Assignment-3

26 May 2023   23:06

Question 5

You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return the resulting array of digits.

Example 1:

Input: digits = [1,2,3]

Output: [1,2,4]

```
class Solution {
public:
    vector<int> plusOne(vector<int>& digits) {
        int n=digits.size();
        for(int i=n-1;i>=0;i--){
            if(digits[i]<9){
                digits[i]++;
                return digits;
            }

            digits[i]=0;
        }
        vector<int>new_number(n+1,0);
        new_number[0]=1;
        return new_number;
    }
};
```

Question 6

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

Input: nums = [2,2,1]

Output: 1

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int ans=0;

        for(auto i:nums){
            ans^=i;
        }
        return ans;
    }
};
```

Question 8

 Given an array of meeting time intervals where intervals[i] = [starti, endi], determine if a person could attend all meetings.

Example 1: Input: intervals = [[0,30],[5,10],[15,20]]
Output: false

```
class Solution {
public:
static bool cmp(pair<int,int> a, pair<int,int> b){
        return a.first<b.first;
    }
    bool canAttendMeetings(vector<Interval>& intervals) {
```

```
        if(intervals.size()==0 || intervals.size())
                return true;

            sort(intervals.begin(),intervals.end(),cmp);

            for(int i=1;i<n;i++){
                if(i[i].start < [i-1].end)
                    return false;
            }
            return true;
        }
};
```

Question 4 Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order. You must write an algorithm with O(log n) runtime complexity.
 Example 1:
Input: nums = [1,3,5,6],
 target = 5
 Output: 2

```
class Solution {
public:
  int searchInsert(vector<int>& nums, int target) {
    int start=0;
    int end=nums.size()-1;

    while(start<=end){
      int mid=(start+end)>>1;

      if(nums[mid]==target){
        return mid;
      }
      else if(nums[mid]>target){
        end=mid-1;
      }
      else{
        start=mid+1;
      }
    }
    return start;
  }
};
```

Question 7 You are given an inclusive range [lower, upper] and a sorted unique integer array nums, where all elements are within the inclusive range. A number x is considered missing if x is in the range [lower, upper] and x is not in nums. Return the shortest sorted list of ranges that exactly covers all the missing numbers. That is, no element of nums is included in any of the ranges, and each missing number is covered by one of the ranges.
 Example 1: Input: nums = [0,1,3,50,75], lower = 0, upper = 99
Output: [[2,2],[4,49],[51,74],[76,99]]
Explanation: The ranges are: [2,2] [4,49] [51,74] [76,99]

```
class Solution {
  string formatRange(int lower, int upper) {
    if (lower == upper) return to_string(lower);
    return to_string(lower) + "->" + to_string(upper);
  }
public:
  vector<string> findMissingRanges(vector<int>& A, int lower, int upper) {
    vector<string> ans;
    int i = lower, j = 0, N = A.size();
    while (j < N) {
      if (i == A[j]) {
        ++i;
        ++j;
      } else {
        int first = i, last = A[j] - 1;
```

```
        ans.push_back(formatRange(first, last));

        i = A[j++] + 1;

      }

    }

    if (i <= upper) ans.push_back(formatRange(i, upper));

    return ans;

  }

};
```

Question 1 Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to the target. Return the sum of the three integers. You may assume that each input would have exactly one solution.
 Example 1: Input: nums = [-1,2,1,-4], target = 1
Output: 2
Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

```
class Solution {

public:

  int threeSumClosest(vector<int>& nums, int target) {

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

    int n=nums.size();

    int ans=nums[0]+nums[1]+nums[2];

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

      int j=i+1;

      int k=n-1;

      while(j<k){

        int sum=nums[i]+nums[j]+nums[k];

        if(abs(sum-target)<abs(ans-target)){

          ans=sum;

        }

        if(sum==target){

          return sum;

        }

        else if(sum>target){

          k--;

        }

        else

          j++;

      }

    }

    return ans;

  }

};
```

Question 2 Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that: ● 0 <= a, b, c, d < n ● a, b, c, and d are distinct. ● nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order.
 Example 1: Input: nums = [1,0,-1,0,-2,2],
target = 0
 Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

```
class Solution {

public:

  vector<vector<int>> fourSum(vector<int>& nums, int target) {

    vector<vector<int>>ans;

    int n=nums.size();

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

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

      if(i>0 && nums[i]==nums[i-1])

        continue;

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

        if(j>i+1 && nums[j]==nums[j-1])
```

```
            continue;
        int k=j+1;
        int l=n-1;

        while(k<l){
            long long sum=nums[i];
            sum+=nums[j];
            sum+=nums[k];
            sum+=nums[l];

            if(sum==target){
                vector<int> temp={nums[i],nums[j],nums[k],nums[l]};
                ans.push_back(temp);
                k++;
                l--;
                while(k<l && nums[k]==nums[k-1])
                    k++;

                while(k<l && nums[l]==nums[l+1])
                    l--;

            }

            else if(sum>target)
                l--;
            else
                k++;

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

💡 **Question 3** A permutation of an array of integers is an arrangement of its members into a sequence or linear order.

For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows it in the sorted container.

If such an arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

● For example, the next permutation of arr = [1,2,3] is [1,3,2]. ● Similarly, the next permutation of arr = [2,3,1] is [3,1,2]. ● While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, find the next permutation of nums. The replacement must be in place and use only constant extra memory.

Example 1: Input: nums = [1,2,3] Output: [1,3,2]class Solution {

```
public:
  void nextPermutation(vector<int>& nums) {
    int ind=-1;
    int n=nums.size();
    for(int i=n-1;i>0;i--){
      if(nums[i]>nums[i-1]){
        ind=i;
        break;
      }
    }
    if(ind==-1){
      reverse(nums.begin(),nums.end());
    }
    else{
```

```
        int prev=ind;
        for(int i=ind+1;i<n;i++){
          if(nums[i]>nums[ind-1] && nums[i]<=nums[prev]){
            prev=i;
          }
        }
        swap(nums[ind-1],nums[prev]);
        reverse(nums.begin()+ind,nums.end());
      }

    }
};
```

```
        int prev=ind;
        for(int i=ind+1;i<n;i++){
          if(nums[i]>nums[ind-1] && nums[i]<=nums[prev]){
            prev=i;
          }
        }
        swap(nums[ind-1],nums[prev]);
        reverse(nums.begin()+ind,nums.end());
```