STRIVER SDE SHEET

Set matrix zeros

C++ Autocomplete

```cpp
class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {
        int m=matrix.size();
        int n=matrix[0].size();
        int colo=1;
        for(int i=0;i<m;i++)
        {  if(matrix[i][0]==0)colo=0;
            for(int j=1;j<n;j++)
            {
                if(matrix[i][j]==0)
                {matrix[0][j]=0;
                 matrix[i][0]=0;}
            }
        }

         for(int i=m-1;i>=0;i--)
        {
            for(int j=n-1;j>=1;j--)
            {
                if(matrix[i][0]==0 || matrix[0][j]==0)
                    matrix[i][j]=0;

            }
             if(colo==0)
                 matrix[i][0]=0;
        }
    }
};
```

Pascal Traingle

C++ Autocomplete

```cpp
class Solution {
public:
    vector<vector<int>> generate(int n) {
        vector<vector<int>> ans;
        ans.push_back({1});
        if(n==1)return ans;
        ans.push_back({1,1});
        if(n==2)return ans;
        cout<<ans.size();
        for(int i=3;i<=n;i++)
        {
            vector<int> level;
            level.push_back(1);
            for(int j=0;j<i-2;j++)
                level.push_back(ans[ans.size()-1][j]+ans[ans.size()-1][j+1]);
           // cout<<1;
            level.push_back(1);
            ans.push_back(level);
        }
        return ans;
    }
};
```

Next Permutation

C++ Autocomplete

```cpp
class Solution {
public:
    void swap(vector<int>& arr, int i,int j)
    {
        int temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
    }
    void reverse(vector<int>& arr, int i, int j )
    {
        while(i<j)swap(arr, i++, j--);
    }
    void nextPermutation(vector<int>& arr) {
        int j;
        if(arr.size()==0 || arr.size()==1)
            return ;
       int i=arr.size()-2;
        while(i>=0 && arr[i]>=arr[i+1])i--;
        if(i>=0)
        {
    j=arr.size()-1;
            while(arr[i]>=arr[j])j--;
            swap(arr, i,j);
        }
        reverse(arr, i+1, arr.size()-1);

    }
};
```

Kadanes Algorithm

C++ Autocomplete

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

Sort arrays of 0s, 1s, and 2s.

C++ Autocomplete

```cpp
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low=0;
        int mid=0;
        int high=nums.size()-1;
        while(mid<=high)
        {switch(nums[mid])
        {
            case 0:  swap(nums[low++], nums[mid++]);
                break;
            case 1:   mid++;
                break;
            case 2:  swap(nums[mid], nums[high--]);
        }

        }

    }
};
```

Buy and sell stock

C++ Autocomplete

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int buy=prices[0];
    int pro=0;
        for(int i=1;i<prices.size();i++)
        {
            if(prices[i]> buy)
                pro=max(pro, prices[i]-buy);
            else
                buy=prices[i];
        }
        return pro;
    }
};
```

Rotate matrix

C++ Autocomplete

```cpp
class Solution {
public:
    //transpose matrix and reverse
    void rotate(vector<vector<int>>& mat) {
        int m=mat.size();
        int n=mat[0].size();
       for(int i=0;i<m;i++)
       {for(int j=i;j<n;j++)
       {swap(mat[i][j], mat[j][i]);
       }
       }
        for(int i=0;i<m;i++)
    {     reverse(mat[i].begin(),mat[i].end());


        }

    }
};
```

Merge Intervals

C++ • Autocomplete

```cpp
class Solution {
public:
    static bool comp(vector<int> a,vector<int> b)
    { return a[0]<b[0];}
    vector<vector<int>> merge(vector<vector<int>>& i) {
        sort(i.begin(), i.end(),comp);
        vector<vector<int>> ans;
        ans.push_back(i[0]);
        int end=i[0][1];
        for(int j=1;j<i.size();j++)
        {if(i[j][0]<=end)
        {ans[ans.size()-1][1]=max(i[j][1],ans[ans.size()-1][1]);
         end=max(i[j][1],ans[ans.size()-1][1]);
        }
         else
         {
             ans.push_back(i[j]);
             end=i[j][1];
         }}
        return ans;
    }
};
```

Merge two sorted array without extra space

```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
//         int j=nums1.size();
//         for(inti=j;i<num1.size();i++)
//         {
//             nums1[i]=nums2[i-j];
//         }
        int i=m-1;
        int j=n-1;
        int k=m+n-1;
        while(i>=0 && j>=0)
        {
            if(nums1[i]<nums2[j])
            {
                nums1[k--]=nums2[j--];
            }
            else
                 nums1[k--]=nums1[i--];

        }
        while(j>=0)
        {nums1[k--]=nums2[j--];

        }

    }
};
```

Find duplicate array in array of N+1.

C++ • Autocomplete

```cpp
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int slow=nums[0];
        int fast=nums[0];
    do{
        fast=nums[nums[fast]];
         slow=nums[slow];

        }while(fast!=slow);
            fast=nums[0];
        while(fast!=slow)
        {
            fast=nums[fast];
            slow=nums[slow];
        }
        return fast;
    }
};
```

Search in 2D matrix

```cpp
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& mat, int target) {
        int low=0;
        int n=mat[0].size();
        int mid;
        int high=mat.size()*mat[0].size()-1;
        while(low<=high)
        {
            mid=low+(high-low)/2;
            if(mat[mid/n][mid%n]==target)
                return true;
            else if(mat[mid/n][mid%n]<target)
                low=mid+1;
            else
                high=mid-1;
        }
        return false;
    }
};
```

Pow(x,n)

```
class Solution {
public:
    double powe(double x, long long n)
    {
        if(n==0)
            return 1;
        if(n==1)
            return x;

    double z=myPow(x,n/2);
        return n%2==0 ? z*z : z*z*x;

    }
    double myPow(double x, long long n) {
       if(n<0)
       { x=1/x;
        n=-n;}
        return powe(x,n);


}};
```

Majority element n/2

```
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int n=nums.size();
        int x=n%2 ? n/2 : n/2;
        return nums[x];
    }
};
```

Unique grid

```
class Solution {
public:
    int uniquePaths(int m, int n) {
        vector<vector<int>>dp(m,vector<int>(n));
     for(int i=0;i<m;i++)
     {
         for(int j=0;j<n;j++)
         {
             if(i==0 || j==0)
                 dp[i][j]=1;
         }
     }
        for(int i=1;i<m;i++)
     {
         for(int j=1;j<n;j++)
         {
           dp[i][j]=dp[i-1][j]+dp[i][j-1];
         }
     }
       return dp[m-1][n-1];
    }
};
```

Reverse pair

```
class Solution {
public:
    int reversePairs(vector<int>& nums) {
        int n=nums.size();
        vector<int> dp(n);
        dp[0]=0;
        for(int i=1;i<n;i++)
        {int count=0;
            for(int j=0;j<i;j++)
            {  long long y=nums[j];
                long long x=nums[i];
             x=x*2;
                if(y> x)
                count++;

            }
         dp[i]=count;
        }
        return accumulate(dp.begin(), dp.end(),0);
    }
};
```

Two sum

C++ Autocomplete

```cpp
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> nn=nums;
        sort(nums.begin(), nums.end());
        int start, end;
        start=0;
        end=nums.size()-1;
        while(start<end)
        {
            if(nums[start]+nums[end]>target)
                 end--;
            else if(nums[start]+nums[end]<target)
                start++;
            else
            break;
        }
        vector<int> ans;
        for(int i=0;i<nn.size();i++)
        { if(nn[i]==nums[start] || nn[i]==nums[end])
            ans.push_back(i);
         if(ans.size()==2)
             break;

        }
        return ans;
    }
};
```

Four sum

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {

       vector<vector<int>> ans;
        sort(nums.begin(), nums.end());

        int n=nums.size();
        for(int i=0;i<n;i++)
        {  int target1=target-nums[i];
            for(int j=i+1;j<n;j++)
            {int target2=target1-nums[j];
             int  low=j+1;
             int high=n-1;
             while(low<high)
             {int ts=nums[low]+nums[high];
                 if(ts>target2)
                     high--;
                 else if(ts<target2)
                     low++;
                 else
                 {vector<int> res(4, 0);
                        res[0] = nums[i];
                        res[1] = nums[j];
                        res[2] = nums[low];
                        res[3] = nums[high];
                        ans.push_back(res);

                 while(low<high && low==res[2])low++;
                 while(low<high && high==res[3])high--;} }
            while(j+1<n && nums[j+1]==nums[j])j++; }
          while(i+1<n && nums[i+1]==nums[i])i++; }
        return ans; }
};
```

Longest consecutive sequence

C++ Autocomplete

```cpp
class Solution {
public:
    int longestConsecutive(vector<int>& nums) {
        if(nums.size()==0)
            return 0;
        unordered_set<int> mpp;
        for(auto it: nums)
        {
            mpp.insert(it);

        }
        int maxa=1;
        for(auto it: nums)
        {
            if(!mpp.count(it-1))
            {
                int curr=it;
                int count=1;
                while(mpp.count(curr+1))
                { count++;
                 curr=curr+1;

                }
                maxa=max(maxa, count);
            }
        }
        return maxa;
    }
};
```

Longest substring without repeating character

C++ Autocomplete

```cpp
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
         unordered_map<char,int> index;
        int start=0,res=0;
        for(int i=0;i<s.length();i++){

            if (index.find(s[i]) != index.end() && index[s[i]] >= start)
                 start = index[s[i]] + 1;

            index[s[i]] = i;
            res=max(res,i-start+1);
        }

        return res;
    }
};
```