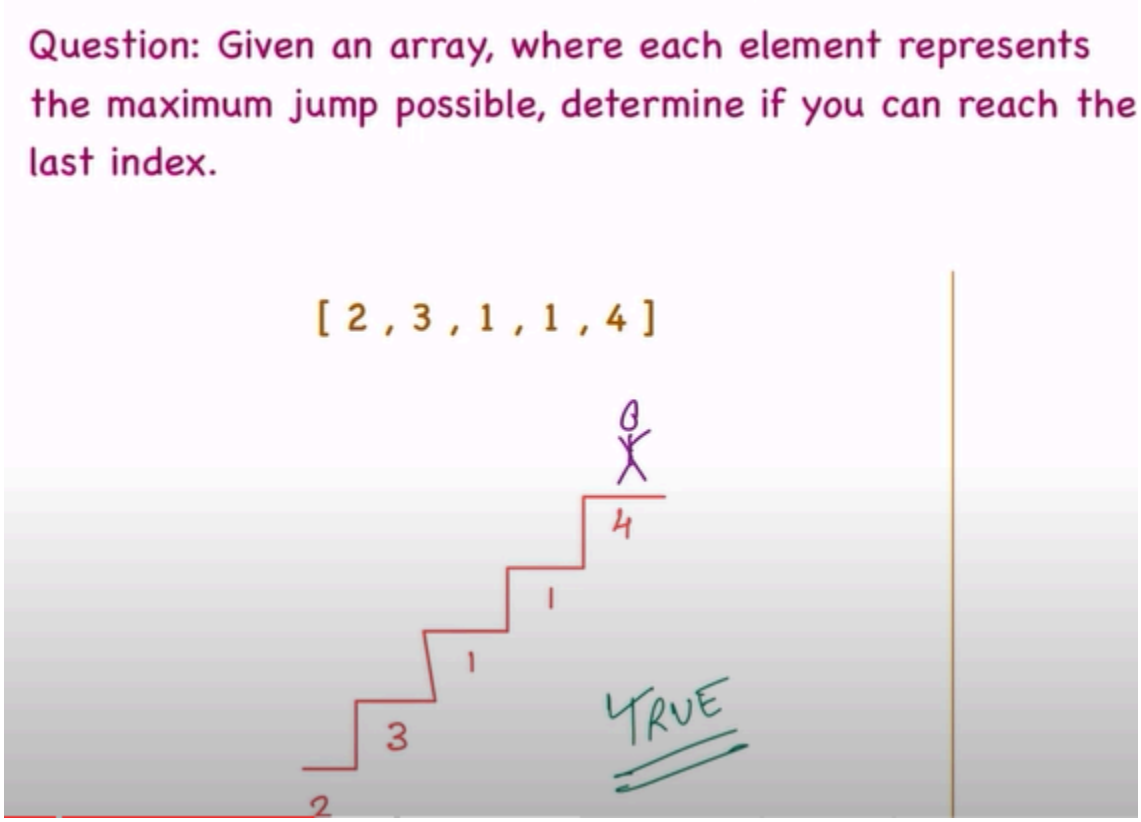
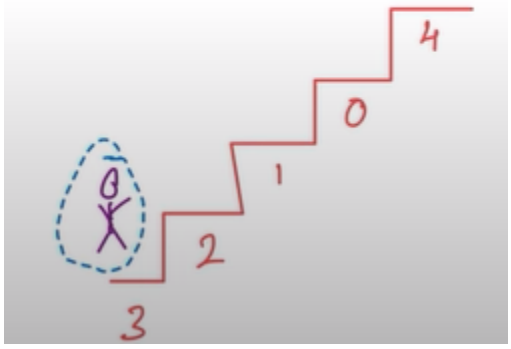
# 55. Jump Game

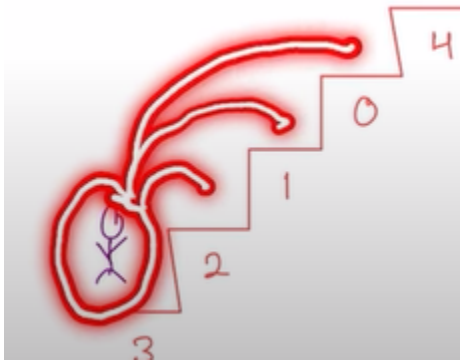
Problem:



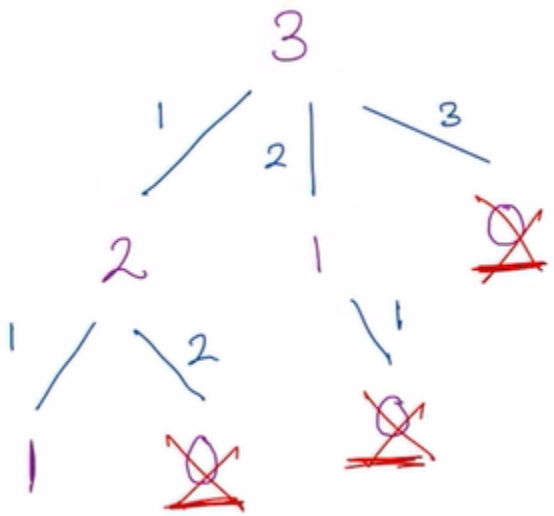




1. Brute Force Approach
2. Given array = [3,2,1,0,4]

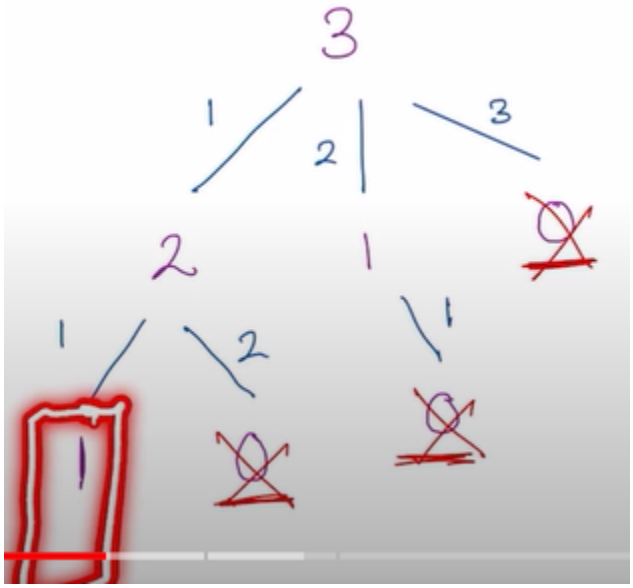

3. Here for 3…we make this jumps



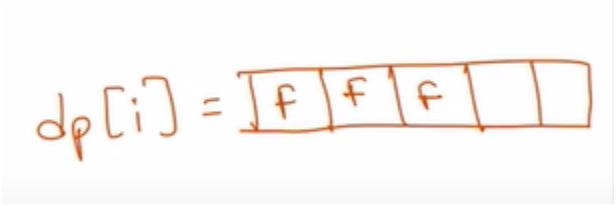

4. Here for 3 we can make 3 jumps…similarly for 2 and 1



5. Here as we have calculated jumps at position 1…so we dont need to calculate that again




to implement this we can use an array to store ..whether we can reach our goal or not




6. Here the work of finding them…would be too high..
7. Using brute force - TC O(n^m)..using DP - TC O(n^2)

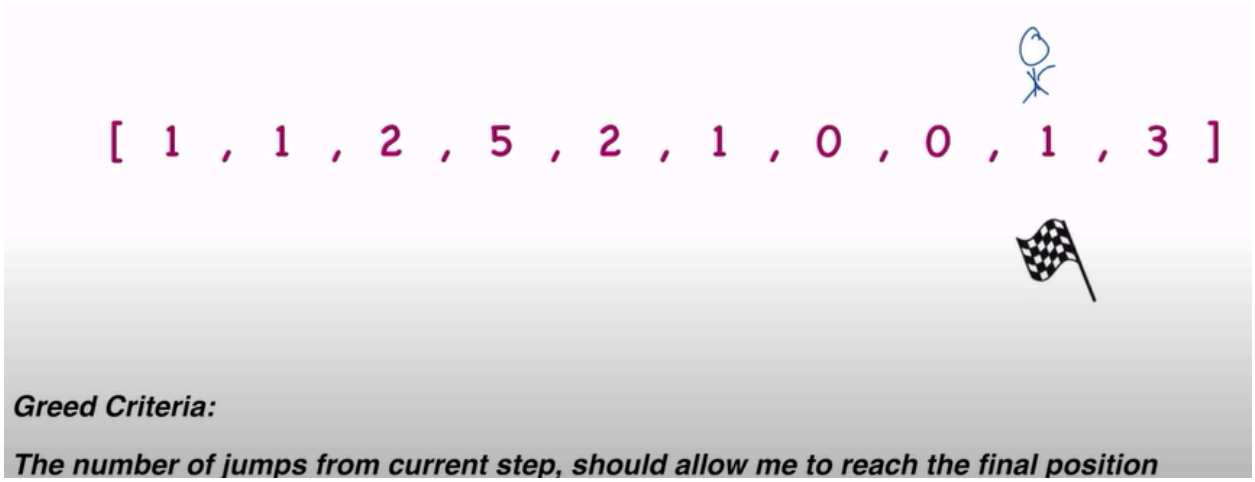
Optimal Greedy Approach - O(n)

1. Lets take this sample test case

[ 2 , 3 , 1 , 1 , 4 ]

2. In this test case…we does not have any 0
3. So here “no matter the size of the array ..if the array does not contain ‘0’ then we can reach our final goal”
4. Lets take a this test case which has ‘0’ in it

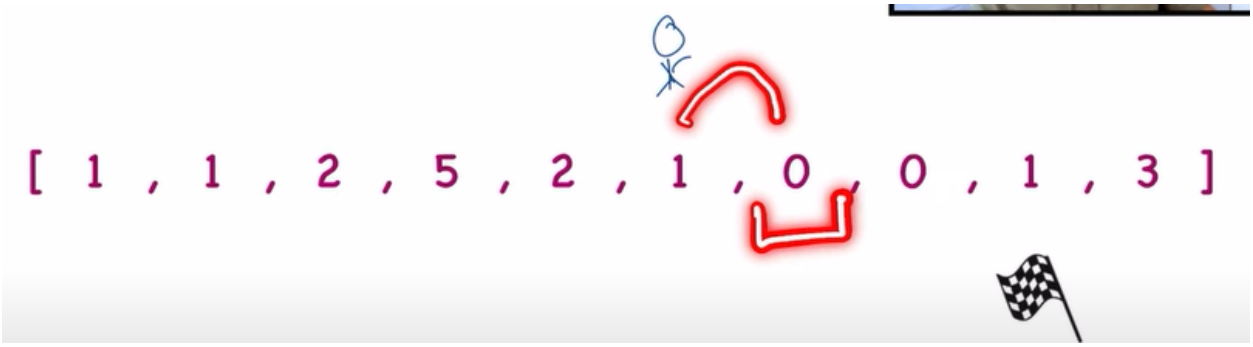
[ 1 , 1 , 2 , 5 , 2 , 1 , 0 , 0 , 1 , 3 ]




5.
6. And now if stand at 0 and we cant jump to ..and final dest remains same which is 1


7. If we go one step backward…then we again face same issue…cannot jump to 1

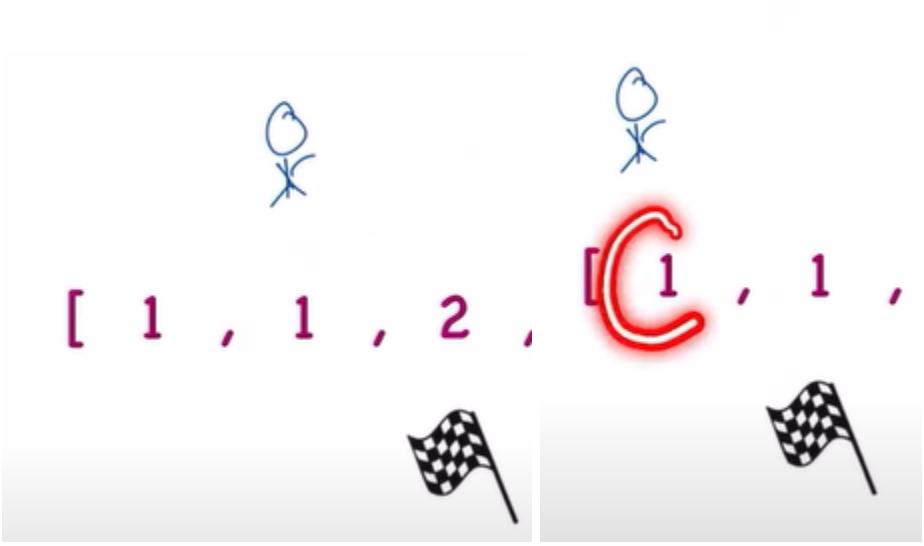
8. One step backward again here we can jump one position to 0..but cannot jump to final dest




9. SImilarly if we move backward to 2…we cant jump to final dest
10. Now from 5..we can reach our final destination (1)


11. Now we'll update our final dest to 5

12. One step backward to 2…we can reach final dest of 5…as we reach final dest..we'll update our final dest to 2 


13. Here we can reach to our goal…for this test case

Python code for Greedy

1. Code is self Explanatory

```
class Solution(object):
    def canJump(self, nums):
        #using greedy approach
        target = len(nums)-1
        for i in range(len(nums)-1,-1,-1):
            if i + nums[i] >= target:
                target = i
        return True if target == 0 else False
```