Jump Game II

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.

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.)

Note:
You can assume that you can always reach the last index.

Solution 1

I try to change this problem to a BFS problem, where nodes in level i are all the nodes that can be reached in i-1th jump. for example. 2 3 1 1 4 , is 2|| 3 1|| 1 4 ||

clearly, the minimum jump of 4 is 2 since 4 is in level 3. my ac code.

```
 int jump(int A[], int n) {
     if(n<2)return 0;
     int level=0,currentMax=0,i=0,nextMax=0;

     while(currentMax-i+1>0){       //nodes count of current level>0
         level++;
         for(;i<=currentMax;i++){   //traverse current level , and update the max 
reach of next level
            nextMax=max(nextMax,A[i]+i);
            if(nextMax>=n-1)return level;   // if last element is in level+1,  th
en the min jump=level
         }
         currentMax=nextMax;
     }
     return 0;
 }
```

written by enriquewang original link here

Solution 2

Hi All, below is my AC solution:

```
public int jump(int[] A) {
    int maxReach = A[0];
    int edge = 0;
    int minstep = 0;

    for(int i = 1; i < A.length; i++) {
        if (i > edge) {
            minstep += 1;
            edge = maxReach;
            if(edge > A.length - 1)
                return minstep;
        }
        maxReach = Math.max(maxReach, A[i] + i);
        if (maxReach == i):
            return -1;
    }

    return minstep;
}
```

When iterate the array, I set an edge for the Search phase, which means that if I exceeds the edge, the minstep must add one and the maxReach will be update. And when the last index is within the range of the edge, output the minstep.

[2, 3, 1, 1, 4]

First, the edge is 0; Second, after start iterate the array, it exceeds the edge 0 when reaching the A[0] and update the edge to 2; Third, after it reach the A[2], it exceeds the edge 2 and update the new edge to the maxReach 4. Finally, end of the array is inside the edge, output the minstep.

written by zhoutsby original link here

Solution 3

```
public int jump(int[] A) {
    int sc = 0;
    int e = 0;
    int max = 0;
    for(int i=0; i<A.length-1; i++) {
        max = Math.max(max, i+A[i]);
        if( i == e ) {
            sc++;
            e = max;
        }
    }
    return sc;
}
```

written by adam20 original link here