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.

Recommended: Please solve it on "*PRACTICE* " first, before moving on to the solution.

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

---

C

```
#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;
}
```

Run on IDE

Python3

```
# Python3 program to find Minimum
# number of jumps to reach end

# Returns minimum number of jumps
# to reach arr[h] from arr[l]
def minJumps(arr, l, 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 float('inf')

    # 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.
    min = float('inf')
    for i in range(l + 1, h + 1):
        if (i < l + arr[l] + 1):
            jumps = minJumps(arr, i, h)
            if (jumps != float('inf') and
                      jumps + 1 < min):
                min = jumps + 1

    return min
```

```
# Driver program to test above function
arr = [1, 3, 6, 3, 2, 3, 6, 8, 9, 5]
n = len(arr)
print('Minimum number of jumps to reach',
     'end is', minJumps(arr, 0, n-1))

# This code is contributed by Soumen Ghosh
```

Run on IDE

Output:

```
Minimum number of jumps to reach end is 4
```

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].

C / C++

```
#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;
}
```

Run on IDE

Java

```
// JAVA Code for Minimum number of jumps to reach end
class GFG{

private static 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 Integer.MAX_VALUE;  // if first element is 0,
                                    // end cannot be reached

    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] = Integer.MAX_VALUE;
         for (j = 0; j < i; j++)
         {
              if (i <= j + arr[j] && jumps[j] != Integer.MAX_VALUE)
              {
                  jumps[i] = Math.min(jumps[i], jumps[j] + 1);
                  break;
              }
          }
    }
        return jumps[n-1];
    }

// driver program to test above function
public static void main(String[] args) {
    int arr[] = {1, 3, 6, 1, 0, 9};

    System.out.println("Minimum number of jumps to reach end is : "+
                                  minJumps(arr,arr.length));
    }
}

// This code is contributed by Arnav Kr. Mandal.
```

Run on IDE

Python3

```python
# Python3 program to find Minimum
# number of jumps to reach end

# Returns minimum number of jumps
# to reach arr[n-1] from arr[0]
def minJumps(arr, n):
    jumps = [0 for i in range(n)]

    if (n == 0) or (arr[0] == 0):
        return float('inf')

    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 in range(1, n):
        jumps[i] = float('inf')
        for j in range(i):
            if (i <= j + arr[j]) and (jumps[j] != float('inf')):
                jumps[i] = min(jumps[i], jumps[j] + 1)
                break
    return jumps[n-1]

# Driver Program to test above function
arr = [1, 3, 6, 1, 0, 9]
size = len(arr)
print('Minimum number of jumps to reach',
      'end is', minJumps(arr,size))

# This code is contributed by Soumen Ghosh
```

Run on IDE

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].

C++

```cpp
// CPP program to find Minimum
// number of jumps to reach end
#include<bits/stdc++.h>
using namespace std;

// Returns Minimum number of
// jumps to reach end
```

```
int minJumps(int arr[], int n)
{
    // jumps[0] will hold the result
    int *jumps = new int[n];
    int min;

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


    // 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 (int 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
        {
            // initialize min value
            min = INT_MAX;

            // following loop checks with all
            // reachable points and takes
            // the minimum
            for (int 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];
}

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

```
}
```

Run on IDE

Python3

```python
# Python3 progrma to find Minimum
# number of jumps to reach end

# Returns Minimum number of
# jumps to reach end
def minJumps(arr, n):

    # jumps[0] will hold the result
    jumps = [0 for i in range(n)]

    # Minimum number of jumps needed
    # to reach last element from
    # last elements itself is always 0
    # jumps[n-1] is also initialized to 0

    # 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]
    # form arr[i]
    for i in range(n-2, -1, -1):

        # If arr[i] is 0 then arr[n-1]
        # can't be reached from here
        if (arr[i] == 0):
            jumps[i] = float('inf')

        # If we can directly reach to
        # the end point from here then
        # jumps[i] is 1
        elif (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:
            # initialize min value
            min = float('inf')

            # following loop checks with
            # all reachavle points and
            # takes the minimum
            for j in range(i + 1, n):
                if (j <= arr[i] + i):
                    if (min > jumps[j]):
                        min = jumps[j]

            # Handle overflow
            if (min != float('inf')):
                jumps[i] = min + 1
            else:
                # or INT_MAX
                jumps[i] = min

    return jumps[0]
```

```
# Driver program to test above function
arr = [1, 3, 6, 3, 2, 3, 6, 8, 9, 5]
n = len(arr)
print('Minimum number of jumps to reach',
      'end is', minJumps(arr, n-1))

# This code is contributed by Soumen Ghosh
```

Run on IDE

Output:

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

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

Minimum number of jumps to reach end | Set 2 (O(n) solution)

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.

GATE CS Corner Company Wise Coding Practice

Dynamic Programming

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Dynamic Programming | Set 34 (Assembly Line Scheduling)
Dynamic Programming | Set 14 (Maximum Sum Increasing Subsequence)
Dynamic Programming | Set 7 (Coin Change)
Maximum size square sub-matrix with all 1s
Dynamic Programming | Set 25 (Subset Sum Problem)
K maximum sums of non-overlapping contiguous sub-arrays
Largest rectangular sub-matrix having sum divisible by k

Minimum number of deletions to make a string palindrome | Set 2
Count ways to reach the nth stair using step 1, 2 or 3
Newman-Conway Sequence

(Login to Rate)

2.8 Average Difficulty : **2.8/5.0**
Based on **202** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!