

12
.2

Jump Game I

MEDIUM

■ What You Will Learn

- ✓ How to think about reachability — not paths
- ✓ The 'maxReach' greedy approach: track how far you can get
- ✓ Why you don't need to explore every path ($O(n)$ instead of exponential)
- ✓ The key insight: if you can reach index i , you can try all its jumps

■ Intuition First

You're standing on index 0 of an array. Each value is how far you can jump from that spot. Can you reach the last index?

Instead of exploring every possible path (which explodes exponentially), track a single variable: `maxReach` — the farthest index you can currently get to. As you walk forward, update `maxReach`. If at any point your current position exceeds `maxReach`, you're stuck — return false.

■ The Greedy Insight

- `maxReach` = farthest index reachable from any index up to i
- At each index i : `maxReach = max(maxReach, i + nums[i])`
- If $i > \text{maxReach}$ at any point: you can't get here — return false
- If `maxReach >= last index`: return true

■ Visual Explanation

Array: [2, 3, 1, 1, 4] → Can we reach index 4?

Index: 0 1 2 3 4

Value: [2] [3] [1] [1] [4]

Reach: ↔ (from 0, can reach up to $0+2=2$)

■■■■■■■■■ (from 1, can reach up to $1+3=4$)

→ `maxReach` becomes 4 = last index ✓

Failing example: [3, 2, 1, 0, 4] → Can we reach index 4?

Index: 0 1 2 3 4

Value: [3] [2] [1] [0] [4]

$i=0$: `max=3` $i=1$: `max=3` $i=2$: `max=3` $i=3$: $3+0=3$

$i=4$: $4 > \text{maxReach}=3$ → STUCK! return false ✗

■ Dry Run — [2, 3, 1, 1, 4]

`nums[]` — start at index 0

2	3	1	1	4
---	---	---	---	---

Step 1 $i=0$: $\text{maxReach} = \max(0, 0+2) = 2$. $0 \leq 2 \rightarrow \text{ok}$

2	3	1	1	4
---	---	---	---	---

Step 2 $i=1$: $\text{maxReach} = \max(2, 1+3) = 4$. $1 \leq 4 \rightarrow \text{ok}$

2	3	1	1	4
---	---	---	---	---

Step 3 $i=2$: $\text{maxReach} = \max(4, 2+1) = 4$. $2 \leq 4 \rightarrow \text{ok}$

2	3	1	1	4
---	---	---	---	---

Step 4 $i=3$: $\text{maxReach} = \max(4, 3+1) = 4$. $3 \leq 4 \rightarrow \text{ok}$

2	3	1	1	4
---	---	---	---	---

Step 5 $i=4$: $\text{maxReach}=4$, $i=4 \rightarrow$ reached last index! return TRUE ✓

2	3	1	1	4
---	---	---	---	---

■ Java Code

```
public static boolean canJump(int[] nums) {
    int maxReach = 0; // Farthest index reachable so far
    for (int i = 0; i < nums.length; i++) {
        if (i > maxReach) {
            return false; // Can't even get to index i
        }
        // Update farthest reachable from here
        maxReach = Math.max(maxReach, i + nums[i]);
        // Early exit: already reached the end
        if (maxReach >= nums.length - 1) {
            return true;
        }
    }
    return true;
}
```

■ Time & Space Complexity

Case	Time	Space	What it really means
All Cases	$O(n)$	$O(1)$	Single pass — just one variable maxReach

■ Why Greedy, Not DP?

- DP approach: $dp[i] = \text{true}$ if reachable from 0. Fill for each index. Still $O(n)$ but $O(n)$ space
- Greedy: we don't care WHICH path — just CAN we reach? One number (maxReach) captures everything
- Greedy is smarter here — $O(1)$ space and same time

■ Common Mistakes

- Checking $\text{maxReach} \geq \text{nums.length}$ (off-by-one) — should be $\text{nums.length}-1$
- Updating maxReach before checking $i > \text{maxReach}$ — check FIRST, then update
- Using $i \leq \text{maxReach}$ as loop condition — misses the $i == \text{maxReach}$ case

■ Pro Tips

- If $\text{nums}[i]=0$ everywhere except first: stuck immediately after index 0
- This problem is the foundation for Jump Game II (minimum jumps)
- Think of maxReach as a 'coverage window' — you're asking: does this window reach the end?

■ Visual Thinking

Imagine a rope laid flat on the ground. Each index is a peg.

Standing on peg i , you can reach pegs up to $i + \text{nums}[i]$.

maxReach is like a rubber band stretched to the farthest peg you've touched.

Walk left to right — the rubber band only stretches, never shrinks.

If you walk past the rubber band's end, you've walked off a cliff.

Animation: rubber band stretching right as you hop forward, snapping red if index passes its end.

■ Practice Problems

[Easy] Can you reach last index in [2,3,0,4]? Trace manually

[Med] Jump Game I with `nums=[0,1]` — what happens?

[Med] Jump Game II (minimum jumps — see next chapter)

[Hard] Jump Game with variable costs: minimize total cost to reach end

■ Quick Revision Box

- Track `maxReach` = farthest index reachable so far
- At `i`: if `i > maxReach` → stuck, return false
- Update: `maxReach = max(maxReach, i + nums[i])`
- If `maxReach >= n-1` at any point → return true
- $O(n)$ time, $O(1)$ space — no DP table needed