

Minimum number of jumps to reach end

Given an array of integers where each element represents the max number of steps that can be made forward from that element. Write a function to return the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, then cannot move through that element.

Example:

Input: arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}

Output: 3 (1-> 3 -> 8 ->9)

First element is 1, so can only go to 3. Second element is 3, so can make at most 3 steps eg to 5 or 8 or 9.

Method 1 (Naive Recursive Approach)

A naive approach is to start from the first element and recursively call for all the elements reachable from first element. The minimum number of jumps to reach end from first can be calculated using minimum number of jumps needed to reach end from the elements reachable from first.

$minJumps(start, end) = Min (minJumps(k, end))$ for all k reachable from start

```
#include <stdio.h>
#include <limits.h>

// Returns minimum number of jumps to reach arr[h] from arr[l]
int minJumps(int arr[], int l, int h)
{
    // Base case: when source and destination are same
    if (h == l)
```

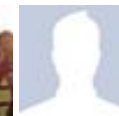
Google™ Custom Search



GeeksforGeeks



53,523 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

How To: Agile Testing

 utest.com/Agile_Testing

Reveal The Secrets of Agile Testing
Get Free Whitepaper to Learn
Today.

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](#)

```
return 0;

// When nothing is reachable from the given source
if (arr[l] == 0)
    return INT_MAX;

// Traverse through all the points reachable from arr[l]. Recursive
// get the minimum number of jumps needed to reach arr[h] from these
// reachable points.
int min = INT_MAX;
for (int i = l+1; i <= h && i <= l + arr[l]; i++)
{
    int jumps = minJumps(arr, i, h);
    if (jumps != INT_MAX && jumps + 1 < min)
        min = jumps + 1;
}

return min;
}
```

```
// Driver program to test above function
int main()
{
    int arr[] = {1, 3, 6, 3, 2, 3, 6, 8, 9, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Minimum number of jumps to reach end is %d ", minJumps(arr,
    return 0;
}
```

If we trace the execution of this method, we can see that there will be overlapping subproblems. For example, minJumps(3, 9) will be called two times as arr[3] is reachable from arr[1] and arr[2]. So this problem has both properties (optimal substructure and overlapping subproblems) of Dynamic Programming.

Method 2 (Dynamic Programming)

In this method, we build a jumps[] array from left to right such that jumps[i] indicates the minimum number of jumps needed to reach arr[i] from arr[0]. Finally, we return jumps[n-1].

```
#include <stdio.h>
#include <limits.h>
```

```
// Returns minimum number of jumps to reach arr[n-1] from arr[0]
int minJumps(int arr[], int n)
{
```

```

int *jumps = new int[n]; // jumps[n-1] will hold the result
int i, j;

if (n == 0 || arr[0] == 0)
    return INT_MAX;

jumps[0] = 0;

// Find the minimum number of jumps to reach arr[i]
// from arr[0], and assign this value to jumps[i]
for (i = 1; i < n; i++)
{
    jumps[i] = INT_MAX;
    for (j = 0; j < i; j++)
    {
        if (i <= j + arr[j] && jumps[j] != INT_MAX)
        {
            jumps[i] = jumps[j] + 1;
            break;
        }
    }
}
return jumps[n-1];
}

// Driver program to test above function
int main()
{
    int arr[] = {1, 3, 6, 1, 0, 9};
    int size = sizeof(arr)/sizeof(int);
    printf("Minimum number of jumps to reach end is %d ", minJumps(arr, size));
    return 0;
}

```

Thanks to [paras](#) for suggesting this method.

Time Complexity: $O(n^2)$

Method 3 (Dynamic Programming)

In this method, we build jumps[] array from right to left such that jumps[i] indicates the minimum number of jumps needed to reach arr[n-1] from arr[i]. Finally, we return arr[0].

```

int minJumps(int arr[], int n)
{

```

Shouldn't
you expect
a cloud with:

**ONE-CLICK
DEPLOYMENTS**

Experience the
Managed Cloud
Difference

TRY TODAY ►

 **rackspace**
the open cloud company

```

int *jumps = new int[n]; // jumps[0] will hold the result
int min;

// Minimum number of jumps needed to reach last element
// from last elements itself is always 0
jumps[n-1] = 0;

int i, j;

// Start from the second element, move from right to left
// and construct the jumps[] array where jumps[i] represents
// minimum number of jumps needed to reach arr[m-1] from arr[i]
for (i = n-2; i >=0; i--)
{
    // If arr[i] is 0 then arr[n-1] can't be reached from here
    if (arr[i] == 0)
        jumps[i] = INT_MAX;

    // If we can directly reach to the end point from here then
    // jumps[i] is 1
    else if (arr[i] >= n - i - 1)
        jumps[i] = 1;

    // Otherwise, to find out the minimum number of jumps needed
    // to reach arr[n-1], check all the points reachable from here
    // and jumps[] value for those points
    else
    {
        min = INT_MAX; // initialize min value

        // following loop checks with all reachable points and
        // takes the minimum
        for (j = i+1; j < n && j <= arr[i] + i; j++)
        {
            if (min > jumps[j])
                min = jumps[j];
        }

        // Handle overflow
        if (min != INT_MAX)
            jumps[i] = min + 1;
        else
            jumps[i] = min; // or INT_MAX
    }
}

return jumps[0];

```



Recent Comments

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

Write a C program to Delete a Tree. · 37 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 41 minutes ago

Sanjay Agarwal bool
tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily..."

Count trailing zeroes in factorial of a number · 1 hour 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

[▶ C++ Code](#)

[▶ Java Array](#)


```
}
```

Time Complexity: $O(n^2)$ in worst case.

Thanks to [Ashish](#) for suggesting this solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.




AdChoices 

[▶ Jumping Jumps](#)

[▶ Programming C++](#)

[▶ Array Max](#)

AdChoices 

[▶ An Array](#)

[▶ Code Number](#)

[▶ Int Byte Array](#)

Related Topics:

- [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](#)



8



Tweet

0



2

Writing code in comment? Please use ideone.com and share the link here.

105 Comments**GeeksforGeeks**

Sort by Newest ▾



Join the discussion...

**AlienOnEarth** · 17 days ago

Hi,

I have $O(n)$ solution. Can someone please tell me problem with the algorithm c

// Returns minimum number of jumps to reach arr[n-1] from arr[0]

```
int minJumps(int arr[], int n)
```

```
{
```

```
int *jumps = (int*)malloc(sizeof(int)*n); // jumps[n-1] will hold the result
```

```
int i=1, j=0;
```

```
jumps[0] = 0;
```

```
for(i=1;i<n;) {="" if(arr[j]+j="" <="" i)="" {="" j++;"="" }="" else="" {="" jumps[i]="ju  
printf("jumps:="" ");="" for(i="0;i<n;i++)" {="" printf("%d,="" ",jumps[i]);="" }="" }
```

^ | v .

**Faiz** · a month ago

is this a possible solution?

Starting from $i = 0$;

Find farthest max number in range (i+1, i+array[i])

set i = index of max value;

repeat until i + max >= n;

with farthest max i mean this

{1,5,6,3,2,6,4}

here the max is 6 but there are two 6. So return the index 5 since that 6 is the

^ | v .



Wellwisher · a month ago

Ready solution in C#

<http://onestopinterviewprep.bl...>

^ | v .



Srikant Aggarwal · 2 months ago

There is one On solution:

```
public int getMinJumps(int[] A) {  
    int i = 0;  
    int r = 0;  
    int length = A.length;  
    int[] J = new int[length];
```

```
    while(r < length-1) {  
        int num = A[i];  
        for (int j = 1; j <= num; ++j) {  
            if (j+r < length)  
                J[j+r] = J[i]+1;  
            else  
                break;  
        }  
        r += num;  
    }
```

```
i++;  
}  
  
return J[length-1];  
}
```

Since each element is scanned at most 2 times, this is an O(n) solution.

^ | v .



Wellwisher · 2 months ago

<http://onestopinterviewprep.bl...>

Array hopper problem.
Dynamic programming method.

Thanks.

^ | v .



Shivam Goel · 2 months ago

in Method 2:
it should be..

```
jumps[i] = min ( jump[i] , jumps[ j ] + 1 );
```

^ | v .



Vijay Apurva · 2 months ago

```
int minJumps(int arr[], int n)
```

```
{
```

```
int jumps[n]; // jumps[n-1] will hold the result
```

```
int i,j,count=0;
```



```

for(i=0;i<n;i++) jumps[i]="0;" if(arr[0]=="0") return="" 0;="" for(i="0;i<=n-1;i++){
;="" for(j="i+1;" j="" <="" n="" &&="" count="" ;j++,count--="" ){="" if(jumps[j]=""

jumps[j]=jumps[i]+1;

}

}

return jumps[n-1];

}
^ | v .

```



Nemil • 3 months ago

Isn't a simple greedy solution enough here
Why do we use DP here

^ | v .



sourav • 3 months ago

<script src="http://ideone.com/e.js/DPEYy" type="text/javascript"></script>

^ | v .



rihansh • 5 months ago

// a bit more optimisation

```

int JumpToEnd(int *A,int n){

if(!n||!A[0])

return -1;

int result[n];

```

```
rep(i,n)
```

```
result[i]=-1;
```

```
result[0]=0;
```

```
int temp=1;
```

```
for(int i=0;i<n;i++) {="" if(result[i]!=-1)" {="" for(int="" j="temp;j<=i+A[i]&
result[j]="result[i]+1;" }="" }="" temp="i+A[i];" }="" return="" result[n-1];="" }="">
```

1 ^ | v .



rihansh · 5 months ago

```
int JumpToEnd(int *A,int n){
```

```
int result[n];
```

```
if(n==0||A[0]==0)
```

```
return -1;
```

```
int i;
```

```
rep(i,n)
```

```
result[i]=-1;
```

```
result[0]=0;// no step to first element
```

```
for(i=0;i<n;i++) {="" if(result[i]!=-1)" for(int="" j="1;(j+i)<n&&j<=A
result[i+j]="result[i]+1;" else="" result[i+j]="minm(result[i+j],result[i]+1);" }="" re
```

^ | v .



rihansh · 5 months ago



```
{{int JumpToEnd(int *A,int n){
int result[n];
```

```
if(n==0||A[0]==0)
```

```
return -1;
```

```
int i;
```

```
rep(i,n)
```

```
result[i]=-1;
```

```
result[0]=0;// no step to first element
```

```
for(i=0;i<n;i++) {="" if(result[i!="-1)" for(int="" j="1;(j+i)&lt;n&amp;&amp;j&lt;=A
result[i+j]="result[i]+1;" else="" result[i+j]="minm(result[i+j],result[i]+1);" }="" if=
return="" result[n-1];}}="">
```

1 ^ | v .



Harshit · 5 months ago

It can be solved in $O(n)$. Please look at the following solution.

```
int jump(int A[], int n) {
```

```
int maxl = A[0];
```

```
int nMaxl = 0;
```

```
int lastlndex = 0;
```

```
vector<int> jumps(n,0);
```

```
jumps[0] = 0 ;
```

```
jumps[i]="jumps[lastIndex]+1;" nmaxi="max(nMaxl," i+a[i]);="" }="" return="" jui
```

^ | v .



Guest · 5 months ago

I don't understand the question. Isn't the path unique? Is it necessary to start fr
If yes, then what is meant by positions 'reachable from a particular position'

^ | v .



Abhi · 6 months ago

Does anyone have junit test cases for the method 3 implementation

^ | v .



vikas · 7 months ago

Can we use the same concept for the snake and ladder problem , where we h
to reach the end , assuming the dice produce favorable results ?

^ | v .



jv · 8 months ago

in Method 2 (Dynamic Programming)

why is the min check missing

jumps[i] = jumps[j] + 1; is updated directly without checking

some thing like

jumps[i] > jumps[j] + 1;

^ | v .



Anon15 → jv · 4 months ago

Exactly! There should be a checking to check whether the current valu
the already assigned value or not.

^ | v .



wu → jv · 8 months ago

because if $j < k < i$
and then $\text{jump}[j] \leq \text{jump}[k]$, so we don't have to make a min check.

3 ^ | v ·



Chirag Gupta → wu · 15 days ago

in the second method, why the second condition $\text{jump}[j] \neq \text{INT_MAX}$
condition ?

^ | v ·



meh → wu · 3 months ago

I don't think the list is sorted if that's what you're assuming.

^ | v ·



meh → meh · 3 months ago

Ah, it was the jump array, sorry about that!

^ | v ·



max · 9 months ago

```
#include
```

```
#include
```

```
int min_steps(int *,int n);
```

```
int maxi(int *arr,int,int);
```

```
int main()
```

```
{
```

```
int arr[100];
```

```
int i,n,d;
```

```
printf("how many elements\n");
```

```
for(i=0;i<n;i++)
{
scanf("%d",&d); arr[i]=d;
}
```

see more

^ | v .



Vinod • 10 months ago

In worst case time complexity $O(n^2)$

Array set = {1,1,1,1,1,1}

In best case time complexity $O(n)$

Array set = {10,1,1,1,1,1}

Solution would work for all cases like {1, 3, 6, 1, 0, 9}

```
#include<stdio.h>

int min_steps(int a[],int size){
    int i,j,step=0;

    // i -> it starts from end
    // j -> it starts from end-1
    // k is use to check whether there exist a path to reach a[k]
    // If any path is exists then there has to be change in the value

    for(i=size-1;i>0;)
```

see more

^ | v .



Raman · 10 months ago

```
class Solver
{
    public int getMinJumps(int a[])
    {
        int MinFromHere []=new int[a.length];
        for(int i=(a.length-1);i>=0;i--)
        {
            if(i==(a.length-1))
                MinFromHere[i]=0;
            else if(a[i]+i>=a.length-1)
                MinFromHere[i]=1;
            else if(a[i]==0)
                MinFromHere[i]=Integer.MAX_VALUE;
            else if(a[i]!=0)
                getMin(MinFromHere,i,a[i]);
        }
        return MinFromHere[0];
    }
}
```

see more

^ | v ·



Shankar · 10 months ago

Farthest way logic wont work the jumps are not in ascending order so it is pos to the farthest node

```
/* Paste your code here (You may delete these lines if not writing code)
```

^ | v ·



Thanks for pointing out the mistake. Here is the rectified code which also has minimum jumps.

```
[sourcecode language="C++"]
//DPP.Min number of jumps.O(n^2)
#include<iostream>
#include<cstdio>
#define inf 2000
using namespace std;
int main()
{int N;

scanf("%d",&N);//Number of entries

int a[N];//array of numbers

int jumps[N];//jumps[i] number of jumps from i to N-1 or end

int next[N];//to print the path with minimum number of jumps.
```

[see more](#)



jaskaran1 • 10 months ago

This can be done in $O(n)$. My solution is similar to the second one but no need for second solution. `jumps[i]` is number of jumps from `i` to end. So if a particular element is at index `j` then `jumps[i] = jumps[j] + 1`.

```
[sourcecode language="C++"]
/*
#include<iostream>
#include<cstdio>
#define inf 2000
```



```

using namespace std;
int main()
{int N;
scanf("%d",&N);//Number of entries
int a[N];//array of numbers
int jumps[N];//jumps[i] number of jumps from i to N-1 or end
int path[N];
jumps[N-1]=0;
for(int i=0;i<N-1;i++)

```

[see more](#)

^ | v .



jaskaran1 · 10 months ago

This can be done in $O(n)$. My solution is similar to the second one but no need second solution

if $i < j$ then $jumps[i] = jumps[j]$ where $jumps[i]$ is number of jumps from i to farthest reachable vertex from i then $jumps[i] = jumps[j] + 1$.

/*//DPP.Min number of jumps. $O(n)$

```
#include<iostream>
```

```
#include<cstdio>
```

```
#define inf 2000
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
int N;
```

```
scanf("%d",&N)//Number of entries
```

```
int a[N];//array of numbers
```

```
int jumps[N];//jumps[i] number of jumps from i to N-1 or end
```

[see more](#)

^ | v ·



ultimate_coder · 11 months ago

Modified method 1, no repeated calls.

Let me know its complexity.

```
for (i = l+1; i <= h && i <= l + arr[l]; i++)
{
    if(!htol[i][h])
    {
        htol[i][h] = minJumps(arr, i, h);
    }
    if(htol[i][h] != INT_MAX && htol[i][h] + 1 < min)
        {htol[l][h]=htol[i][h] + 1;}
}

return htol[l][h];
```

^ | v ·



Lakshay Nagpal · 11 months ago

I suppose this can be done greedily:

Time complexity $O(n)$.

Here is the pseudocode:

```
while(i!=n){
    maxindex=i;
    for(int j=i;j<i+a[i];j++)
        if(j+a[j]>maxindex+a[maxindex])
```

```

}
i=j;
minjmp++;
}

```

3 ^ | v .



Mohammad Faizan Ali · 11 months ago

Hey admin I think my method is better. It looks as if it is in $O(n^2)$ but the inner indexes, so the solution is in $O(n)$.

Starting from the first to the end we calculate the no. of hops required to reach hops to reach any index from index i is less than the hops to reach any index f . The array `min_dist` stores the no. of hops required to reach any index.

Initially all values of `min_dist` is 500 (very large) except for the first index which we keep track of the index before which values of `min_dist` are updated/final. Variable `temp` stores the farthest index reachable from a given index. If we can

Code in C++:

```

#include<stdio.h>
int min_hops(int *arr, int x);
int min(int, int);
int main()
{
    int x, i;

```

[see more](#)

^ | v .



Sam · a year ago

Hi,

The complexity of this algorithm is $O(n)$. @GeeksforGeeks
Kindly please let me know if there is anything wrong with the approach.

```
public class Jumping {  
  
    public static int jump=0;  
  
    public static void main(String args[]) {  
        int arr[] = {1, 3, 6, 3, 2, 3, 6, 8, 9, 5};  
        int j=jump(arr,arr.length-1);  
        System.out.println("The min jump is "+ j);  
    }  
  
    public static int jump(int array[], int n) {  
  
        int []jumpArray = new int[n+1];  
  
        if(n==0) {
```

[see more](#)

5 ^ | v •



sakib → Sam • 4 months ago

arr[] = {1, 3, 6, 1, 0, 9}
for this array the correct ans = 3

but ur code show ans = 2
so ur code have small mistake

^ | v •



aspiring coder • a year ago

can the array elements contain negative integers??
where the negative integers imply backward jumps?

/* Paste your code here (You may **delete** these lines **if not** writing c)

^ | v .



abhishek08aug · a year ago

Intelligent :D

/* Paste your code here (You may **delete** these lines **if not** writing c)

^ | v .



keshav · a year ago

```
#include<stdio.h>
#include<conio.h>
```

```
int func(int *arr,int *next,int *jump,int n,int index);
int main()
{
    int i,*arr,*next,*jump,n;
    printf("enter number of elements to input\n");
    scanf("%d",&n);
    arr=(int *)(malloc(n*sizeof(int)));
    next=(int *)(malloc(n*sizeof(int)));
    jump=(int *)(malloc(n*sizeof(int)));
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
        next[i]=i;
        jump[i]=0;
    }
}
```

[see more](#)



Sudhanshu Shekhar · a year ago

I am fairly new to Dynamic Programming. Can you please tell me if the code is a correct question?

```
[sourcecode language="C++"]
for(i=0;i<n;i++)jumps[i]=INT_MAX;

for(i=0;i<n;i++)
{
    if(arr[i]==0)continue;
    for(j=i+1;j<=i+arr[i];j++)
    {
        if(jumps[j]>jumps[i]+1)jumps[j]=jumps[i]+1;
    }
}
```

^ | v ·



gargsanjay · a year ago

We can do this question in $O(n)$ time. let $arr[k]$ has value L , so we examine $arr[k+1]$ to $arr[k+L]$ as the next step where j varies from 1 to L .

```
/* Paste your code here (You may delete these lines if not writing code)
```

1 ^ | v ·



Shredder → gargsanjay · a year ago

That is correct.

That is a Greedy strategy for solving the problem.

I understand that when both Dynamic and Greedy are applicable to a problem, the Greedy solution is often the DP solution. This is an example of this.

^ | v ·



zyfo2 → Shredder · a year ago

right. greedy algorithm is the one here

^ | v ·



st0le → gargsanjay · a year ago

k varies from 1 - L, and for each k varies for 1 - L.

That's N^2 !



```
/* Paste your code here (You may delete these lines if not wr
```

^ | v ·



gargsanjay → st0le · a year ago

k does not vary it is the next step choosen.



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

^ | v ·



naren596 · a year ago

Can anyone find probs in my code. I tried to improve it using previous index

```
public void find(int[] ary)
{
    int prevind=0;
    int minjumps[]=new int[ary.length];
    minjumps[0]=0;

    for(int i=1;i<ary.length;i++)
    {
        if(ary[i]==0) return;
```

```

if(i-prevind<=ary[prevind])
{
    minjumps[i]=minjumps[prevind]+1;
}
else
{
    int j=prevind+1;

```

see more

^ | v .



coderAce · a year ago

In method2-Inner for loop, the check for `jump[j]!-INT_MAX` is not required. `j` is a `jump[i]` can be equal to `INT_MAX` in every iteration. For any `j<i`, `jump[j]=""` is=""
whole="" point="" of="" dp.=">

^ | v .



Patrick · a year ago

Time Complexity $O(n)$

Space Complexity $O(1)$

please let me know if I am wrong.

```

/* Paste your code here (You may delete these lines if not writing c
int minimumNoJumps(int * A, int len){
    int current_jump,i=0,count=0;
    while(i<len){
        current_jump=A[i];
        printf("%d->",current_jump);
        if(i+current_jump>=(len-1))
            break;
        int j=1,max=0,max_index;
        while(j<=current_jump){

```



```
if(A[i+j]>max){
    max=A[i+j];
    max_index=i+j;
}
```

see more

^ | v .



cakester → Patrick · 4 months ago

a = [2, 5, 10, 1, 1, 3, 0, 0, 2, 5, 1, 1, 1, 2, 0, 1, 1] doesnt work, but it sup

^ | v .



coderAce → Patrick · a year ago

Sorry, my earlier comment was not targeted at your solution. But your solution. Thanks for sharing it.

^ | v .



Patrick → coderAce · a year ago

:)

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

^ | v .



coderAce → Patrick · a year ago

In method2-Inner for loop, the check for jump[j]!-INT_MAX is not require and only jump[i] can be equal to INT_MAX in every iteration. For any j<i the whole point of DP.

^ | v .

Load more comments



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

Contact Us!

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