Striver sde sheet

485. Max Consecutive Ones

```
i C++

    Autocomplete

        class Solution {
   2
        public:
            int findMaxConsecutiveOnes(vector<int>& nums) {
   3 ₹
                 int count=0;
   4
   5
                 int maxa=INT_MIN;
                 for(int i=0; i<nums.size(); i++)</pre>
   6
   7 🔻
                 \{if(nums[i]==1)
   8 *
   9
                          count++;
                          maxa=max(maxa, count);
 10
 11
                 else
 12
 13
                     count=0;}
                 if(maxa==INT_MIN)
 14
 15
                     return 0;
 16
                 return maxa;
 17
             }
 18
        };
```

26. Remove Duplicates from Sorted Array

```
i C++

    Autocomplete

        class Solution {
   1 *
        public:
   2
            int removeDuplicates(vector<int>& nums) {
   3 ▼
   4
                 if(nums.size() == 0) return 0;
   5
                 int left = 0;
                 for(int right =1; right< nums.size(); right++){</pre>
  6 ▼
                    if(nums[left] != nums[right])
   7
  8
                        left++;
                        nums[left] = nums[right];
  9
 10
            return left+1;
 11
 12
 13
        };
```

15.3Sum

```
i C++

    Autocomplete

        class Solution {
   1 •
        public:
   2
   3 ₹
             vector<vector<int>> threeSum(vector<int>& nums) {
   4
                 int n= nums.size();
                 int h=0;
   5
                 int target;
   6
   7
                 vector<vector<int>> ans;
   8
                 sort(nums.begin(), nums.end());
   9
             for(int i=0;i<n;i++)</pre>
             { if(i=0 ||( i>0 && nums[i-1]!=nums[i]))
  10 ▼
             {int lo=i+1;
  11 ▼
  12
             int h=nums.size()-1;
  13
              int sum=-nums[i];
              while(lo<h)
  14
              {target=nums[lo]+nums[h];
 15 ▼
  16
               if(target>sum)
  17
                   h--;
               else if(target<sum)</pre>
  18
  19
                   10++;
  20
               else
  21 *
               { vector<int> res(3);
  22
                   res[0]=nums[i];
  23
                   res[1]=nums[lo];
  24
                   res[2]=nums[h];
  25
                   ans.push back(res);
                   while(lo<h && nums[lo] == nums[lo+1])lo=lo+1;
  26
  27
                   while(lo<h && nums[h] == nums[h-1])h=h-1;</pre>
  28
                   10++;
  29
                   h--;
  30
               } } }
  31
  32
                 return ans;
  33
             }
        };
  34
```

61. Rotate List

```
i C++

    Autocomplete

  9
         */
 10
 11 ▼
        class Solution {
 12
        public:
            ListNode* rotateRight(ListNode* head, int k) {
 13 ▼
 14
               if(!head || !head->next || k==0)
                    return head;
 15
                int counta=1,pos;
 16
 17
                ListNode *ss;
                ListNode *temp=head;
 18
                while(temp->next!=NULL)
 19
 20 ▼
                 { temp=temp->next;
 21
                     counta++;}
 22
                k=k%counta;
 23
                if(k==0)
 24
                     return head;
 25
                 pos=counta-k;
 26
                 counta=1;
 27
                temp=head;
                while(counta!=pos)
 28
                 { temp=temp->next;
 29 🔻
 30
                     counta++;
 31
                 ss=temp->next;
 32
                temp->next=NULL;
 33
                temp=ss;
                while(temp->next!=NULL)
 34
 35 ▼
                 {
 36
                     temp=temp->next;
 37
                temp->next=head;
 38
 39
                head=ss;
 40
                 return head;
 41
 42
        };
```

994. Rotting Oranges

```
i C++

    Autocomplete

        class Solution {
        public:
   2
            int orangesRotting(vector<vector<int>>& grid) {
   3 ₹
                 if(grid.size()==0)return 0;
   4
   5
                int row=grid.size();
   6
                int col=grid[0].size();
   7
                int total=0;
                queue<pair<int,int>> q;
  8
                vector<vector<int>> visited(row, vector<int> (col,0));
  9
                for(int i=0;i<row;i++)</pre>
 10
 11 ·
                {for(int j=0; j<col; j++)
                     {if(grid[i][j]==1 || grid[i][j]==2)total++;
 12 *
                         if(grid[i][j]==2)q.push({i,j}); }}
 13
                int dx[4]=\{0,0,1,-1\};
 14
 15
                int dy[4]=\{1,-1,0,0\};
 16
                int count=0;
 17
                int ans=0;
 18
                int size=0;
 19
                int u,v;
 20
                int t=0;
 21
                while(!q.empty())
                       {size=q.size();
 22 🔻
 23
                        t+=size;
                        while(size--)
 24
 25 *
                        {auto node=q.front();
                        int f=node.first;
 26
                        int s=node.second;
 27
                        q.pop();
 28
                        for(int i=0;i<4;i++)
 29
 30 ▼
                        { u=f+dx[i];
 31
                            v=s+dy[i];
                            if(u<0 || v<0 || u>=row || v>=col)
 32
 33
                            continue;
 34
                          if(grid[u][v]==1)
```

```
i C++

    Autocomplete

  16
                 int count=0;
  17
                 int ans=0;
  18
                 int size=0;
  19
                 int u,v;
  20
                 int t=0;
  21
                 while(!q.empty())
  22 *
                        {size=q.size();
  23
                         t+=size;
  24
                        while(size--)
  25 ▼
                         {auto node=q.front();
  26
                         int f=node.first;
  27
                         int s=node.second;
  28
                         q.pop();
                         for(int i=0;i<4;i++)
  29
  30 ▼
                         { u=f+dx[i];
  31
                             v=s+dy[i];
  32
                             if(u<0 || v<0 || u>=row || v>=col)
  33
                             continue;
  34
                           if(grid[u][v]==1)
  35 ▼
                           {grid[u][v]=2;
  36
                            q.push({u,v});
  37
                         }
  38
  39
  40
  41
                        if(!q.empty())ans++;
  42
  43
  44
                 if(t==total)
  45
                     return ans;
  46
                 else
  47
                     return -1;
  48
             }
  49
        };
```

146. LRU Cache

```
i C++

    Autocomplete

   1
        class LRUCache {
   2 🔻
   3
        public:
            class Node{
   4 ▼
   5
                 public:
                 int key;
   6
   7
                 int val;
   8
                 Node* next;
                 Node* prev;
   9
                 Node(int k, int v)
  10
  11 v
                 { key=k;
                 val=v;
 12
 13
                 next=NULL;
 14
                  prev=NULL;
  15
                 }
  16
  17
             };
            unordered_map<int, Node*> mpp;
  18
            Node* head=new Node(-1, -1);
  19
             Node* tail=new Node(-1, -1);
  20
  21
            int cap;
  22
  23 ▼
            LRUCache(int capacity) {
  24
                 cap=capacity;
  25
                 head->next=tail;
  26
                 tail->prev=head;
  27
  28
  29
            }
           void deletee(Node* node)
  30
  31 ▼
  32
                Node *pre=node->prev;
  33
             Node *post=node->next;
  34
             pre->next=post;
```

Your previous code was restored from your local storage. Rese

```
28
19
         void deletee(Node* node)
30
31 *
             Node *pre=node->prev;
32
33
           Node *post=node->next;
34
           pre->next=post;
35
           post->prev=pre;
36
37
          void add(Node* node)
38
          {cout<<"y";
}9 ▼
10
              Node* l=head->next;
              head->next=node;
1
12
              node->prev=head;
13
              node->next=1;
14
              1->prev=node;
          // Node * temp = head -> next;
15
          // newnode -> next = temp;
16
          // newnode -> prev = head;
17
          // head -> next = newnode;
18
19
          // temp -> prev = newnode;
0
51
52 ₹
          int get(int key) {
53
14
              if(mpp.find(key)==mpp.end())
55 ₹
              {return -1;
66
57
58
              else
i9 ₹
              {Node* u=mpp[key];
50
                   deletee(mpp[key]);
51
```

our previous code was restored from your local storage. Reset to

```
C++

    Autocomplete

55 ▼
               {return -1;
56
57
               else
58
59 ▼
               {Node* u=mpp[key];
60
                   deletee(mpp[key]);
61
                   add(u);
62
                mpp[key]=u;
63
64
65
66
               return mpp[key]->val;
67
          }
68
69
70
          void put(int key, int value) {
71 🔻
72
73
74
               if(mpp.find(key)!=mpp.end())
75 v
               { Node* u=mpp[key];
                mpp.erase(key);
76
77
                   deletee(u);
78
79
                if(mpp.size()==cap)
80
               {mpp.erase(tail->prev->key);
81 ▼
                deletee(tail->prev);
82
83
84
               add(new Node(key, value));
85
86
               mpp[key]=head->next;
87
88
      };
29
```

496. Next Greater Element I

```
i C++
                                                                            i {} 5 ⊙

    Autocomplete

        class Solution {
   1 •
        public:
   2
   3 🔻
            vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {
            unordered_map<int, int> mpp;
   4
   5
                 stack<int> s;
   6
                 vector<int> ans;
   7
                 for(int i=nums2.size()-1;i>=0;i--)
   8 🕶
        //
//
                    {if(s.size()==0)
   9
  10
 11
        //
 12
 13
                     if(!s.empty() && s.top()>nums2[i])
                     mpp[nums2[i]]=s.top();
 14
                  else if( !s.empty() && s.top()<nums2[i])</pre>
 15
 16 ▼
                      while(!s.empty() && s.top()<nums2[i] )</pre>
 17
 18 ▼
                      {
 19
                          s.pop();
  20
                      if(!s.empty())
  21
                          mpp[nums2[i]]=s.top();
  22
  23
 24
                 s.push(nums2[i]);
 25
                     for(auto it: nums1)
 26
 27 •
                     ans.push back(mpp.count(it) ? mpp[it] : -1);
 28
  29
  30
  31
                 return ans;}
  32
        };
```

20. Valid Parentheses

```
C++

    Autocomplete

      class Solution {
 1 v
 2
      public:
          bool isValid(string s) {
 3 ▼
 4
               stack<char> st;
 5
               for(auto it: s)
 6 ▼
 7
                   if(it=='(' || it=='[' || it=='{')
 8
                   st.push(it);
 9
                   else
10 •
                   { if(st.empty())
                       return false;
11
12
                       else if(st.top()=='('
                                               && it!=')')
13
                           return false;
                       else if(st.top()=='{' && it!='}')
14
15
                           return false;
                           else if(st.top()=='[' && it!=']')
16
17
                                return false;
18
                       else
19
                           st.pop();
20
                   }
21
22
               if(st.empty()==0)
23
                   return false;
24
               return true;
25
          }
      };
26
```

347. Top K Frequent Elements

```
Autocomplete
C++
      class Solution {
 1 v
 2
      public:
          vector<int> topKFrequent(vector<int>& nums, int k) {
 3 ₹
 4
              priority_queue<pair<int, int>> maxh;
 5
              unordered_map<int, int> mpp;
              for(auto it: nums)
 6
 7 🔻
                   mpp[it]++;
8
9
              for(auto it: mpp)
10
11 ▼
                  maxh.push({it.second, it.first});
12
13
14
              vector<int> ans;
              int i=0;
15
              while(i++<k)
16
              {ans.push back(maxh.top().second);
17 v
18
               maxh.pop();
19
20
21
              return ans;
22
          }
23
      };
```

215. Kth Largest Element in an Array

```
    Autocomplete

i C++
        class Solution {
   1 •
   2
        public:
   3 ▼
             int findKthLargest(vector<int>& nums, int k) {
                 priority_queue<int, vector<int> , greater<int>> maxh;
   4
   5
                 for(auto it: nums)
   6 ▼
                 { maxh.push(it);
                     if(maxh.size()>k) maxh.pop();}
   7
   8
                 return maxh.top();
   9
  10
        };
```

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++
                                                                          i {} 5 ⊙

    Autocomplete

  1 ▼
       class Solution {
       public:
  2
            void rev(int c, vector<int> nums, vector<int> box,vector<vector<int>>& ans)
  3
  4 ▼
  5
                if(c==nums.size())
  6 ₹
                    ans.push_back(box);
  7
                    return;
  8
  9
                for(int i=0;i<nums.size();i++)</pre>
 10
                \{ if(box[i]==-11) \}
 11 ▼
 12 v
                    box[i]=nums[c];
 13
                    rev(c+1, nums, box, ans);
 14
 15
                    box[i]=-11;
                }
 16
 17
 18
                }
 19
 20 ▼
            vector<vector<int>> permute(vector<int>& nums) {
 21
                int n=nums.size();
                vector<int> box(n,-11);
 22
 23
                vector<vector<int>> ans;
                rev(0, nums, box, ans);
 24
 25
                return ans;
 26
 27
       };
```

40. Combination Sum II

```
12
13 🔻
      class Solution {
14
      public:
           void rev(int idx, vector<int> c, int target, int sum, vector<int> temp,
15
      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++)</pre>
               { if(i>idx && c[i-1]==c[i])
22 *
                    continue;
23
24
                temp.push back(c[i]);
25
                rev(i+1, c, target, sum+c[i],temp,ans);
                temp.pop_back();}}
26
          vector<vector<int>> combinationSum2(vector<int>& c, int target) {
27 ▼
28
            set<vector<int>> ans;
29
              sort(c.begin(), c.end());
30
              vector<int> temp;
              rev(0, c,target, 0, temp, ans);
31
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

X

39. Combination Sum

```
i C++ ▼ • Autocomplete
                                                                       i {} ○ ② □
   1 •
        class Solution {
   2
        public:
            void comb(int index, int target, vector<vector<int>>& ans, vector<int>& temp,
   3
        vector<int>& arr)
   4 *
            {
   5
                if(index==arr.size())
  6 ▼
   7
                    if(target==0)
  8 🔻
                    {
  9
                        ans.push_back(temp);}
  10
                    return;
  11
                if(arr[index]<=target)</pre>
  12
                {temp.push back(arr[index]);
  13 ▼
                 comb(index, target-arr[index], ans, temp,arr);
  14
  15
                 temp.pop_back();
  16
 17
                comb(index+1, target, ans, temp, arr);
 18
            vector<vector<int>> combinationSum(vector<int>& arr, int target) {
  19 ▼
  20
                vector<vector<int>> ans;
  21
                vector<int> temp;
                comb(0, target, ans, temp, arr);
  22
  23
                return ans;
  24
            }
  25
        };
```

90. Subsets II

```
i C++

    Autocomplete

   1 •
         class Solution {
   2
         public:
             vector<vector<int>>> subsetsWithDup(vector<int>& nums) {
   3 ▼
   4
                  sort(nums.begin(), nums.end());
   5
                  int total=1<<nums.size();</pre>
                  total=total-1;
   6
   7
               set<vector<int>> a;
   8
                  for(int i=0;i<=total;i++)</pre>
   9 ▼
                    vector<int> ans;
                      for(int bits=0;bits<nums.size();bits++)</pre>
  10
  11 v
                          if((i&(1<<bits)))
  12
  13
                               ans.push_back(nums[bits]);
  14
  15
  16
                  a.insert(ans);
  17
  18
                  vector<vector<int>> aa;
  19
                  for(auto it: a)
  20
                      aa.push_back(it);
  21
                  return aa;
  22
  23
         };
```