LeetCode Jump Game II: Greedy vs DP vs BFS

Overview

Greedy algorithm could solve the LeetCode problem Jump Game II in linear time, another two alternatives are DP and BFS with more time complexit

LeetCode Jump Game II Problem

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Your goal is to reach the last index in the minimum number of jumps.

For example:
Given array A = `[2,3,1,1,4]`

The minimum number of jumps to reach the last index is `2`. (Jump `1` step from index 0 to 1, then `3` steps to the last index.)

Greedy, DP, BFS Solution Analysis

(1) DP: let F(i) denote the minimum number of jumps, then we have F(i) = min(F(j)) + 1 where j = 0, … i – 1, this is O(N^2) approach and will get TLE by the OJ

(2) BFS: For each i in A, set the number of steps to all the reachable potions (not visited ones) as the number of steps to reach i plus 1, this is O(N * max(A[i])), and still will get TLE by the OJ

(3) Greedy: still, keep the current maximum reach distance, and the number of steps to reach this current maximum distances, and keep another variable to record the next maximum reachable distance, which cost the current steps plus 1. The key idea behind is that, all the positions before the maximum reachable distance would be able to be reached! Then we linear scan the array to keep updating the current maximum and the next maximum as well as the number of steps. We can achieve the linear time algorithm.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

The following code is accepted by the LeetCode OJ to pass the Jump Game II Problem:

```
int jump(int A[], int n) {
    if (n <= 1) return 0;
    int maxReachableDistance = 0;
    int maxNextAvailableDist = 0;
    int minSteps = 0;

    for (int i = 0; i < n; ++i) {
        if (i > maxReachableDistance) {
            if (maxNextAvailableDist > maxReachableDistance) {
                maxReachableDistance = maxNextAvailableDist;
                ++minSteps;
            }
            else
                return -1;
        }
        maxNextAvailableDist = max(maxNextAvailableDist, A[i] + i);
        if (maxNextAvailableDist >= n - 1) return minSteps + 1;
    }
}
```

Remarks: The key idea behind the linear algorithm is that instead of keeping to know

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

every position is reachable by how many steps, we only need to keep a single maximum reachable distances and the steps needed. So every time we only need to update this single value in constant time rather than update a linear portion of positions.

Summary

Greedy algorithm could solve the LeetCode problem Jump Game II in linear time, another two alternatives are DP and BFS with more time complexit.

Written on May 16, 2013

« LeetCode Validate Binary Search Tree: Bottom Up, Top Down and Inorder

LeetCode Jump Game: Greedy vs DP vs BFS »

Copyright © 2016 Sigmainfy. All Rights Reserved.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com