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.

Determine if you are able to reach the last index.

For example:  
A = [2,3,1,1,4], return true.

A = [3,2,1,0,4], return false.

题目：

我们从第一个出发，第一个值为2，我们可以最远跳两个位置，也就是调到1，也可以调到小于2的是位置，也就是3，依次类推，如果由一条路径能跳到4的话返回true，否则返回false。

这道题我首先的想法是就是回溯法。

但是因为我对回溯法不是很熟练，没有写出来。之前做的排列组合这种回溯法用到了一个保存本次遍历结果的list，回溯的时候，是再那个list中删除失败的当前数据，所以我这次想到的是用一个sum保存距离开始位置的距离。这样明显是多此一举的。

我们直接跳就好了，跳到的位置等于nums.length-1的话就说明成功了返回true，否则返回false。

首先我们想的应该是子问题，每次要怎么跳呢？

我们有个起点，起点的数值就是我们走的最远步长。那么我们可以走的就是在[position+1,position+nums[position]之间，我们用for()来遍历。注意，步长我们需要我们需要跟数列长度对比。因为有的时候你当前位置+步长是大于当前数列长度的，这样会走出去，造成错误，所以我们需要判断一样，要是大于的时候，我们就去剩下的长度作为步长。

递归的时候我们递归下一个位置就可以了。

那么最关键的问题，我们怎么回溯呢？

我们看代码

public boolean canJump(int[] nums) {

return jump(nums,0);

}

public boolean jump(int[] nums,int position){

if(position==nums.length-1)

return true;

int furthestJump = Math.min(position+nums[position],nums.length-1);

for(int nextposition = position+1;nextposition<=furthestJump;position++){

**if(jump(nums,nextposition))**

**return true;**

}

return false;

}

注意，我标红的那部分，我们开始递归的走每一步，走完一步走下一步，此时递归没有到头，是没有返回值的，当走的头的时候position还没有等于length-1，我们就继续走，循环是进不去的，应为position+1不满足条件，所以不会进入递归中，然后返回false。我们这里的每条路径的递归都是到最后有返回值的。如果走到最后成功了，就说明这个数列是满足条件的，不需要继续往下进行了，直接返回true，彻底结束了。当返回的是false，不进入终止程序的，也就是我表中的代码，程序就会继续完成当前位置所有路径的遍历，然后回到上一层次的递归，继续遍历，直到满足要求终止程序。

注意的是，递归回溯法，我们每次都是从第一路径开始，到第二个，第三个，一直到最后一个。然后最后一个执行完毕，才会到上一个，到上一个，最后到第一个。我们可以在末尾设置条件，当到当前位置满足的时候退出，就不会回退了。如果不满足条件是一定会回退的。