Competitive programming

## Max sum algo

Name-kadane’s algorithm

Code in python

From sys import maxint

def maxSubArraySum(a, size):

x = -maxint - 1

y = 0

for i in range(0, size):

y = y + a[i]

if (x < y):

x = y

if y < 0:

y = 0

if x<0:

return 0

else:

return x

ord() function->

argument- char

It returns ASCII value of given argument.

For example ord(‘a’) will be 97.

My solution for maximum sum of subarray-

def maxSubArraySum(L, size):

            s1=0

            for p in range(size):

                s2=L[p]

                if s1<s2:

                    s1=s2

                if p!=size-1:

                    for q in range(p+1,size,1):

                        s2+=L[q]

                        if s1<s2:

                            s1=s2

            return s1

25-4-2023

Def \_\_str\_\_(self):

Return f”{self.name}({self.age})”

Str is a function of python class and it will call after using object of a class.

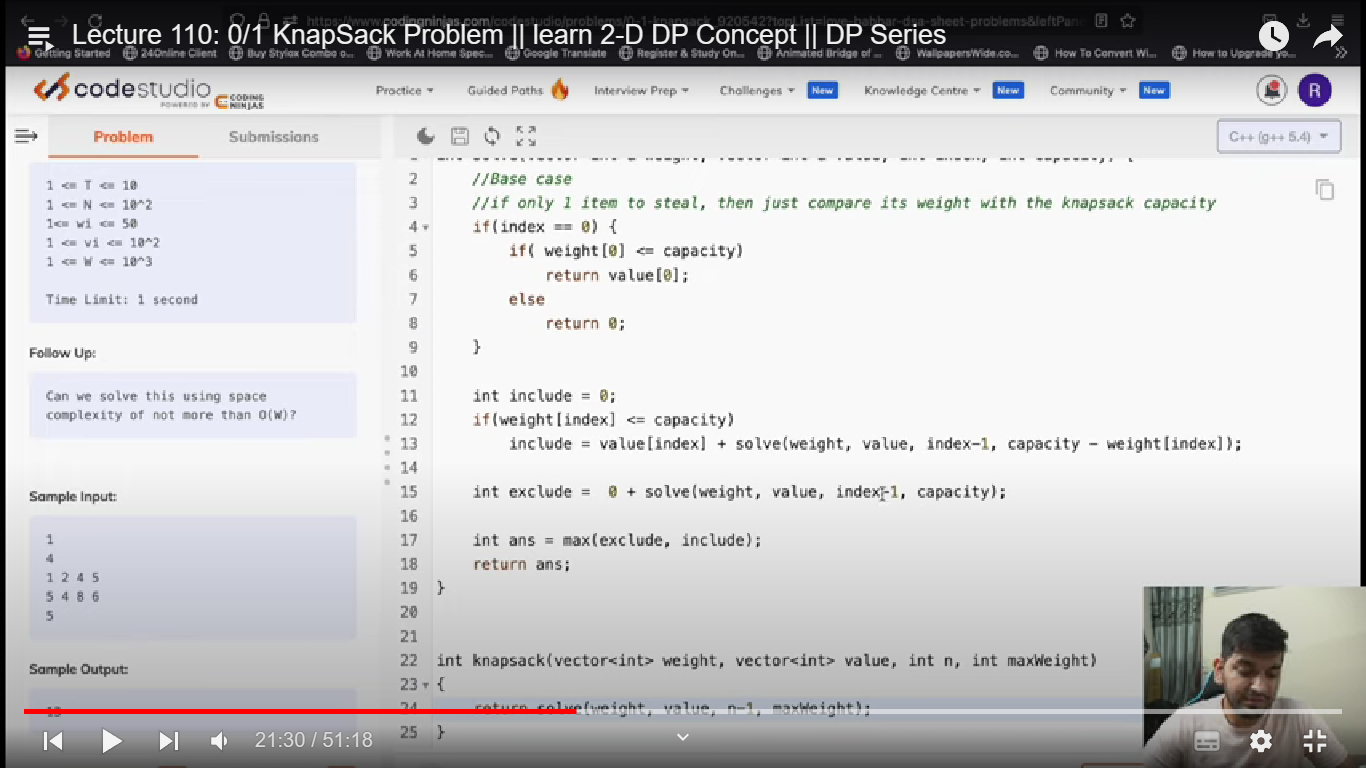
* Del keyword :

It can use for deleting a object of a class in python and object property also.

* For sorting list of list on the basis of their particular position element
* static bool cmp(vector<int> &a,vector<int> &b ){
* return a[1]>b[1];
* }

Sort(nums.begin(),nums.end(),cmp);

* Stoi(string) function converts string to int like “7” will become 7.



Level order traversal

/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     TreeNode \*left;

 \*     TreeNode \*right;

 \*     TreeNode() : val(0), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

 \* };

 \*/

class Solution {

public:

    void level(TreeNode\* root,vector<vector<int>> &ans){

        queue<TreeNode\*>q;

        vector<int>sub;

        q.push(root);

        q.push(NULL);

        while(!q.empty()){

            TreeNode\* temp=q.front();

            q.pop();

            if(temp==NULL){

                ans.push\_back(sub);

                if(!q.empty()){

                    q.push(NULL);

                    sub.clear();

                }

            }

            else{

                sub.push\_back(temp->val);

                if(temp->left)

                    q.push(temp->left);

                if(temp->right)

                    q.push(temp->right);

            }

        }

    }

    vector<vector<int>> levelOrder(TreeNode\* root) {

        vector<vector<int>>ans;

        if(root==NULL)

            return ans;

        level(root,ans);

        return ans;

    }

};

Zigzag traversal (level-order) for binary tree

/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     TreeNode \*left;

 \*     TreeNode \*right;

 \*     TreeNode() : val(0), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

 \* };

 \*/

class Solution {

public:

        void level(TreeNode\* root,vector<vector<int>> &ans){

        queue<TreeNode\*>q1;

        queue<TreeNode\*>q2;

        vector<int>sub;

        q1.push(root);

        q1.push(NULL);

        q2.push(root);

        q2.push(NULL);

        bool flag=false;

        while(!q1.empty() ){

            TreeNode\* temp1=q1.front(),\*temp2=q2.front();

            q1.pop();

            q2.pop();

            if(temp1==NULL){

                if(flag)

                    flag=false;

                else

                    flag=true;

                ans.push\_back(sub);

                if(!q1.empty()){

                    q1.push(NULL);

                    q2.push(NULL);

                    sub.clear();

                }

            }

            else{

                if(flag)

                    sub.push\_back(temp2->val);

                else

                    sub.push\_back(temp1->val);

                if(temp1->left)

                    q1.push(temp1->left);

                if(temp1->right)

                    q1.push(temp1->right);

                if(temp2->right)

                    q2.push(temp2->right);

                if(temp2->left)

                    q2.push(temp2->left);

            }

        }

    }

    vector<vector<int>> zigzagLevelOrder(TreeNode\* root) {

        vector<vector<int>>ans;

        if(root==NULL)

            return ans;

        level(root,ans);

        return ans;

    }

};