

        }
        return total==n;
    }
};
```

💡 **Question 1** Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2),..., (an, bn) such that the sum of min(ai, bi) for all i is maximized. Return the maximized sum.

Example 1: Input: nums = [1,4,3,2] Output: 4

Explanation: All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4 So the maximum possible sum is 4

```
class Solution {
public:
    int arrayPairSum(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        int sum=0;

        for(int i=0;i<nums.size()-1;i+=2){
            sum+=min(nums[i],nums[i+1]);
        }
       return sum;
    }
};
```

Question 2
Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she started to gain weight, so she visited a doctor. The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Alice likes her candies very much, and she wants to eat the maximum number of different types of candies while still following the doctor's advice. Given the integer array candyType of length n, return the maximum number of different types of candies she can eat if she only eats n / 2 of them.
Example 1: Input: candyType = [1,1,2,2,3,3]
Output: 3
Explanation: Alice can only eat 6 / 2 = 3 candies. Since there are only 3 types, she can eat one of each type.

```
class Solution {
public:
    int distributeCandies(vector<int>& candyType) {
        int n=candyType.size();
        unordered_map<int,int>mp;

        for(int i=0;i<n;i++){
            if(mp.find(candyType[i]) !=mp.end()){
                mp[candyType[i]]++;
            }
            else{
                mp[candyType[i]]=1;
            }
        }
        if(mp.size()>=n/2){
            return  n/2;
        }
        else
            return mp.size();
    }
};
```

Question 5 Given an integer array nums, find three numbers whose product is maximum and return the maximum product.
Example 1: Input: nums = [1,2,3]
Output: 6

```
class Solution {
public:
    int maximumProduct(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        int n=nums.size();

        return max(nums[n-1]*nums[n-2]*nums[n-3],nums[0]*nums[1]*nums[n-1]);
         }
};
```

Question 3 We define a harmonious array as an array where the difference between its maximum value and its minimum value is exactly 1. Given an integer array nums, return the length of its longest harmonious subsequence among all its possible subsequences. A subsequence of an array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements.
Example 1:
Input: nums = [1,3,2,2,5,2,3,7]
Output: 5
Explanation: The longest harmonious subsequence is [3,2,2,2,3].

```
class Solution {
public:
    int findLHS(vector<int>& nums) {
        unordered_map<int,int>mp;

        for(auto i:nums){
            mp[i]++;
        }
        int maxi=0;

        for(auto [num,val]:mp){
            if(mp.find(num+1) !=mp.end()){
                maxi=max(maxi,val+mp[num+1]);
            }
        }
        return maxi;
 }
};
```