

6899. Maximum Number of Jumps to Reach the Last Index

[My Submissions \(/contest/weekly-contest-353/problems/maximum-number-of-jumps-to-reach-the-last-index/submissions/\)](#)
[Back to Contest \(/contest/weekly-contest-353/\)](#)

You are given a **0-indexed** array `nums` of `n` integers and an integer `target`.

You are initially positioned at index `0`. In one step, you can jump from index `i` to any index `j` such that:

- $0 \leq i < j < n$
- $-target \leq nums[j] - nums[i] \leq target$

Return the **maximum number of jumps** you can make to reach index `n - 1`.

If there is no way to reach index `n - 1`, return `-1`.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Example 1:

Input: `nums = [1,3,6,4,1,2]`, `target = 2`

Output: 3

Explanation: To go from index `0` to index `n - 1` with the maximum number of jumps, you can perform the following jumping sequence

- Jump from index `0` to index `1`.
- Jump from index `1` to index `3`.
- Jump from index `3` to index `5`.

It can be proven that there is no other jumping sequence that goes from `0` to `n - 1` with more than 3 jumps. Hence, the answer is

Example 2:

Input: `nums = [1,3,6,4,1,2]`, `target = 3`

Output: 5

Explanation: To go from index `0` to index `n - 1` with the maximum number of jumps, you can perform the following jumping sequence

- Jump from index `0` to index `1`.
- Jump from index `1` to index `2`.
- Jump from index `2` to index `3`.
- Jump from index `3` to index `4`.
- Jump from index `4` to index `5`.

It can be proven that there is no other jumping sequence that goes from `0` to `n - 1` with more than 5 jumps. Hence, the answer is

Example 3:

Input: `nums = [1,3,6,4,1,2]`, `target = 0`

Output: -1

Explanation: It can be proven that there is no jumping sequence that goes from `0` to `n - 1`. Hence, the answer is `-1`.

Constraints:

- $2 \leq \text{nums.length} == n \leq 1000$
- $-10^9 \leq \text{nums}[i] \leq 10^9$
- $0 \leq \text{target} \leq 2 * 10^9$

JavaScript



```

1 function Deque() {
2   let m = {}, first = 0, last = -1;
3   return { unshift, shift, push, pop, front, back, size, show }
4   function push(...args) {
5     let i = 0;
6     if (size() == 0) {
7       first = last = 0;
8       m[first] = args[i++];
9     }
10    for (; i < args.length; i++) m[++last] = args[i];

```

```

11     }
12     function unshift(...args) {
13         let i = 0;
14         if (size() == 0) {
15             first = last = 0;
16             m[first] = args[i++];
17         }
18         for (; i < args.length; i++) m[--first] = args[i];
19     }
20     function pop() {
21         let res = m[last];
22         delete m[last];
23         last--;
24         return res;
25     }
26     function shift() {
27         let res = m[first];
28         delete m[first];
29         first++;
30         return res;
31     }
32     function front() {
33         return m[first];
34     }
35     function back() {
36         return m[last];
37     }
38     function size() {
39         if (first > last) return 0;
40         return last - first + 1;
41     }
42     function show() {
43         let a = Object.keys(m), res = [];
44         a.sort((x, y) => x - y);
45         for (const k of a) res.push(m[k]);
46         return res;
47     }
48 }
49
50 const maximumJumps = (a, t) => {
51     let n = a.length, g = new Map();
52     for (let i = 0; i < n; i++) {
53         for (let j = i + 1; j < n; j++) {
54             if (Math.abs(a[i] - a[j]) <= t) {
55                 if (!g.has(i)) g.set(i, new Set());
56                 g.get(i).add(j);
57             }
58         }
59     }
60     let q = new Deque(), dis = Array(n).fill(Number.MIN_SAFE_INTEGER)
61     dis[0] = 0;
62     q.push(0);
63     while (q.size()) {
64         let cur = q.shift();
65         let d = g.get(cur) || [];
66         for (const next of d) {
67             if (next > cur) {
68                 if (dis[next] < dis[cur] + 1) {
69                     dis[next] = dis[cur] + 1;
70                     q.push(next);
71                 }
72             }
73         }
74     }
75     return dis[n - 1] == Number.MIN_SAFE_INTEGER ? -1 : dis[n - 1];
76 };

```

☐ Custom Testcase☒ Use Example Testcases

Run

Submit

Submission Result: **Accepted** (/submissions/detail/989802301/) ?

More Details > (/submissions/detail/989802301/)

Share your acceptance!