55. Jump Game

Medium 10730 615 Add to List Share

You are given an integer array `nums`. You are initially positioned at the array's **first index**, and each element in the array represents your maximum jump length at that position.

Return `true` *if you can reach the last index, or* `false` *otherwise*.

Example 1:

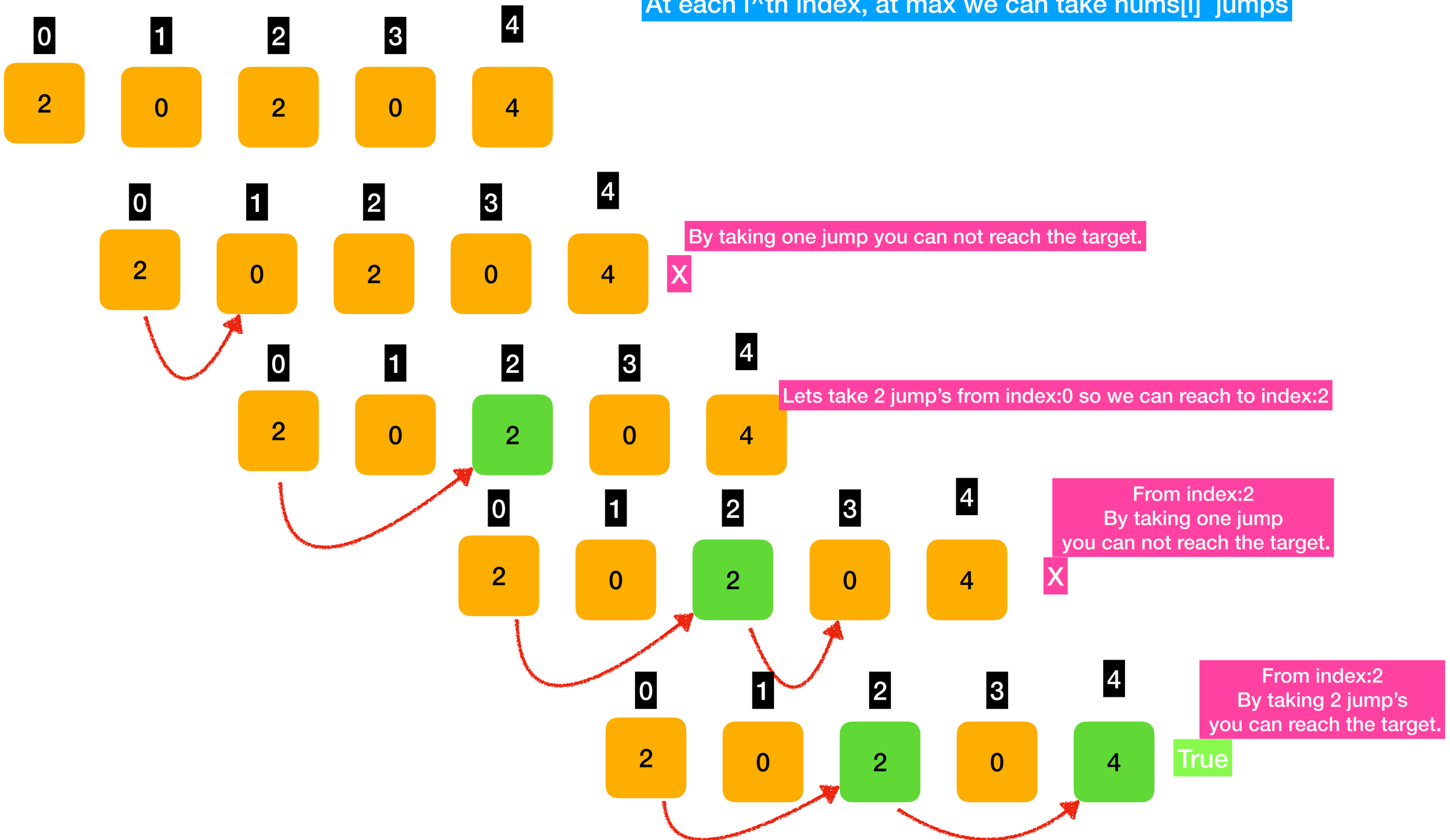
```
Input: nums = [2,3,1,1,4]
Output: true
Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.
```

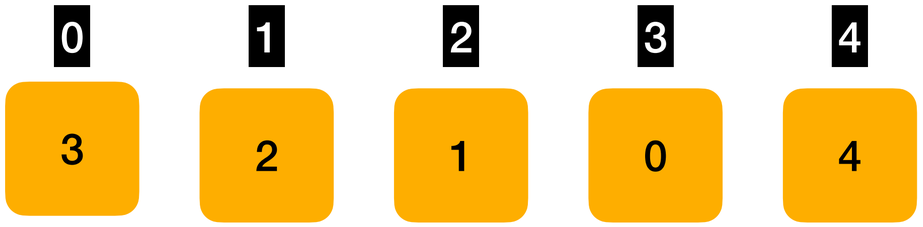
Example 2:

```
Input: nums = [3,2,1,0,4]
Output: false
Explanation: You will always arrive at index 3 no matter what. Its maximum jump length is 0, which makes it impossible to reach the last index.
```

Constraints:

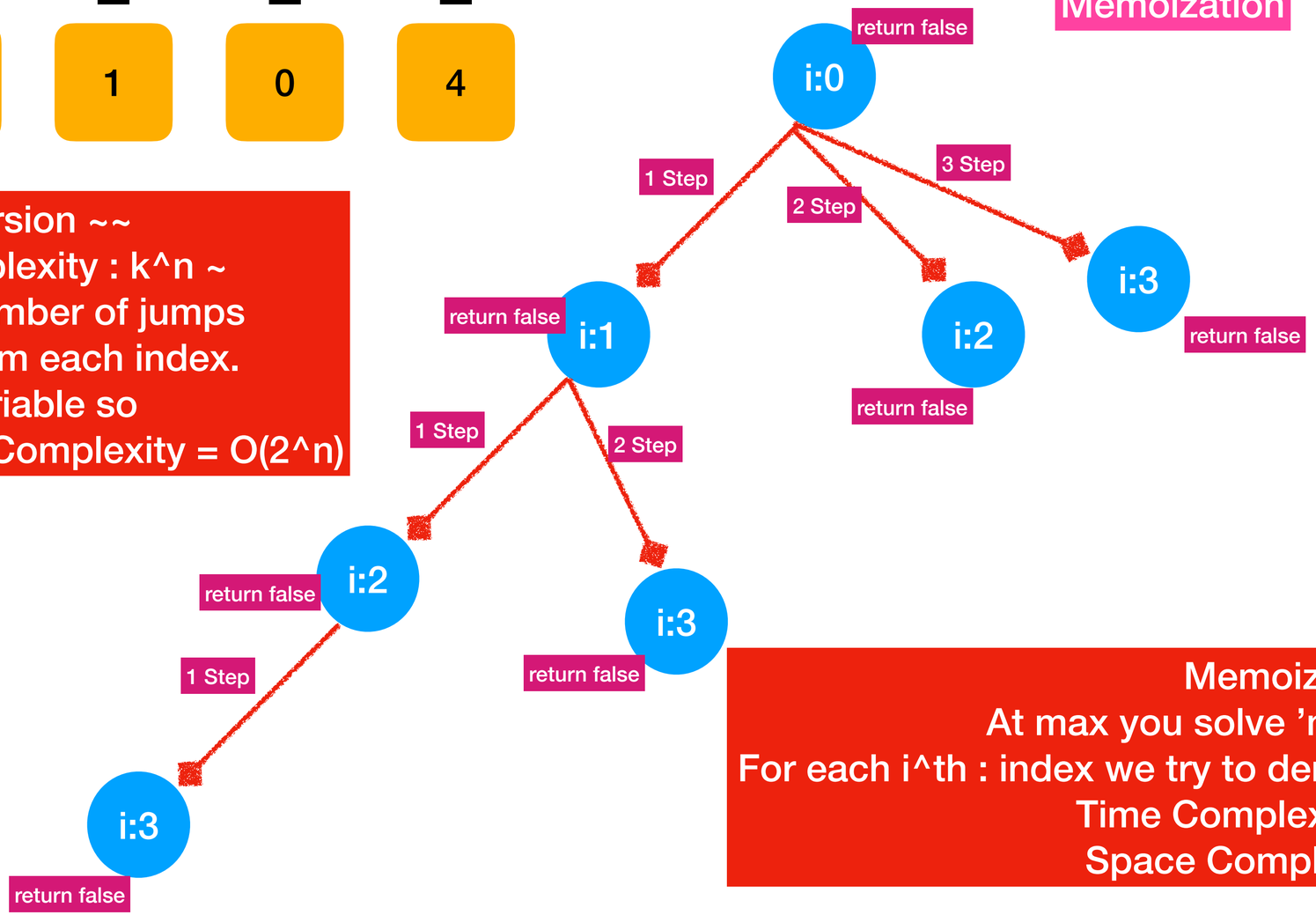
- `1 <= nums.length <= 10^4`
- `0 <= nums[i] <= 10^5`


At each i^th index, at max we can take nums[i] jumps
0 1 2 3 4
2 0 2 0 4
By taking one jump you can not reach the target.
X
Lets take 2 jump's from index:0 so we can reach to index:2
From index:2
By taking one jump
you can not reach the target.
X
From index:2
By taking 2 jump's
you can reach the target.
True

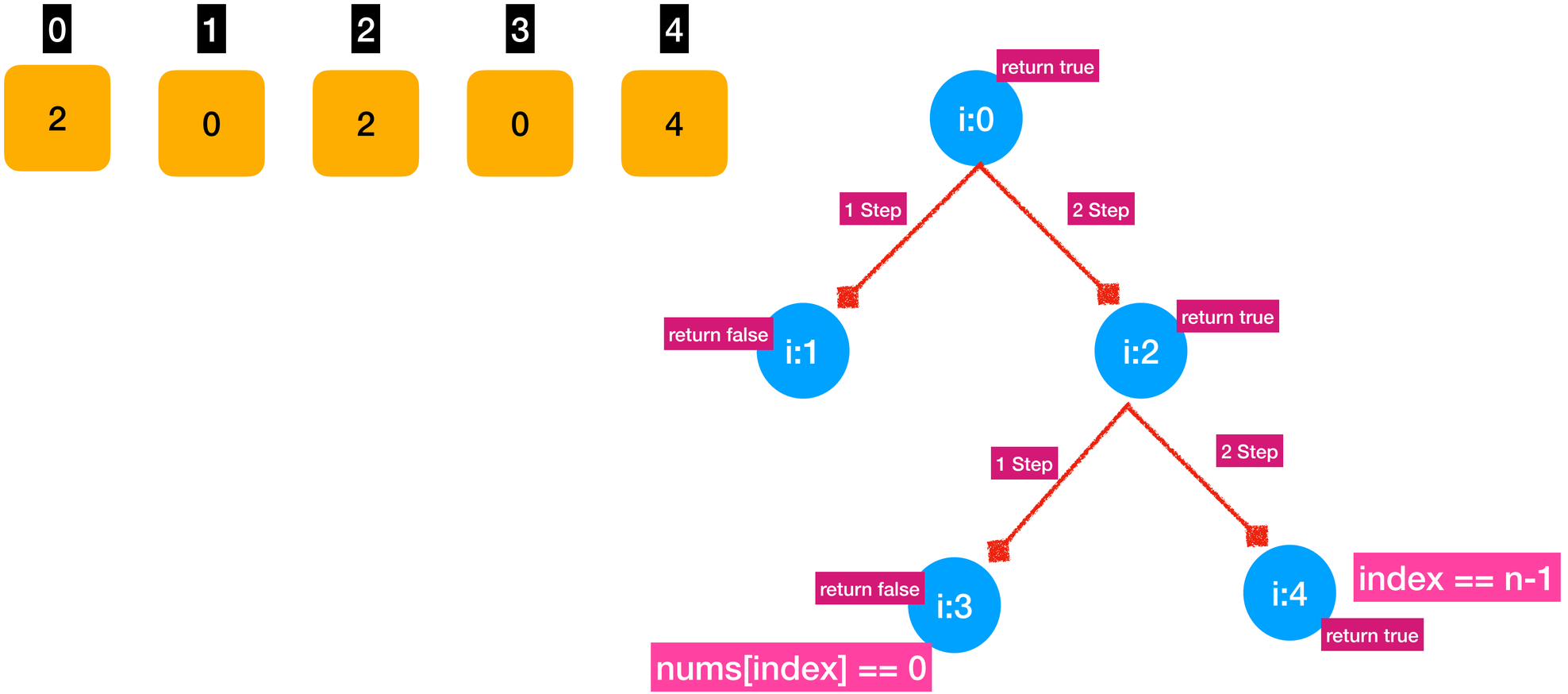


Recursion ~~
Time Complexity : k^n ~
Here k is number of jumps
we take from each index.
Its variable so
on and avg Time Complexity = O(2^n)

Memoization



Memoization
At max you solve 'n' Sub Problems.
For each i^th : index we try to derive (I+1) to (n-1) sub problems.
Time Complexity : O(n^2)
Space Complexity : O(n)

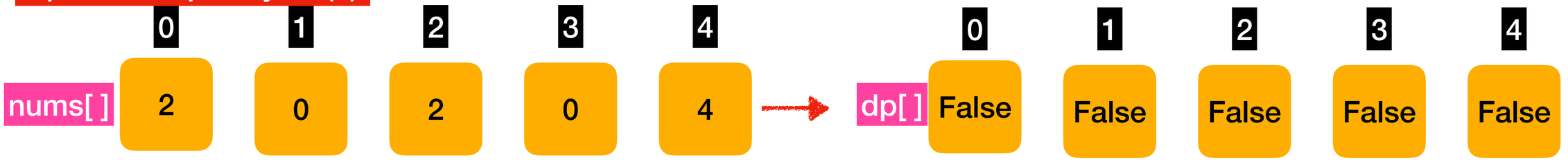


Memoization
At max you solve ’n’ Sub Problems.
For each i^th : index we try to derive (I+1) to (n-1) sub problems.
Time Complexity : O(n^2)
Space Complexity : O(n)

Tabulation
Time Complexity : O(n^2)
Space Complexity : O(n)

Tabulation

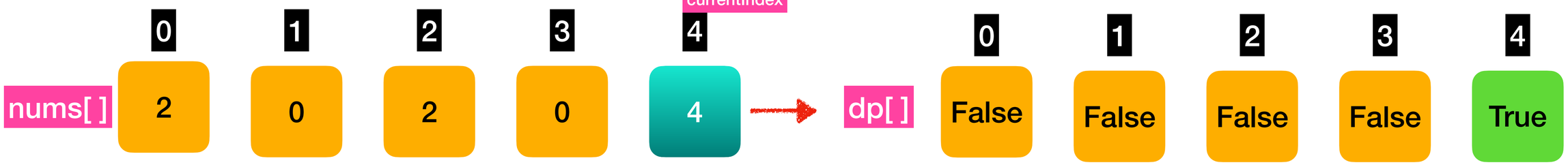
Buttom-Up Approach,
we solve the subproblem then reuse in next sub problem.

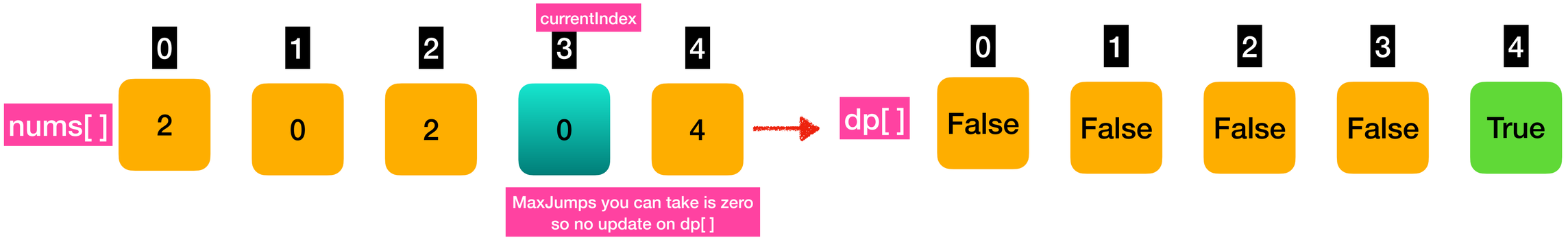


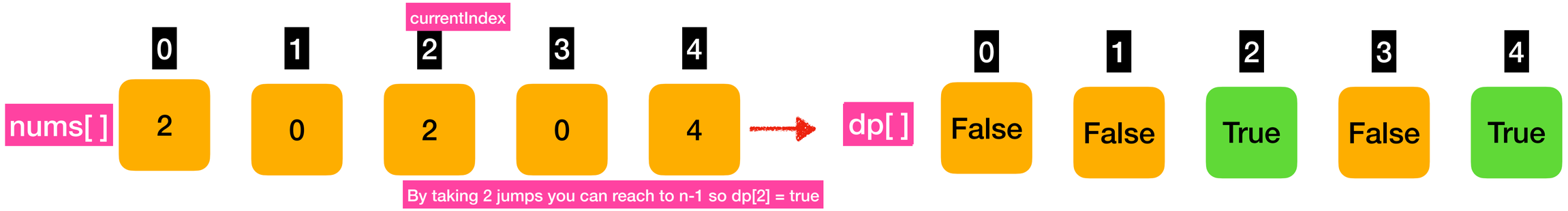
We should start from n-1 index to achieve bottom up approach

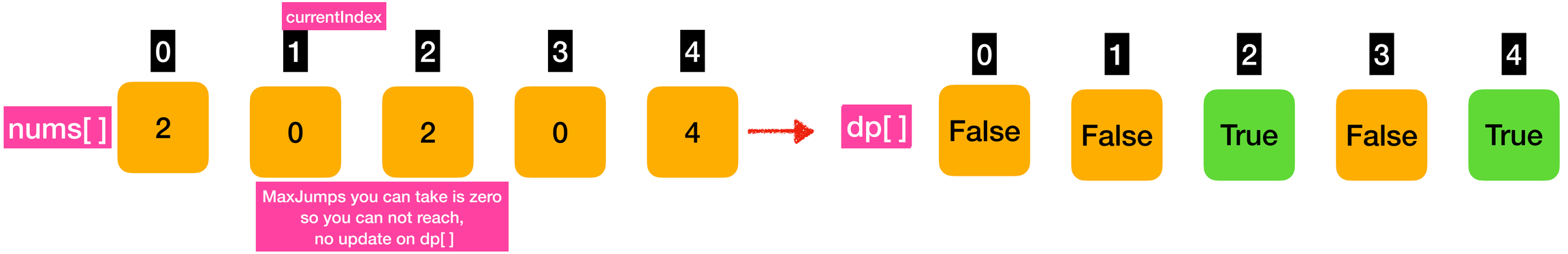
Feeding SubProblem Results

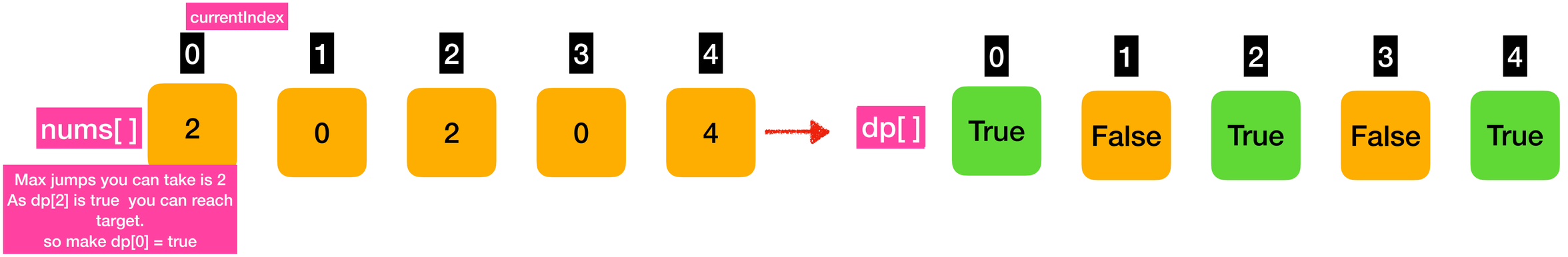
If you are at n-1 index,
it always be true.
dp[n-1] = True






currentIndex
0
1
2
3
4
nums[ ]
2
0
2
0
4
dp[ ]
False
False
True
False
True
By taking 2 jumps you can reach to n-1 so dp[2] = true


currentIndex
0
1
2
3
4
nums[ ]
2
0
2
0
4
dp[ ]
False
False
True
False
True
MaxJumps you can take is zero
so you can not reach,
no update on dp[ ]


currentIndex
0
1
2
3
4
nums[ ]
2
0
2
0
4
dp[ ]
True
False
True
False
True
Max jumps you can take is 2
As dp[2] is true you can reach
target.
so make dp[0] = true