

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



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



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

```

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)
{

```



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

```

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 jumps[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 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. · 15 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 19 minutes ago

Sanjay Agarwal bool
tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 44 minutes ago

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

Count trailing zeroes in factorial of a number · 45 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

► [C++ Code](#)

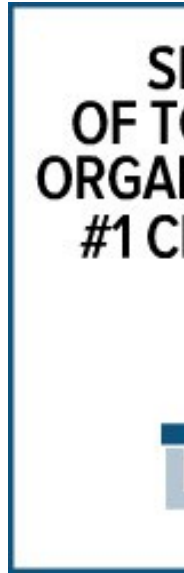
► [Java Array](#)


```
}
```

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 in Array](#)

Related Topics:

- [Remove minimum elements from either side such that \$2 \cdot \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.

@geeksforgeeks, **Some rights reserved**

Contact Us!

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