```
  0 1 2 3 4     0 1 2 3 4.
[2. 3. 1. 1. 4] [3. 2. 1. 0. 4]
```

- iterate over nums.
  - for each num, we keep updating furthest place we can reach.
  - if f >= i , it's possible to reach i.

```
const fn = (nums) => {
    let f = 0;
    for(let i = 0; i < nums.length; i++) {
        if (f >= i) {
            f = Math.max(f, i + nums[i]);
        }
    }
    return f >= nums.length - 1;
}
```

```
  0 1 2 3 4          0 1 2 3 4
[2,3,1,1,4]   [3,2,1,0,4].
```

$dp[i] = dp[j]\ \&\&\ (j + nums[j] >= i)$

j from 0 to i-1

Accepted solution,

but there is one more optimization!

```
const fn = (nums) => {
    const n = nums.length;
    const dp = new Array(n).fill(false);
    dp[0] = true;
    for(let i = 1; i < n; i++) {
        dp[i] = (() => {
            for(let j = 0; j < i; j++) {
                if(dp[j] && (j + nums[j] >= i)) {
                    return true;
                }
            }
        })
    }
    return dp[n-1];
}
```