

## The Celebrity Problem

Another classical problem.

*In a party of  $N$  people, only one person is known to everyone. Such a person **may be present** in the party, if yes, (s)he doesn't know anyone in the party. We can only ask questions like “**does A know B?** “. Find the stranger (celebrity) in minimum number of questions.*

We can describe the problem input as an array of numbers/characters representing persons in the party. We also have a hypothetical function *HaveAcquaintance*( $A, B$ ) which returns *true* if  $A$  knows  $B$ , *false* otherwise. How can we solve the problem, try yourself first.

We measure the complexity in terms of calls made to *HaveAcquaintance*( $\cdot$ ).

### Graph:

We can model the solution using graphs. Initialize indegree and outdegree of every vertex as 0. If  $A$  knows  $B$ , draw a directed edge from  $A$  to  $B$ , increase indegree of  $B$  and outdegree of  $A$  by 1. Construct all possible edges of the graph for every possible pair  $[i, j]$ . We have  $N_{C_2}$  pairs. If celebrity is present in the party, we will have one sink node in the graph with outdegree of zero, and indegree of  $N-1$ . We can find the sink node in  $(N)$  time, but the overall complexity is  $O(N^2)$  as we need to construct the graph first.

### Recursion:

We can decompose the problem into combination of smaller instances. Say, if we know celebrity of  $N-1$  persons, can we extend the solution to  $N$ ? We have two possibilities, Celebrity( $N-1$ ) may know  $N$ , or  $N$  already knew Celebrity( $N-1$ ). In the former case,  $N$  will be celebrity if  $N$  doesn't know anyone else. In the later case we need to check that Celebrity( $N-1$ ) doesn't know  $N$ .

Google™ Custom Search



GeeksforGeeks



53,521 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

Solve the problem of smaller instance during divide step. On the way back, we may find a celebrity from the smaller instance. During combine stage, check whether the returned celebrity is known to everyone and he doesn't know anyone. The recurrence of the recursive decomposition is,

$$T(N) = T(N-1) + O(N)$$

$T(N) = O(N^2)$ . You may try Writing pseudo code to check your recursion skills.

### Using Stack:

The graph construction takes  $O(N^2)$  time, it is similar to brute force search. In case of recursion, we reduce the problem instance by not more than one, and also combine step may examine M-1 persons (M – instance size).

We have following observation based on elimination technique (Refer *Polya's How to Solve It* book).

- If A knows B, then A can't be celebrity. Discard A, and *B may be celebrity*.
- If A doesn't know B, then B can't be celebrity. Discard B, and *A may be celebrity*.
- Repeat above two steps till we left with only one person.
- Ensure the remained person is celebrity. (Why do we need this step?)

We can use stack to verify celebrity.

1. Push all the celebrities into a stack.
2. Pop off top two persons from the stack, discard one person based on return status of *HaveAcquaintance(A, B)*.
3. Push the remained person onto stack.
4. Repeat step 2 and 3 until only one person remains in the stack.
5. Check the remained person in stack doesn't have acquaintance with anyone else.

We will discard N elements utmost (Why?). If the celebrity is present in the party, we will call *HaveAcquaintance()*  $3(N-1)$  times. Here is code using stack.

```
#include <iostream>
#include <list>
using namespace std;
```



## Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

```
// Max # of persons in the party
#define N 8

// Celebrities identified with numbers from 0 through size-1
int size = 4;
// Person with 2 is celebrity
bool MATRIX[N][N] = {{0, 0, 1, 0}, {0, 0, 1, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};

bool HaveAcquaintance(int a, int b) { return MATRIX[a][b]; }
```

```
int CelebrityUsingStack(int size)
{
    // Handle trivial case of size = 2

    list<int> stack; // Careful about naming
    int i;
    int C; // Celebrity

    i = 0;
    while( i < size )
    {
        stack.push_back(i);
        i = i + 1;
    }

    int A = stack.back();
    stack.pop_back();

    int B = stack.back();
    stack.pop_back();

    while( stack.size() != 1 )
    {
        if( HaveAcquaintance(A, B) )
        {
            A = stack.back();
            stack.pop_back();
        }
        else
        {
            B = stack.back();
            stack.pop_back();
        }
    }

    // Potential candidate?
```

# Deploy Early. Deploy Often.

DevOps from  
Rackspace:

## Automation

[FIND OUT HOW ►](#)



## Recent Comments

**Aman** Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 18 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 21 minutes ago

**Sanjay Agarwal** bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...

Root to leaf path sum equal to a given number · 46 minutes ago

**GOPI GOPINATH** @admin Highlight this

sentence "We can easily...


Count trailing zeroes in factorial of a number · 48 minutes ago

**newCoder3006** If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

**newCoder3006** Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

AdChoices 

[► Matrix in Java](#)

[► C++ Code](#)

[► Celebrity Job](#)

```
C = stack.back();
stack.pop_back();

// Last candidate was not examined, it leads one excess comparison
if( HaveAcquaintance(C, B) )
    C = B;

if( HaveAcquaintance(C, A) )
    C = A;

// I know these are redundant,
// we can simply check i against C
i = 0;
while( i < size )
{
    if( C != i )
        stack.push_back(i);
    i = i + 1;
}

while( !stack.empty() )
{
    i = stack.back();
    stack.pop_back();

    // C must not know i
    if( HaveAcquaintance(C, i) )
        return -1;

    // i must know C
    if( !HaveAcquaintance(i, C) )
        return -1;
}

return C;
}

int main()
{
    int id = CelebrityUsingStack(size);
    id == -1 ? cout << "No celebrity" : cout << "Celebrity ID " << id;
    return 0;
}
```

Output

Celebrity ID 2

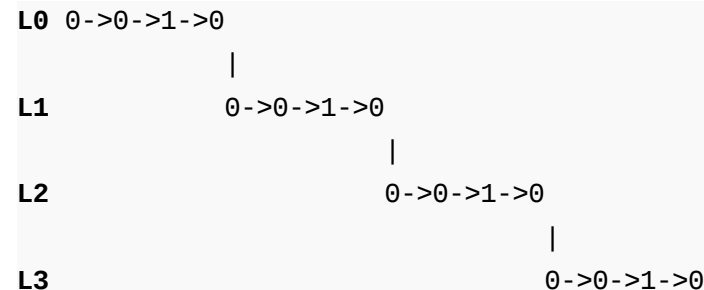
[► Celebrity Gossip](#)[► People Celebrity](#)[► 8 Celebrity](#)[► 8 Celebrity](#)[► At the Celebrity](#)[► Celebrity Stack](#)

Complexity  $O(N)$ . Total comparisons  $3(N-1)$ . Try the above code for successful MATRIX  $\{\{0, 0, 0, 1\}, \{0, 0, 0, 1\}, \{0, 0, 0, 1\}, \{0, 0, 0, 1\}\}$ .

### A Note:

You may think that why do we need a new graph as we already have access to input matrix. Note that the matrix MATRIX used to help the hypothetical function *HaveAcquaintance*(A, B), but never accessed via usual notation MATRIX[i, j]. We have access to the input only through the function *HaveAcquaintance*(A, B). Matrix is just a way to code the solution. We can assume the cost of hypothetical function as  $O(1)$ .

If still not clear, assume that the function *HaveAcquaintance* accessing information stored in a set of linked lists arranged in levels. List node will have *next* and *nextLevel* pointers. Every level will have N nodes i.e. an N element list, *next* points to next node in the current level list and the *nextLevel* pointer in last node of every list will point to head of next level list. For example the linked list representation of above matrix looks like,

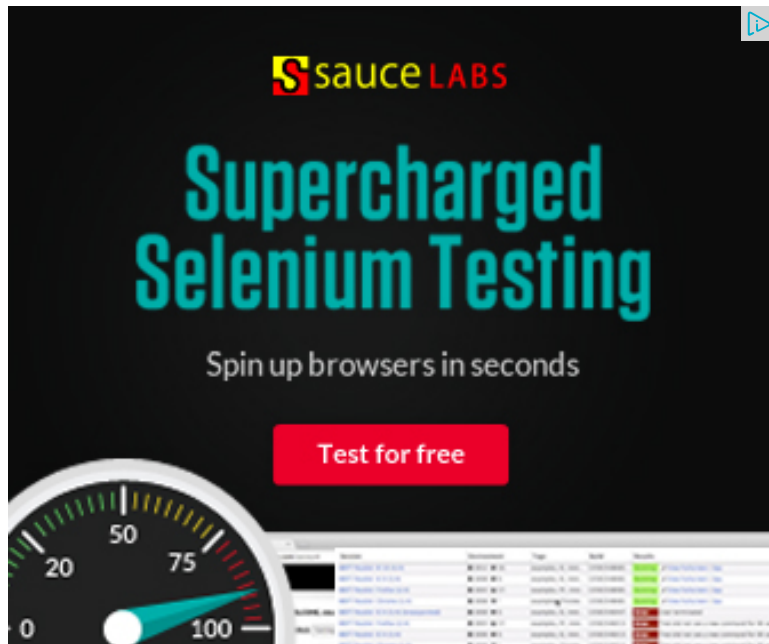


The function *HaveAcquaintance*(i, j) will search in the list for *j-th* node in the *i-th* level. Our goal is to minimize calls to *HaveAcquaintance* function.

### Exercises:

1. Write code to find celebrity. Don't use any data structures like graphs, stack, etc... you have access to *N* and *HaveAcquaintance*(int, int) only.
2. Implement the algorithm using Queues. What is your observation? Compare your solution with [Finding Maximum and Minimum](#) in an array and [Tournament Tree](#). What are minimum number of comparisons do we need (optimal number of calls to *HaveAcquaintance*()?)

— [Venki](#). Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



## Related Tpoics:

- [Remove minimum elements from either side such that  \$2 \times \text{min}\$  becomes more than max](#)
- [Divide and Conquer | Set 6 \(Search in a Row-wise and Column-wise Sorted 2D Array\)](#)
- [Bucket Sort](#)
- [Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1](#)
- [Find the number of zeroes](#)
- [Find if there is a subarray with 0 sum](#)
- [Divide and Conquer | Set 5 \(Strassen's Matrix Multiplication\)](#)
- [Count all possible groups of size 2 or 3 that have sum as multiple of 3](#)



3

Tweet

2



1

**Writing code in comment?** Please use [ideone.com](#) and share the link here.

**31 Comments**

**GeeksforGeeks**

Sort by Newest ▼



Join the discussion...



vamos\_sagar • 9 months ago

```
package com.main;

import java.util.ArrayList;
import java.util.List;
import java.util.Stack;

public class CelebrityIdentifier {

    /**
     * @param args
     */

    Stack<Member> membersStack = new Stack<Member>();
    static Member noCelebrity = null;

    private static class Member{
        private List<Member> acquaintanceList = new ArrayList<
        String name;
```

[see more](#)

^ | v • Reply • Share ›



Ronny • 10 months ago

@GeeksforGeeks

Shouldn't N in the stack method be 4 since we have four persons in the party.  
typo ??



**frugal** · 2 years ago

It takes atmost  $2n$  calls to the **function** HaveAcquaintance.  
in this the first loop checks for the candidate element for the celeb  
  
first loop ensures that all element after candidate+1 know candidate ;  
  
please tell me if anything is wrong in this approach.

```
int celebrity(int a[],int size)
{
    candidate = 0;
    for(int i = 0; i<size;i++){
        if(!HaveAcquaintance(a[i],candidate)){
            candidate = a[i];
            index = i;
        }
    }

    for(int i= 0;i<index;i++){
        if(!HaveAcquaintance(a[i],candidate))
            return -1;
    }
    return candidate;
}
```

^ | v · Reply · Share ›



**oops** · 2 years ago

it takes atmost  $2n$  calls to the function HaveAcquaintance.  
in this the first loop checks for the candidate element for the celebrity. that eler



everybody must know him/her.

first loop ensures that all element after candidate+1 know candidate and second loop ensures that candidate-1 knows or not..

please tell me if anything is wrong in this approach.

```
int celebrity(int a[],int size)
{
    candidate = 0;
    for(int i = 0; i<size;i++){
        if(!HaveAcquaintance(i,candidate)){
            candidate = i;
        }
    }

    for(int i= 0;i<candidate-1;i++){
        if(!HaveAcquaintance(i,candidate))
            return -1;
    }
    return candidate;
}
```

^ | v • Reply • Share ›



**frugal** • 2 years ago

This can be done in 2n function calls here is the solution:

we firstly find the candidate element for the celebrity.

first loop takes care of this. after finding the candidate celebrity, we check if it is

The approach is that the element who denies to know other element can be the celebrity. then it can't be.

so we check through the array to find the same. this part takes at most n calls to

and now the second part checks the element before the candidate element, if

celebrity .. .)

please tell me if it is wrong...!!

```
boolean printCelebrity(int arr[], int size) {  
  
    int candidate = 0;  
    for(int i = 0; i<size; i++){  
        if(!HaveAcquaintance(i, candidate)){  
            candidate = i;  
        }  
    }  
}
```

[see more](#)

^ | v • Reply • Share ›



**doom** • 2 years ago

I don't understand the use of auxiliary space for the above problem. The logic I already mentioned in the code.

So why not do this?

keep celeb as index 0.

run a loop from i=1 to n;

if celeb knows i, update celeb to i and increment i;

else increment i;

Calls to acq are n-1.

At the end of above iteration, we get a candidate for being celebrity.

Now check if our celeb does not know anyone (calls to acq n-1) and everyone return celeb else not found.

Total number of calls are 3(n-1).

Please help me to point out the flaw in this approach.

^ | v • Reply • Share ›



**Venki** → doom · 2 years ago

Your approach seems to be fine. I have used stack to easily explain the solution without any data structures as an exercise.

As it not trivial code, I would request you to provide the code. Our logic while running code. Don't forget that we are not interested in whether c

^ | v · Reply · Share ›



**doom** → Venki · 2 years ago

Just to clarify..

Whether the celebrity knows himself or not is insignificant.

For every person i, haveAcq(i,i) returns true.

Are the above assumptions correct?

^ | v · Reply · Share ›



**doom** → doom · 2 years ago

<http://ideone.com/titjR>

Here is the code.

^ | v · Reply · Share ›



**Mohammad Abuomar** → doom · 4 months ago

@Doom this won't work since you're only considering the backward in your first iteration, thus missing potentially i you've got 5 persons, where 3 knows 2 but doesn't know your first loop will keep 3 as the candidate and will ignore will fail thus returning no success.

^ | v · Reply · Share ›



**Guddu sharma** · 2 years ago

various test cases & they are working perfectly without using the below condition

```
// Last candidate was not examined, it leads one excess comparison (if)
if( HaveAcquaintance(C, B) )
    C = B;

if( HaveAcquaintance(C, A) )
    C = A;
```

Can you give some test cases where above condition become relevant?

Thank you.

^ | v • Reply • Share ›



**Venki** → Guddu sharma • 2 years ago

**@Guddu, the while loop runs till there is only one candidate left not be examined as it will not enter the loop. But we will have either candidate in the stack (i.e. after while loop we will have pseudo winner with the one left in the stack. Use a flag to discriminate between who would be the last winner. Instead I have used one more excess**

**The problem is, in the content I have explained to pop off two candidates code I am popping off only one, and keeping tracking track of last clauses. If we pop off two candidates and push the winner back to conditions.**

**Take the input like the first candidate who will be pushed into stack there is no celebrity.**

^ | v • Reply • Share ›



**Guddu sharma** → Venki • 2 years ago

Thanks, I got it.

^ | v • Reply • Share ›



**joker** • 2 years ago

hey can someone suggest some test case's

```
#include
```

```
#include
```

```
using namespace std;
```

```
bool HaveAcquaintance(int i, int j){
```

```
if(i==j) return 0;
```

```
int knows[][4] = { {0,0,1,1},
```

```
{0,0,1,0},
```

```
{0,0,0,0},
```

```
{0,0,1,0} };
```

```
return knows[i][j];
```

```
}
```

```
int Celebrity(int size)
```

```
{ int a,b;
```

```
stack s;
```

```
for(int i=0;i<size)
```

```
{
```

```
a=s.top(); s.pop();
```

[see more](#)

^ | v • Reply • Share ›



**Arnab sen** • 2 years ago

Here is my code using stack which have O(n) time complexity..

Plz comment on my code

```
int findCeleb(int celeb[])
```

```
{
```

```
stack s;
for(i=0;i<n;i++)
{
s.push(celeb[i]);
}
while(s.top>0)
{
a=stack.pop();
b=stsck.pop();
ab=HaveAcquaintance(a,b);
ba=HaveAcquaintance(b,a);
if(ab && !ba) s.push(b);
```

[see more](#)

^ | v • Reply • Share ›



**Arnab sen** • 2 years ago

Here is my code using stack which have  $O(n)$  time complexity..  
Plz cpomment on my code

```
int findCeleb(int celeb[])
{
stack s;
for(i=0;i<n;i++)
{
s.push(celeb[i]);
}
while(s.top>0)
{
a=stack.pop();
b=stsck.pop();
```

```
ab=HaveAcquaintance(a,b);
ba=HaveAcquaintance(b,a);
if(ab && !ba) s.push(b);
```

[see more](#)

^ | v • Reply • Share ›



**mvsinquest** • 2 years ago

@venki, according to your input celeb knows himself :-), so vipul code was fall below.

```
[sourcecode language="C++"]
#include <iostream>

using namespace std;

int HaveAcquaintance(int i, int j) {

    int knows[][4] = {
        {0, 0, 1, 0},
        {0, 0, 1, 0},
        {0, 0, 1, 0},
        {0, 0, 1, 0}
    };

    return knows[i][j];
}
```

[see more](#)

^ | v • Reply • Share ›



**Venki** → mvsinquest • 2 years ago

Yeah, the celebrity will be known to himself, otherwise, for simplicity, y

skip when  $i == j$  (as you did). It means we are not interested in "whethe

Check the code for the following input,

[sourcecode language=""]

{0, 0, 0, 0},

{0, 0, 1, 0},

{0, 0, 1, 0},

{0, 0, 1, 0}

and

{0, 0, 1, 0},

{0, 0, 1, 0},

{0, 0, 1, 0},

{0, 0, 0, 0}

But do you encourage this kind of trial and error coding? I request to fol

[see more](#)

^ | v • Reply • Share ›



**Arnab sen** → Venki • 2 years ago

Here is my code using stack which have  $O(n)$  time complexity.  
Plz comment on my code

```
int findCeleb(int celeb[])
{
    stack s;
    for(i=0;i<n;i++)
    {
        s.push(celeb[i]);
    }
}
```



```
J
while(s.top>0)
{
a=stack.pop();
b=stscck.pop();
ab=HaveAcquaintance(a,b);
ba=HaveAcquaintance(b,a);
if(ab && !ba) s.push(b);
}
```

[see more](#)

^ | v • Reply • Share ›



**mvsinquest** → Venki • 2 years ago

Argh, I should have written the code instead of copying it from r

I prefer to follow algorithmic approach but try to keep things sim  
complicating.

```
#include <iostream>

using namespace std;

int HaveAcquaintance(int i, int j) {

    int knows[][4] = {
        {0, 0, 1, 0},
        {0, 0, 1, 0},
        {1, 0, 1, 0},
        {0, 1, 1, 1}
    };
}
```

[see more](#)

^ | v • Reply • Share ›



**Venki** → mvsinquest • 2 years ago

Try for the input.

```
[sourcecode language=""]
```

```
{0, 0, 1, 0},
```

```
{0, 0, 1, 0},
```

```
{0, 0, 0, 0},
```

```
{0, 1, 1, 1}
```

or

```
{0, 0, 1, 0},
```

```
{0, 0, 1, 0},
```

```
{0, 0, 1, 0},
```

```
{0, 1, 1, 1}
```

Sorry dude, no more testing, it is not the way to design :

List down the steps. Identity corner cases, then code. Y

**Never rush to code, it is of last importance. Our the**

...

[see more](#)

^ | v • Reply • Share ›



**mvsinquest** → Venki • 2 years ago

The code is working perfectly, above mentioned code s  
missing in the my last but one comment.

```
/* Paste your code here (You may delete these lin
```

^ | v • Reply • Share ›



**Venki** → mvsinquest • 2 years ago

:) Aaha moments... I am enjoying testing your code.

Carefully check your matrix. It returns id 2, where as in 1 id 2 knew person with id 0. This is what I mean by trial a

I agree with you on simplicity, brevity and readability of c

One suggestion is, let us not interested in checking a pe

^ | v • Reply • Share ›



**mvsinquest** → Venki • 2 years ago

sorry my bad, didn't notice that i removed existing check

```
if(i != celb && HaveAcquaintance(celb, i)) {  
    cout << "No Celebrity." << endl;  
    exit(0);  
}
```

[sourcecode language="C"]

/\* Paste your code here (You may delete these lines if n

^ | v • Reply • Share ›



**vipul** • 2 years ago

This can be done in  $O(n)$  without using stack.

One iteration is enough:

```
void printCelebrity(int arr[], int size) {  
  
    int i = 0, celb = 0;
```

```

for (i=0;i<size;i++) {
    if(HaveAcquaintance(celb, i)) {
        celb = i;
    }
}
for (i =0;i<size;i++) {
    if(HaveAcquaintance(celb, i)) {
        printf("No Celebrity.");
        return;
    }
}
printf("Celebrity is at index: " + celb);
}

```

^ | v • Reply • Share ›



**saurabh** → vipul • 2 years ago

Your solution is based on assumption that all(N-1) persons know celeb  
Consider input { {0,0,1,1},{0,0,1,0},{0,0,0,0},{0,0,1,0}} ,your code  
will print "NO CELEBRITY" , But correct output is ID 2.  
So, Mathematical intuition behind the problem is :

if C is celebrity than -->  $\text{Matrix}[c][c][j] = 0$  for all j and  
 $\text{Matrix}[i][c][c] = 1$  for all i except i=c.(We don't need to bother about oth

^ | v • Reply • Share ›



**saurabh** → saurabh • 2 years ago

Your solution is based on assumption that all(N-1) persons know  
themselves. Consider input { {0,0,1,1},{0,0,1,0},{0,0,0,0},{0,0,1,  
will print "NO CELEBRITY" , But correct output is ID 2.  
So, Mathematical intuition behind the problem is :

if c is celebrity then  $\text{Matrix}[i][c] = 1$  for all i except i=c. (We don't need to bother al

^ | v • Reply • Share ›



**Venki** → vipul • 2 years ago

@Vipul, It is not such trivial implementation. Please check the output for example.

^ | v • Reply • Share ›



**Rohit** → Venki • 2 years ago

I made the same mistake as Vipul did. I missed to take into consideration celebrity and additionally would know other folks as well. Had the code would have been perfect except for one scenario if the celebrity was easily done using one simple if condition as I have appended below

```
void printCelebrity(int arr[], int size) {

    int i = 0, celb = 0;
    for (; i < size; i++) {
        if (HaveAcquaintance(celb, i)) {
            celb = i;
        }
    }
    //Appending the check for position 0
    if (celeb == 0)
        {if (HaveAcquaintance(1, 0)) celeb = 0;}
```

[see more](#)

^ | v • Reply • Share ›



I would say vipul is correct, it's just based on a simple logic  
if x knows y, we can eliminate x as celebrity otherwise eliminate y

Thanks!

^ | v • Reply • Share ›



**Venki** → mvsinquest • 2 years ago

Please check the code for all possible scenarios. To simplify  
assume  $N = 3$ .

As a sample, check the code against the matrix

{0->0->1->0}

{0->0->1->0}

{0->0->1->0}

{0->0->1->0}

It should return ID 3 (location 2).

^ | v • Reply • Share ›



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

**Contact Us!**

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team