GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

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)
     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]. Recursively
   // 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, 0, n-1)
  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>

int min(int x, int y) { return (x < y)? x: y; }

// 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] = min(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;
}
```

Output:

```
Minimum number of jumps to reach end is 3
```

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)
{
    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 direcly 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];
}
```

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.

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Tags: Dynamic Programming

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

nivak • 15 days ago

we dont need min() function for 2nd method right?

• Reply • Share ›

Sahil Sethi • 21 days ago

It can also be done in O(nlogn) using segment trees and Lazy Propogation. With ofcourse memory complexity O(n).

arr[]=given array.
dist[]=minimum no of jumps required to reach (i) from 0.

Here is the idea: Let us suppose we have already computed minimum jumps to reach array element (i) i.e dist[i]. Now all the array elements from (i) to (i+arr[i]) can be reached from (i) with atmost dist[i]+1 jumps

almost dist[i]+1 jumps.

Pseudo code: Build a segment tree with n leaf nodes and initialise first node to zero and the rest to INT_MAX. This is because dist[0]=0 and the rest will be calculated on the fly.

Traverse the array from left to right starting from 0. For every element in the array, calculate the minimum distance from segment tree for that node, then range update the segment tree.

Range update query will be like (jump_from=i, jump_till=i+arr[i], no_of_jumps=dist[i]+1). Range update will ensure every node is updated only to the minimum number of jumps.

Here is my code for your reference: http://ideone.com/Oja30n

• Reply • Share ›

Mahesh Kasana • a month ago

simplest solution worst case n*n

void array :: jumps()

{

int *jum;

int i,j,k,min;

jum=new int[n];

for(i=0;i<n;i++) jum[i]="n;" jum[0]="0;" for(i="0;i<n-1;i++)" {="" k="jum[i]+1;" for(j="i+1;j<n" &&="" j<="i+arr[i];j++)" jum[j]="((jum[j]<k)?" jum[j]="" :="" k);="" }="" cout<<"\nminimum="" no="" of="" jump's="" req="" :="" "<<jum[n-1];="" return;="" }="">

• Reply • Share ›

mrigendra kashyap • a month ago

hey friends check out this solution

int[] arr = { 1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9 };

int[] ar = new int[arr.Length];
int step = 1;
ar[arr.Length - 1] = -1;

for (int i = arr.Length - 2; i >= 0; i--)
{
if (arr[i] == 0)
ar[i] = -1;
else if (arr[i] >= step)
ar[i] = 1;
else

```
{
int min = int.MaxValue;
for (int j = 1; j <=arr[i]; j++)
{
if (ar[i+j]< min && ar[i+j]>0)
min = ar[i+j];
}
ar[i] =min+ 1;
}
step++;
}
Console.WriteLine(ar[0]);
```


Santosh Kumar • a month ago

```
public static void minJump(int[] arr, int[] path, int m){

int reach = 0;
for(int i = m;i>=0;i--){
if(arr[i]>m)
reach = i;
}
path[path.length]=reach;
if(reach>0)
minJump(arr, path, reach);
}
```


newbie • a month ago

```
int minJumpToReachEnd(int arr[], int n)
{
if(n == 0 || n == 1)
return 0;

for(int i = 0; i < n; i = i + arr[i]) //handling case when jumped at arr[i] = 0
{
if(arr[i] == 0)
return -1;
}

for(int i = 1; i < n; i++)
{
arr[i] = max(arr[i], arr[i-1]+1);
}
```

```
int steps = 0;
for(int i = 0; i < n; i = i + arr[i])
{
steps++;
}
return steps;
}
```

• Reply • Share ›

Tequila • a month ago

DP O(n) soln

http://ideone.com/AMeDFq

• Reply • Share ›

Siya Tequila • a month ago

Its not o(n) in fact you are checking recursively all the possibilities .

• Reply • Share ›

Tequila Siya • a month ago

it wont check overlappind subproblems twice . I have used memoized DP table. top-down soln

• Reply • Share ›

Siya Tequila • a month ago

But you are calling jump multiple times. Take example of a[5]={3,3,3,3,3} now acc to your code
call1.for jump (0) this for loop will run for 3 times i.e (n-1) times.call2.for jump(i+j) which is jump(1) where again loop run for (n-1) times.call3. for jump(i+j) which is jump(2) loop run for (n-1) times.similarly till jump(n-1) for every time you are running (n-1) times.
which obviously not gives O(n) in worst case.

• Reply • Share ›

Tequila Siya • a month ago

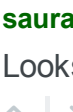
sorry my bad. I forgot the for loop .
its O(n^2) .Thanks for correcting me.

• Reply • Share ›

Tejwinder • a month ago

```
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] = min(jumps[i], jumps[j] + 1);
break;
}
}
}
// Here jumps[i]= jumps[j]+1 should also work , since
break is already added
Correct me if I am wrong.
```

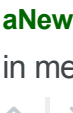


saurabh tiwari • a month ago

Looks quite similar to finding the shortest path problem, where each edge is of weight 1.

Abettik saurabh tiwari • a month ago

i agree, this is Dijkstra's shortest path in my view, but also has a DP solution. The difference is that DP can give you all paths, DSP will give you THE shortest path ?



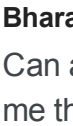
aNewBornCoder • a month ago

in method 2 there is a break statement. is it correct?

Sneha • 2 months ago

@GeeksForGeeks
Why is the break statement in inner for loop of method 2?

1



Bharath • 2 months ago

Can anyone tell me how to insert the data in the form of a file for third method, if possible give me the code
help me please

GOPI GOPINATH Bharath • 2 months ago

Dint get ur question.

Bharath GOPI GOPINATH • 2 months ago

Here we are declaring array values

Here we are declaring array values
instead of that i need to put all the values in a file

• Reply • Share ›

Guest • 2 months ago

Why should we go for DP even when it is O(n^2). Just use the BFS and check for the end element in the array, we can get the minimum number of jumps. This actually becomes a simple graph connectivity problem.

• Reply • Share ›

coder12489 • 2 months ago

I think we can solve it using greedy approach which is very simple we just need to add value to their corresponding index for example arr[3] = [7], 3+7 = 10. It simply means from array 3rd index value we can MAXIMUM reach upto 10th index value. We can compute this for every element of array can keep picking maximum reach element. Plz correct me if u guys find some flaw in this approach, if u want i can post program as well.
Thanx in advance

• Reply • Share ›

Aditya Goel • 3 months ago

2nd Method - http://ideone.com/p8TYsL

• Reply • Share ›

mantri Aditya Goel • a month ago

Oh man.. You are everywhere ? :D

• Reply • Share ›

Chouhan Vikram Singh • 3 months ago

Complexity = O(n)

```
static int minJumps(int [] A) {

int jumps=0;

int max=A[0];

int count = 0;

for(int i =0; i < A.length; i++) {

if(count == 0){

count = max;

}

count--;
```

count--;

if(count == 0 && i != A.length -1){

see more

• Reply • Share ›

Aditya Goel ➦ Chouhan Vikram Singh • 3 months ago

Care to provide comments?

• Reply • Share ›

Sai Nikhil • 3 months ago

Forgot to write base case for n=1 and arr[0]==0 in method 3. It should return INT_MAX.

• Reply • Share ›

Anand Thakur • 3 months ago

```
public class MinJump {

static int[] arr = { 1, 3, 5, 8,9, 2, 6, 7, 6, 8, 9};

public static void main(String[] s){
System.out.println(findMinJump(0));
}

public static int findMinJump(int i){

int jump = 0;
if(i == (arr.length-1)){
return jump ;
}

int k = arr[i];
int last = 0;
for(int j=i+1; j<=i+k && j<(arr.length-1); j++){

jump = findMinJump(j) + 1;
```

see more

• Reply • Share ›

Guest • 3 months ago

```
public class MinJump {

static int[] arr = { 1, 3, 5, 8,9, 2, 6, 7, 6, 8, 9};
static boolean isFirstJump = false;

public static void main(String[] s){
System.out.println(findMinJump(0));
```

```
}

public static int findMinJump(int i){

int jump = 0;
if(i == (arr.length-1)){
return jump ;
}

int k = arr[i];
int last = 0;
for(int j=i+1; j<=i+k && j<(arr.length-1); j++){
```

see more

• Reply • Share ›

nlogn • 3 months ago

in first DP solution why we need jumps[j] != INT_MAX. We are always calculating minimum, so if i am unable to reach i-th position, there are no way i can reach i+1 position. And, as it is a DP solution, which means it also support optimal substructure, which also means if i do not have a solution for i-1 index i can not make one for i-th index. isn't that so? what really am I missing here?

• Reply • Share ›

abhineet kumar • 4 months ago

why we need to take min(jumps[i], jumps[j] + 1); in method 2, is not jumps[j]+1 will always be minimum?

• Reply • Share ›

rohit_90 • 4 months ago

@GeeksforGeeks , Here is O(n) solution.

http://ideone.com/PE6H60

• Reply • Share ›

Vignesh Miriyala • 4 months ago

why not greedy which should solve in O(n) ?

• Reply • Share ›

Ambika • 4 months ago

@GeeksforGeeks , why in naive approach , min is not made universal or passed through the function.
i.e why min value remains same on each recursive calling

• Reply • Share ›

avinash • 4 months ago

http://www.writeulearn.com/jum...

• Reply • Share ›

Sumit Kesarwani • 5 months ago

HI All,

Can u please tell me why this condtion is using in 2nd method..

i <= j + arr[j]

• Reply • Share ›

Ambika ➦ Sumit Kesarwani • 4 months ago

To check whether we can reach to i from index j.
Let say from i=4; j=0; arr[j]=4;
then we can reach at i from j

• Reply • Share ›

Deepesh Maheshwari ➦ Ambika • a month ago

@ambika condition shouldn't like this :

j + arr[j] <= i

As, if j goes out of the range of i, it is not required condition,it should be within the range of i.

Also, why we are taking min value?

• Reply • Share ›

Sumit Kesarwani ➦ Ambika • 4 months ago

Thanks geek

• Reply • Share ›

shiva • 6 months ago

Is n't it good enough to choose maximum element in the given range and step thru?

for example

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. here choose 9

• Reply • Share ›

toothless ➦ shiva • 4 months ago

Can you solve for this series using your algo:
1,3,5,6,7,5,4,3,3,3,0,0,9

Answer is: 4(1,3,6,3)
I don't think your algorithm will find that answer.
1 • Reply • Share ›

john ➚ toothless • 4 months ago
1,3,7,3 can also work
• Reply • Share ›

Guest • 6 months ago
What is the complexity of naive approach?? It looks like O(n^2).
• Reply • Share ›

Ravi • 7 months ago
There is an worst-case O(n) solution for it.
• Reply • Share ›

Priyal Rathi • 7 months ago
Another recursive approach: http://ideone.com/2TT4No
• Reply • Share ›

ryan • 7 months ago
@GeeksforGeeks
robust code in O(n) time complexity

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

int jump(int A[], int n)
{
if(n==0||n==1)
return 0;

int Ar[n],i=1,j=0;
Ar[0]=0;
for(int p=1;p<n;p++) ar[p]="INT_MAX;" while(i<n)="" {="" if(j="">=i)
i++;
else if(i-j<=A[j])
{
Ar[i]=Ar[j]+1;
i+=1;
```

see more

• Reply • Share ›

Chirayu Modi • 7 months ago

O(n) Solution :

http://ideone.com/6E5xrA

• Reply • Share ›

sukanya ➚ Chirayu Modi • 7 months ago

plz check it for {2,0,0,3}
{1,1,0,3}

• Reply • Share ›

ryan ➚ sukanya • 7 months ago

a bit of modification is needed

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

int jump(int A[], int n)
{
if(n==0||n==1)
return 0;

int Ar[n],i=1,j=0;
Ar[0]=0;
for(int p=1;p<n;p++) ar[p]="INT_MAX;" while(i<n)="" {="" if(j="">=i)
i++;
else if(i-j<=A[j])
{
Ar[i]=Ar[j]+1;
i+=1;
}
```

see more

• Reply • Share ›

Ameya • 7 months ago

I solved it in O(n) time on leetcode.
It was accepted by all of their test cases.
And I think I'm correct also.

```
public class Solution {

public int jump(int[] A)

{
```

```
int steps = 0;

if(A.length ==0 || A.length==1) return 0;

for(int i=1; i< A.length-1 ;i++)

{

A[i]=Math.max(A[i], A[i-1] -1);
```

see more

4 • Reply • Share ›

The_Geek Ameya • 2 months ago

Awsum !

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

- [Interview Experiences](#)
- [Advanced Data Structures](#)
- [Dynamic Programming](#)

- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

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

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

@GeeksforGeeks i don't n know what is this long...

Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago