Striver sde sheet

485. Max Consecutive Ones

C++ Autocomplete

```cpp
class Solution {
public:
    int findMaxConsecutiveOnes(vector<int>& nums) {
        int count=0;
        int maxa=INT_MIN;
        for(int i=0;i<nums.size();i++)
        {if(nums[i]==1)
            {
                count++;
                maxa=max(maxa, count);
            }
        else
            count=0;}
        if(maxa==INT_MIN)
            return 0;
        return maxa;
    }
};
```

26. Remove Duplicates from Sorted Array

C++ Autocomplete

```cpp
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if(nums.size() == 0) return 0;
        int left = 0;
        for(int right =1; right< nums.size(); right++){
           if(nums[left] != nums[right])
               left++;
               nums[left] = nums[right];
           }
    return left+1;
    }
};
```

15. 3Sum

C++ Autocomplete

```cpp
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        int n= nums.size();
        int h=0;
        int target;
        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());
    for(int i=0;i<n;i++)
    { if(i==0 ||( i>0 && nums[i-1]!=nums[i]))
    {int lo=i+1;
    int h=nums.size()-1;
     int sum=-nums[i];
     while(lo<h)
     {target=nums[lo]+nums[h];
      if(target>sum)
          h--;
      else if(target<sum)
          lo++;
      else
      { vector<int> res(3);
          res[0]=nums[i];
          res[1]=nums[lo];
          res[2]=nums[h];
          ans.push_back(res);
          while(lo<h && nums[lo] == nums[lo+1])lo=lo+1;
          while(lo<h && nums[h] == nums[h-1])h=h-1;
          lo++;
          h--;

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

61. Rotate List

C++ Autocomplete

```
 * };
 */
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
       if(!head || !head->next || k==0)
           return head;
        int counta=1,pos;
        ListNode *ss;
        ListNode *temp=head;
        while(temp->next!=NULL)
        { temp=temp->next;
            counta++;}
        k=k%counta;
        if(k==0)
            return head;
        pos=counta-k;
        counta=1;
        temp=head;
        while(counta!=pos)
        { temp=temp->next;
            counta++;}
        ss=temp->next;
        temp->next=NULL;
        temp=ss;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=head;
        head=ss;
        return head;
    }
};
```

994. Rotting Oranges

C++ Autocomplete

```cpp
class Solution {
public:
    int orangesRotting(vector<vector<int>>& grid) {
        if(grid.size()==0)return 0;
        int row=grid.size();
        int col=grid[0].size();
        int total=0;
        queue<pair<int,int>> q;
        vector<vector<int>> visited(row, vector<int> (col,0));
        for(int i=0;i<row;i++)
        {for(int j=0;j<col;j++)
            {if(grid[i][j]==1 || grid[i][j]==2)total++;
                if(grid[i][j]==2)q.push({i,j}); }}
        int dx[4]={0,0,1,-1};
        int dy[4]={1,-1,0,0};
        int count=0;
        int ans=0;
        int size=0;
        int u,v;
        int t=0;
        while(!q.empty())
              {size=q.size();
               t+=size;
               while(size--)
               {auto node=q.front();
               int f=node.first;
               int s=node.second;
               q.pop();
               for(int i=0;i<4;i++)
               { u=f+dx[i];
                   v=s+dy[i];
                   if(u<0 || v<0 || u>=row || v>=col)
                   continue;
                 if(grid[u][v]==1)
```

C++ Autocomplete

```
        int count=0;
        int ans=0;
        int size=0;
        int u,v;
        int t=0;
        while(!q.empty())
              {size=q.size();
               t+=size;
               while(size--)
               {auto node=q.front();
               int f=node.first;
               int s=node.second;
               q.pop();
               for(int i=0;i<4;i++)
               { u=f+dx[i];
                   v=s+dy[i];
                   if(u<0 || v<0 || u>=row || v>=col)
                   continue;
                 if(grid[u][v]==1)
                 {grid[u][v]=2;
                  q.push({u,v});
                 }
               }

               }
               if(!q.empty())ans++;
              }

        if(t==total)
            return ans;
        else
            return -1;
    }
};
```

146. LRU Cache

C++ • Autocomplete

```cpp

class LRUCache {
public:
    class Node{
        public:
        int key;
        int val;
        Node* next;
        Node* prev;
        Node(int k, int v)
        {  key=k;
        val=v;
         next=NULL;
         prev=NULL;

        }
    };
    unordered_map<int, Node*> mpp;
    Node* head=new Node(-1, -1);
     Node* tail=new Node(-1, -1);
    int cap;

    LRUCache(int capacity) {
        cap=capacity;
        head->next=tail;
        tail->prev=head;


    }
   void deletee(Node* node)
    {
       Node *pre=node->prev;
     Node *post=node->next;
     pre->next=post;
```

Your previous code was restored from your local storage. Rese

```
    }
   void deletee(Node* node)
    {
       Node *pre=node->prev;
     Node *post=node->next;
     pre->next=post;
     post->prev=pre;

    }
    void add(Node* node)
    {cout<<"y";
        Node* l=head->next;
        head->next=node;
        node->prev=head;
        node->next=l;
        l->prev=node;
    //  Node * temp = head -> next;
    // newnode -> next = temp;
    // newnode -> prev = head;
    // head -> next = newnode;
    // temp -> prev = newnode;
    }

    int get(int key) {

        if(mpp.find(key)==mpp.end())
        {return -1;

        }
        else
        {Node* u=mpp[key];

            deletee(mpp[key]);
```

C++ • Autocomplete

```
        {return -1;

        }
        else
        {Node* u=mpp[key];

            deletee(mpp[key]);
            add(u );
         mpp[key]=u;

        }
        return mpp[key]->val;

    }


    void put(int key, int value) {


        if(mpp.find(key)!=mpp.end())
        {  Node* u=mpp[key];
         mpp.erase(key);
            deletee(u);

        }
         if(mpp.size()==cap)
        {mpp.erase(tail->prev->key);
         deletee(tail->prev);

        }
        add(new Node(key, value));
        mpp[key]=head->next;
    }
};
```

496. Next Greater Element I

C++ • Autocomplete

```cpp
class Solution {
public:
    vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {
    unordered_map<int, int> mpp;
        stack<int> s;
        vector<int> ans;
        for(int i=nums2.size()-1;i>=0;i--)
        {
//         {if(s.size()==0)
//         {

//         }
            if(!s.empty() && s.top()>nums2[i])
            mpp[nums2[i]]=s.top();
         else if( !s.empty() && s.top()<nums2[i])
         {
             while(!s.empty() && s.top()<nums2[i] )
             {
                 s.pop();
             }
             if(!s.empty())
                 mpp[nums2[i]]=s.top();
         }
        s.push(nums2[i]);
        }
            for(auto it: nums1)
            {
            ans.push_back(mpp.count(it) ? mpp[it] : -1);
            }

        return ans;}
};
```

20. Valid Parentheses

C++ • Autocomplete

```cpp
class Solution {
public:
    bool isValid(string s) {
        stack<char> st;
        for(auto it: s)
        {
            if(it=='(' || it=='[' || it=='{')
            st.push(it);
            else
            { if(st.empty())
                return false;
                else if(st.top()=='('  && it!=')')
                    return false;
                else if(st.top()=='{'  && it!='}')
                    return false;
                    else if(st.top()=='['  && it!=']')
                        return false;
                else
                    st.pop();
            }
        }
        if(st.empty()==0)
            return false;
        return true;
    }
};
```

347. Top K Frequent Elements

C++ • Autocomplete

```cpp
class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        priority_queue<pair<int, int>> maxh;
        unordered_map<int, int> mpp;
        for(auto it: nums)
        {
            mpp[it]++;
        }
        for(auto it: mpp)
        {
            maxh.push({it.second, it.first});
        }
        vector<int> ans;
        int i=0;
        while(i++<k)
        {ans.push_back(maxh.top().second);
         maxh.pop();

        }
        return ans;
    }
};
```

215. Kth Largest Element in an Array

C++ Autocomplete

```cpp
class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {
        priority_queue<int, vector<int> , greater<int>> maxh;
        for(auto it: nums)
        { maxh.push(it);
            if(maxh.size()>k) maxh.pop();}
        return maxh.top();
    }
};
```

540. Single Element in a Sorted Array

```
class Solution {
public:
    int singleNonDuplicate(vector<int>& nums) {
        if(nums.size()==1)return nums[0];
        int low=0;
        int high=nums.size()-1;
        while(low<=high)
        {
            int mid=low+(high-low)/2;
            if(mid%2==0)
            { if(nums[mid]==nums[mid+1])
                low=mid+1;
             else
                 high=mid-1;

            }
            else
            {  if(nums[mid]==nums[mid-1])
                low=mid+1;
             else
                 high=mid-1;

            }
        }
        return nums[low];
    }
};
```

46. Permutations

C++ Autocomplete

```cpp
class Solution {
public:
    void rev(int c, vector<int> nums, vector<int> box,vector<vector<int>>& ans)
    {
        if(c==nums.size())
        {
            ans.push_back(box);
            return;
        }
        for(int i=0;i<nums.size();i++)
        { if(box[i]==-11)
        {
            box[i]=nums[c];
            rev(c+1, nums, box, ans);
            box[i]=-11;
        }

        }
    }
    vector<vector<int>> permute(vector<int>& nums) {
        int n=nums.size();
        vector<int> box(n,-11);
        vector<vector<int>> ans;
        rev(0, nums, box, ans);
        return ans;
    }
};
```

40. Combination Sum II

```
12
13  class Solution {
14  public:
15       void rev(int idx, vector<int> c, int target, int sum, vector<int> temp,
    set<vector<int>>& ans)
16       {if(sum>target || (idx==c.size() && sum!=target))
17              return;
18           if(target==sum)
19           { ans.insert(temp);
20              return; }
21           for(int i=idx;i<c.size();i++)
22           {  if(i>idx && c[i-1]==c[i])
23              continue;
24            temp.push_back(c[i]);
25            rev(i+1, c, target, sum+c[i],temp,ans);
26            temp.pop_back();}}
27       vector<vector<int>> combinationSum2(vector<int>& c, int target) {
28         set<vector<int>> ans;
29           sort(c.begin(), c.end());
30           vector<int> temp;
31           rev(0, c,target, 0, temp, ans);
32           vector<vector<int>> a;
33           for(auto it: ans)
34           {
35               a.push_back(it);
36           }
37           return a;
38       }
39  };
```

Your previous code was restored from your local storage. Reset to default

39. Combination Sum

C++ • Autocomplete

```cpp
class Solution {
public:
    void comb(int index, int target, vector<vector<int>>& ans, vector<int>& temp,
vector<int>& arr)
    {
        if(index==arr.size())
        {
            if(target==0)
            {
                ans.push_back(temp);}
            return;
        }
        if(arr[index]<=target)
        {temp.push_back(arr[index]);
         comb(index, target-arr[index], ans, temp,arr);
         temp.pop_back();
        }
        comb(index+1, target, ans, temp, arr);
    }
    vector<vector<int>> combinationSum(vector<int>& arr, int target) {
        vector<vector<int>> ans;
        vector<int> temp;
        comb(0, target, ans, temp, arr);
        return ans;
    }
};
```

90. Subsets II

C++ Autocomplete

```cpp
class Solution {
public:
    vector<vector<int>> subsetsWithDup(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int total=1<<nums.size();
        total=total-1;
      set<vector<int>> a;
        for(int i=0;i<=total;i++)
        {  vector<int> ans;
            for(int bits=0;bits<nums.size();bits++)
            {
                if((i&(1<<bits)))
                    ans.push_back(nums[bits]);

            }
         a.insert(ans);
        }
        vector<vector<int>> aa;
        for(auto it: a)
            aa.push_back(it);
        return aa;
    }
};
```