# LEET ds

binary search, sliding window, greedy, dp(subsequence, multiple state dp, partitioning, knapsacks etc), linked list, stack, monotonic stack, queue, graphs, tree, segment tree, DSU, min spanning tree, network algorithm, string matching algo's, tries etc.... after doing this much you go solving standard questions...

# LEET Patterns

## 1 lock and combination /word ladder/finding all jumping no less than x

### How to find its pattern ( BFS )

1. Ask about number or word
2. Each digit or character can be changed
3. You have to reach a target word /number in minimum moves
4. Lists of word or number allowed

# Cp patterns-

Finding substring such that they have something .

Logic🡪

00010101

Think of total subarray or substring be like

🡺n\*n+1/2 and which have equal 0 and 1 be E

Than 1\*(T-E){which give all with 1 mode} and 2\*E so =>t+E

#include <bits/stdc++.h>

using namespace std;

#define int long long

void solve(){

int n;

cin>>n;

string s;

cin>>s;

map<int, int> mp;

int sum = 0;

int cnt = 0;

mp[0] = 1;

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

sum += (s[i]=='0') ? 1 : -1;

if(mp.find(sum) != mp.end()){

cnt += mp[sum];

}

mp[sum]++;

}

int ans = n\*(n+1)/2;

cout<<ans+cnt<<"\n";

}

int32\_t main() {

// your code goes here

int t;

cin>>t;

while(t--){

solve();

}

}

/\* learnings are-

1. use of bool for printing

2. for vector

{1,2,4,5}=>{1,2,3,4,5}

v.insert(v.begin()+2,3)  
V.count(v.begin()+I,v.end(),v[i])

v.erase(v.begin()+2)

for set

setname.contain(nnumber)

set.erase(num)

# INTITUTION

🡪IF A🡪B SRC TO DESTINATION IT COULD BE DONE BY GOING FROM DEST TO SRC

# Prefix sum+ hashtable

<https://leetcode.com/problems/continuous-subarray-sum/solutions/5276981/prefix-sum-hashmap-patterns-7-problems/>

## very good technique to find different ways around subarray problem

unordered\_map<int,int>mp;

mp[0]=-1;

int prefmod=0;

for(int i=0;i<nums.size();i++){

prefmod=(prefmod+nums[i])%k;

if(mp.find(prefmod)!=mp.end()){

if(i-mp[prefmod]>1)return true;

}

else{

mp[prefmod]=i;

}

}

return false;

}

### For prefix sum to find sum between a range i to j we use

### nums[j]-nums[i-1]

### If(i==0)nums[j]

# MATRIX CHAIN MULTIPLICATION

class Solution{

public:

int dp[101][101];

int solve(int i,int j,int arr[]){

int ans=INT\_MAX;

if(i>=j)return 0;

if(dp[i][j]!=-1)return dp[i][j];

for(int k=i;k<j;k++){

ans=min(ans,solve(i,k,arr)+solve(k+1,j,arr)+arr[i-1]\*arr[k]\*arr[j]);

}

return dp[i][j]=ans;

}

int matrixMultiplication(int N, int arr[])

{

memset(dp,-1,sizeof(dp));

int i,j,ans;

solve(1,N-1,arr);

// code here

}

# Maths

## SIEVE

SOME VARIATION WITH HIGHEST AND LOWEST PRIME

void SieveOfEratosthenes(long long s, long long e) {

    vector<bool> prime(e + 1, true);

vector<int>hp(e-s+1,0);

vector<int>lp(e-s+1,0);

    prime[0] = prime[1] = false;

    for (long long p = 2; p \* p <= e; p++) {

        if (prime[p] == true) {

lp[p]=hp[p]=p;

            for (long long i = p \* p; i <= e; i += p)

                prime[i] = false;

🡪 hp[i]=p;

🡪if(lp[i]==0)lp[i]=p;

        }

    }

    // Print all prime numbers

    for (long long p = max(2ll, s); p <= e; p++) {

        if (prime[p]) {

            cout << p << " ";

        }

    }

}

## MOD

# For negative x🡪X% P=====(X%P+P)%P

# In [mathematics](https://en.wikipedia.org/wiki/Mathematics), Legendre's three-square theorem states that a [natural number](https://en.wikipedia.org/wiki/Natural_number) can be represented as the sum of three squares of integers�=�2+�2+�2if and only if *n* is not of the form �=4�(8�+7)N=4^a(8b+7) for nonnegative integers *a* and *b*.

\*/

\*\*For finding the maxfreq of any element in array 🡪

 cin>>n;

      vector<int>cnt(n+1);

      for (int i = 0; i <n; i++)

      {

        int x;

        cin>>x;

        ++cnt[x];

      }

      int mx=\*max\_element(cnt.begin(),cnt.end());

# 2 FOR STRINGS

1 ahhahhah 2 check substring

If we have to find the required string from jumbled string what we do is looping ->

Ex-1->SKDJHhENSALLFOG 2>FIND HELLO

hello=”hello”

cin >> word;

    int i,cnt=0,j=0;

    fo(i,word.length()){

        if(word[i]==hello[j]){

            cnt++,j++;

        }

    }

    if(cnt==5){

        cout<<"YES";

    }

    else{

        cout<<"NO";

    }

2.

* To lower case->transform(s.begin(),s.end(),s.begin(), ::tolower);
* String mai kuch add ya remove krne ke lie new string bhi use kr skte hai
* Remove krne ke lie continue kr skte hai

## CHECKING STRING PALINDROME OR NOT CAN REMOVE “K” ELEMENTS

   int cnt=0;

      for (int i = 0; i <n; i++)

      {

        mp[s[i]-'a']++;

      }

      for (auto c:mp)

      {

        if(c.second%2!=0)cnt++;

      }

      if(cnt>k+1)cout<<"NO"<<endl;

      else cout<<"YES"<<endl;

    }

* If you wants to find the count of character in a string like “8989889” want to find cnt of 8 or 9 you can do this🡪
* Vector of appropriate size vector<int>cnt(10)
* For(auto ch:s) ++cnt[ch-‘0’];
* string s;
* cin>>s;
* vector<int>cnt(10);
* for(auto ch:s)++cnt[ch-'0'];
* int mx=\*max\_element(cnt.begin(),cnt.end());

## IF YOU WANT TO FIND GOOD OR BAD STRINGS

BINARY WITH 0,1

CHECK CONDITIONS AND BOOLEAN

\*\* TO CHECK IF A string contain unique digit approach change no. to string using by to\_string(n) then put element of string to a set<int>s then compare length

\*\*agar koi number bhut bda ho to phele use string mai convert krke

String func(string s ) string he return krdo..(long long ka issue he nhi);

\*\*binary removal-🡪if 1100 then not possible to form a sorted string from 010111010100 in which can be formed by removing.

std::cout << "Before memset: " << str << std::endl; 🡪hello,world

// Set the first 5 characters of str to 'a'

memset(str, 'a', 5);

std::cout << "After memset: " << str << std::endl;s

## memset(str, 'a', 5);🡪aaaaa,world

## CREATING STRING –

STRING(SIZE,char);

# Stringstream

To convert this 1.09.34.52-----🡪{“1”,”09”,”34”,”52”}

We use

stringstream ss(string to be converted)

Vector<string>ans;

While getline(ss,jisme daalna hai(s), ”.”(delimiter)){

String s=””;

ans.push\_back(s);

}

### Creating substring

String original;

String temp=original.substr(starting index, length);

#### If we don’t know the starting index=

String temp=original.substr(original.find(“name”),len);

## Check substring

bool isSubstring(const string& s, const string& b) {

    return s.find(b) != string::npos;

}

# KDANE ALGO

1. int best = 0, sum = 0; for (int k = 0; k < n; k++) {

sum = max(array[k],sum+array[k]);

best = max(best,sum); }

cout << best << "\n";

### variation of kdane in which sum cannot be less than 0

ll best = 0, sum = 0;

 for (int j = 0; j < n; j++) {

 sum +=v[j];

 best = max(best,sum);

 sum=max(sum,0ll);

 }

## 🡪MAXI VALA TRIKA

# Comparator is used for swapping of pairs-

Bool comp(vector<int>&ele1,vector<int>&ele2){

return ele1[1]<ele2[1];

}

# Prefix and suffix

vector<int>prefix(n,0);

        vector<int>suffix(n,0);

        vector<int>ans(n,0);

        prefix[0]=nums[0];

        for(int i=1;i<n;i++){

          prefix[i]=prefix[i-1]\*nums[i];

        }

        suffix[n-1]=nums[n-1];

        for(int i=n-2;i>=0;i--){

          suffix[i]=suffix[i+1]\*nums[i];

        }

        ans[0]=suffix[1];

        ans[n-1]=prefix[n-2];

USE OF PREFIX AND SUFFIX SUM TO FIND UNIQUE ELEMENT

vi prefix(n,0),suff(n,0);

 fo(i,n){

    se.insert(s[i]);

    prefix[i]=se.size();

 }

 se.clear();

 for(int i=n-1;i>=0;i--){

    se.insert(s[i]);

    suff[i]=se.size();

 }

se.clear();

one of the imp concept---

## prefix sum+hashing to find a subarray with 0 sum;

bool subarray(vector<int>&v,int n){

  unordered\_set<int>us;

  int sum=0;

  for (int i = 0; i <n;i++)

  {

   //v[i]\*=i%2==0?-1:1;

    sum+=v[i];

   if(sum==0||us.find(sum)!=us.end()){

    return true;

   }

   us.insert(sum);

 }

 return false;

  }

## Prefix sum

prefixs[n+1]{};

prefixs[0]=a[0];

for (int  i = 1; i <=n; i++)

{

  prefixs[i]+=prefixs[i-1]+a[i];

}

//return prefixs[r]-prefixs[l-1];

# QUESTIONS->

### Problem Statement:

Given an array A of size N, initially all 0's. There are Q queries of the form: +LRx -> add x to all indices in the range [L, R]. After all queries, you need to find the final array.

# Structure binding

For(auto[a,b]:mp(anydatastructure name){  
cout<<a<<b<<endl;

}

1. 1 ques reference for using 2 array as one-
2. **vector**<pair<**int**,**int**>> sorted(n);
3. **for**(**int** i = 0; i < n; i++){
4. cin >> sorted[i].second;
5. }
6. **for**(**int** i = 0; i < n; i++){
7. cin >> sorted[i].first;
8. sorted[i].first = abs(sorted[i].first);
9. }

## maxoccurence

// max occurence

map<char,int>mp;

for(auto c:s){

  maxi=max(maxi,++mp[c]);

}

for(auto c:s){

  if(mp[c]==maxi)

    cout<<c<<" ";

  cout<<endl;

}

# sliding window

Subarray/subtring+largest or max+k(size of window)

sliding window

int findSubstring(string s){

vector<int> map(128,0);

int counter; // check whether the substring is valid

int i=0,j=0,cnt=0; //two pointers, one point to tail and one head

int n; //the length of substring

for(int i=0;i<n;i++) { /\* initialize the hash map here \*/

}

while(j<n){

if(map[s[end++]]-- ?){ /\* modify counter here \*/ }

while(/\* counter condition \*/){

/\* update d here if finding minimum\*/

//increase begin to make it invalid/valid again

}

/\* update cnt here if finding maximum\*/

}

return cnt;

}

# 6 For map or pairs

Give pairs of map

1. Mp.find()🡪iterator
2. Mp.contains()

# 7 set

Set can use pairs but uset can not

Subset(not ordr)=powerset!=subsequence(not cont but order) substring(continuos) \*\*subarray (ordered)or substring=continuous

## Iterating a set and printing its element

void display(set<string>&s)

{

  set<string>::iterator itr;

  // Displaying set elements

  for (itr = s.begin();

       itr != s.end(); itr++)

  {

    cout << \*itr << " ";

  }

}

# 8 iterators

Iterator.begin()+x=interator to index

## Index of max element

### Max\_element(v.begin(),v.end())-v.begin() gives the index of max element

# 9 comaparator

bool comparatar(pair<int,int>a ,pair<int,int>b){

 //asending swap

if(a.first<b.first)return true;

    else return false;

}

sort(v.begin(), v.end(), [](const vector<int>& a, const vector<int>& b) {

if (a[0] == b[0]) return a[1] < b[1];

return a[0] < b[0];

});

Jp apko chaiye vo return mai daaldo

bool comparatar(datatype a ,datatype b){

return a.second<b.second;

}

## Inbuilt sort

Inbuilt sort function bolta hai ki agar false hoga to mai swap kr dunga to cmp mai false vale condition ke hisabh se hoga game.

# BS special case:

## [LC Study Guide: Binary Search on Answer](https://leetcode.com/discuss/study-guide/3444552/binary-search-on-answer-template-generic-template) Must do to master the binary search on answer:

* <https://leetcode.com/problems/minimum-time-to-repair-cars/> <-- this one is gold!!!
* <https://leetcode.com/problems/minimum-speed-to-arrive-on-time/>
* <https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/>
* <https://leetcode.com/problems/koko-eating-bananas/>
* <https://leetcode.com/problems/maximum-candies-allocated-to-k-children/>
* <https://leetcode.com/problems/magnetic-force-between-two-balls/>
* <https://leetcode.com/problems/sell-diminishing-valued-colored-balls/>
* <https://leetcode.com/problems/minimum-limit-of-balls-in-a-bag/>
* <https://leetcode.com/problems/divide-chocolate/> <-- this one is gold!!! (hard)

# 10 binarysearch STL

BINARY\_SEARCH(V,V+7,VALUE )🡪TRUE OR FALSE

# UPPER BOUND AND LOWER BOUND

LOWER BOUND->FIRST VAL>=T

UPPER BOUND ->FIRST VAL >T( location of agla element agar t present ho to bhi)

LB-1<T

UB-1<=T

## syntax

auto ptr=lower\_bound(a,a+n,x)

WATCH EDGE CASES SUCH AS WHEN (IT==A)BEGINNING INDEX THEN WE CAN NOT IT--;

## Bounds in set and map different

set.upperbound()

\*\*Floor and ceil should lean toward the changing side. var!=mean

### = on the side which we have to get rid of.(last or first index)

bs wiill move l🡪mid when pred true(left udao)

bs will move r->mid when pred false(right udao)

bt find tf junction ans is last true or first false(l,r)

## ALTERNATE BINARY SEARCH

l=minspace-1,r=maxspace+1

   long firstBadVersion(long n) {

     long l = 0;

    long r = n+1;

    while (r-l>1) {

        long m = l + (r - l) / 2;

        if (isBadVersion(m)==0) {

            l = m;

        } else {

            r = m;

        }

    }

        return r;

    }

in real valyue preferred log2max/e

BS special case: [LC Study Guide: Binary Search on Answer](https://leetcode.com/discuss/study-guide/3444552/binary-search-on-answer-template-generic-template)  
Must do to master the binary search on answer:

* //done<https://leetcode.com/problems/minimum-time-to-repair-cars/> <-- this one is gold!!!
* <https://leetcode.com/problems/minimum-speed-to-arrive-on-time/>
* <https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/>
* <https://leetcode.com/problems/koko-eating-bananas/>
* <https://leetcode.com/problems/maximum-candies-allocated-to-k-children/>
* <https://leetcode.com/problems/magnetic-force-between-two-balls/>
* <https://leetcode.com/problems/sell-diminishing-valued-colored-balls/>
* <https://leetcode.com/problems/minimum-limit-of-balls-in-a-bag/>
* <https://leetcode.com/problems/divide-chocolate/> <-- this one is gold!!! (hard)

recurssion

int count = 0; // Global parameters that are useful for results.

int minEnd = INT\_MAX; // Key parameters characterizing the "active set" for overlapping intervals, e.g. the minimum ending point among all overlapping intervals.

sort(points.begin(), points.end()); // Sorting the intervals/pairs in ascending order of its starting point

for each interval {

      if(interval.start > minEnd) { // If the

     // changing some states, record some information, and start a new active set.

    count++;

    minEnd = p.second;

      }

     else {

    // renew key parameters of the active set

    minEnd = min(minEnd, p.second);

      }

 }

return the result recorded in or calculated from the global information;

int findMinArrowShots(vector<pair<int, int>>& points) {

int count = 0, minEnd = INT\_MAX;

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

for(auto& p: points) {

if(p.first > minEnd) {count++; minEnd = p.second;}

else minEnd = min(minEnd, p.second);

}

return count + !points.empty();

}

# LINKEDLIST

## REVERSE A LINKED LIST

ListNode\* revl(ListNode\* HEAD) {

    ListNode\* CURR = HEAD;

    ListNode\* PREV = NULL;

    while(CURR != NULL) {

        ListNode\* NEXTNODE = CURR->next;

        CURR->next = PREV;

        PREV = CURR;

        CURR = NEXTNODE;

    }

    return PREV;

}

## Merging 2 or more linked list

Node\* merge(Node\*r1,Node\*r2){

    if(!r1){

        return r2;

    }

    if(!r2){

        return r1;

    }

    Node\* r3=r1;

    if(r1->data<=r2->data){

        r3=r1;

        r3->bottom=merge(r1->bottom,r2);

    }

    else{

         r3=r2;

        r3->bottom=merge(r1,r2->bottom);

    }

    return r3;

}

# Bit manipulation

vector<vector<int>> findAllSubsets(vector<int>& nums) {

    vector<vector<int>> subsets;

    int n = nums.size();

    int totalSubsets = 1 << n; // 2^n subsets

    for (int subsetMask = 0; subsetMask < totalSubsets; ++subsetMask) {

        vector<int> subset;

        for (int i = 0; i < n; ++i) {

            // Check if the i-th element is included in the subset

            if (subsetMask & (1 << i)) { //1 bit

                subset.push\_back(nums[i]);

            }

        }

        subsets.push\_back(subset);

    }

    return subsets;

}

Or

   vector<vector<int>> subsets(vector<int>& nums) {

       vector<vector<int>>ans;

       int n=nums.size();

        for(int i=0;i<(1<<n);i++){

            vector<int>subset;

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

                 if(i&(1<<bitset))subset.push\_back(nums[bitset]);

            }

            ans.push\_back(subset);

        }

        return ans;

    }

# Subarray with given xor-

int Solution::solve(vector<int> &A, int B) {

      unordered\_map<int,int>mp;

      int cnt=0,xorr=0;

      for(auto it:A){

          xorr^=it;

          if(xorr==B)cnt++;

          if(mp.find(xorr^B)!=mp.end()){

              cnt+=mp[xorr^B];

          }

           mp[xorr]+=1;

       }

      return cnt;

}

# BACKTRACKING

Steps can be think of as subset because result is a 2d vector and steps is 1d vector of subset

## TEMPLATE

vector<vector<int>> subsets(vector<int>& nums) {

vector<int> step;

vector<vector<int>> result;

helper(nums, 0, step, result);

return result;

}

void helper(const vector<int>& nums, int start, vector<int>& step, vector<vector<int>>& result) {

///CONDITION TO PUSH IN RESULT

result.push\_back(step);

for (int i = start; i < nums.size(); ++i) {

//condition to push in step

step.push\_back(nums[i]);

helper(nums, i + 1, step, result);

step.pop\_back();

}

}

## GENERATE ALL STIRNG PERMUTATION

class Solution {

private:

void recurPermute(int index, vector<int> &nums, vector<vector<int>> &ans) {

if(index == nums.size()) {

ans.push\_back(nums);

return;

}

for(int i = index;i<nums.size();i++) {

swap(nums[index], nums[i]);

recurPermute(index+1, nums, ans);

swap(nums[index], nums[i]);

}

}

public:

vector<vector<int>> permute(vector<int>& nums) {

vector<vector<int>> ans;

recurPermute(0, nums, ans);

return ans;

}

};

# DP

## LCS-longest common subsequence

  int dp[1001][1001];

    int LCS(string text1, string text2) {

        int n=text1.size();

        int m=text2.size();

         for(int i=0;i<n+1;i++){

        for(int j=0;j<m+1;j++){

             if(i==0||j==0)dp[i][j]=0;

            }

        }

         for(int i=1;i<n+1;i++){

        for(int j=1;j<m+1;j++){

             if(text1[i-1]==text2[j-1]){

                dp[i][j]=1+dp[i-1][j-1];

             }

             else {

               dp[i][j]=max(dp[i-1][j],dp[i][j-1]);

             }

            }

        }

        return dp[n][m];

    }

# LONGEST INCREASING SUBSEQUENCE

//Approach-2 (Bottom Up DP) O(n\*n)

class Solution {

public:

int lengthOfLIS(vector<int>& nums) {

int n = nums.size();

vector<int> t(n, 1);

int maxL = 1;

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

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

if(nums[j] < nums[i]) {

t[i] = max(t[i], t[j]+1);

maxL = max(maxL, t[i]);

}

}

}

return maxL;

}

};

## Using prev

class Solution {

public:

int n;

int t[2501][2501];

int lis(vector<int>& nums, int prev\_idx, int curr\_idx) {

if(curr\_idx == n)

return 0;

if(prev\_idx != -1 && t[prev\_idx][curr\_idx] != -1)

return t[prev\_idx][curr\_idx];

int taken = 0;

if(prev\_idx == -1 || nums[curr\_idx] > nums[prev\_idx])

taken = 1 + lis(nums, curr\_idx, curr\_idx+1);

int not\_taken = lis(nums, prev\_idx, curr\_idx+1);

if(prev\_idx != -1)

t[prev\_idx][curr\_idx] = max(taken, not\_taken);

return max(taken, not\_taken);

}

int lengthOfLIS(vector<int>& nums) {

memset(t, -1, sizeof(t));

n = nums.size();

return lis(nums, -1, 0);

}

};

# Graphs

## Adjacency list-

unordered\_map<int, vector<int>> adj;

// Construct adjacency list

for (int u = 0; u < V; u++) {

for (int v : mp[u]) {

adj[u].push\_back(v);

}

}

# Bfs

void bfs(vector<bool>&visited,int V, unordered\_map<int, vector<int>> &adj,vector<int>&result){

queue<int>q;

q.push(V);

visited[V]=true;

result.push\_back(V);

while(!q.empty()){

int u=q.front();

q.pop();

for(auto v:adj[u]){

if(!visited[v]){

q.push(v);

visited[v]=true;

result.push\_back(v);

}

}

}

}

# Dfs

void dfs(unordered\_map<int, vector<int>>& adj, int u, vector<bool>& visited, vector<int>& result) {

if (visited[u]) return;

visited[u] = true;

result.push\_back(u);

for (int v : adj[u]) {

if (!visited[v]) {

dfs(adj, v, visited, result);

}

}

}

# Toposort(kahn algo)

        vector<int>indeg(n,0);

        vector<int>adj[n];

        vector<int>topo;

        for(auto it:prerequisites){

                 adj[it[0]].push\_back(it[1]);

                 indeg[it[1]]++;

        }

        queue<int>q;

        for(int i=0;i<n;i++){

            if(indeg[i]==0)q.push(i);

        }

        while(!q.empty()){

            auto node=q.front();

            q.pop();

            topo.push\_back(node);

            for(auto it:adj[node]){

                indeg[it]--;

                if(indeg[it]==0)q.push(it);

            }

        }

        return topo.size()==n; //bool

    }

RETURNING VECTOR

vector<int> topoSort(int V, vector<int> adj[]){

int indegree[V] = {0};

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

for(auto it:adj[i]){

indegree[it]++;

}

}

queue<int>q;

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

if(indegree[i]==0){

q.push(i);

}

}

vector<int>topo;

while(!q.empty()){

int node = q.front();

q.pop();

topo.push\_back(node);

// now node is in topo sort so remove it from indegree

for(auto it:adj[node]){

indegree[it]--;

if(indegree[it]==0) q.push(it);

}

}

return topo;

# tree

d=log2(n+1)-1

n=2^(depth+1)]-1

# TRIE

 struct TrieNode{

     TrieNode\* child[26];

     bool isend;

    };

    TrieNode\* node;

Public:

    Trie() {

      node=new TrieNode();

    }

    void insert(string word) {

        TrieNode\* curr=node;

        for(char c:word){

            if(curr->child[c-'a']==NULL)curr->child[c-'a']=new TrieNode();

            curr=curr->child[c-'a'];

        }

        curr->isend=true;

    }

    bool search(string word) {

        TrieNode\* curr=node;

        for(char c :word){

            if(curr->child[c-'a']==NULL)return false;

            curr=curr->child[c-'a'];

        }

        if(curr->isend==true)return true;

        else return false;

    }

  bool startsWith(string prefix) {

        TrieNode\* curr=node;

        for(char c:prefix){

            if(curr->child[c-'a']==NULL)return false;

            curr=curr->child[c-'a'];

        }

        return true;

    }

};