



## 2571. Minimum Operations to Reduce an Integer to 0

order-traversal/)

[My Submissions \(/contest/weekly-contest-333/problems/minimum-operations-to-reduce-an-integer-to-0/submissions/\)](#)
[Back to Contest \(/contest/weekly-contest-333/\)](#)

You are given a positive integer  $n$ , you can do the following operation **any** number of times:

- Add or subtract a **power** of 2 from  $n$ .

Return the **minimum** number of operations to make  $n$  equal to 0.

A number  $x$  is power of 2 if  $x == 2^i$  where  $i \geq 0$ .

## Example 1:

**Input:**  $n = 39$ **Output:** 3**Explanation:** We can do the following operations:

- Add  $2^0 = 1$  to  $n$ , so now  $n = 40$ .
- Subtract  $2^3 = 8$  from  $n$ , so now  $n = 32$ .
- Subtract  $2^5 = 32$  from  $n$ , so now  $n = 0$ .

It can be shown that 3 is the minimum number of operations we need to make  $n$  equal to 0.

## Example 2:

**Input:**  $n = 54$ **Output:** 3**Explanation:** We can do the following operations:

- Add  $2^1 = 2$  to  $n$ , so now  $n = 56$ .
- Add  $2^3 = 8$  to  $n$ , so now  $n = 64$ .
- Subtract  $2^6 = 64$  from  $n$ , so now  $n = 0$ .

So the minimum number of operations is 3.

## Constraints:

- $1 \leq n \leq 10^5$

[Discuss \(https://leetcode.com/problems/minimum-operations-to-reduce-an-integer-to-0/discuss/\)](https://leetcode.com/problems/minimum-operations-to-reduce-an-integer-to-0/discuss/)

JavaScript



```

1 const cutMaxConsecutive = (as) => { let d = [], l = 0, n = as.length; for (let i = 0; i + 1 < n; i++) { if (as[i + 1]
2   != as[i]) { d.push(as.slice(l, i + 1)); l = i + 1; } } d.push(as.slice(l)); return d; };
3
4 function Bisect() {
5   return { insert_right, insert_left, bisect_left, bisect_right }
6   function insert_right(a, x, lo = 0, hi = null) {
7     lo = bisect_right(a, x, lo, hi);
8     a.splice(lo, 0, x);
9   }
10  function bisect_right(a, x, lo = 0, hi = null) { // > upper_bound
11    if (lo < 0) throw new Error('lo must be non-negative');
12    if (hi == null) hi = a.length;
13    while (lo < hi) {
14      let mid = parseInt((lo + hi) / 2);
15      a[mid] > x ? hi = mid : lo = mid + 1;
16    }
17    return lo;
18  }
19  function insert_left(a, x, lo = 0, hi = null) {
20    lo = bisect_left(a, x, lo, hi);
21    a.splice(lo, 0, x);
22  }
23  function bisect_left(a, x, lo = 0, hi = null) { // >= lower_bound

```

```

23     if (lo < 0) throw new Error('lo must be non-negative');
24     if (hi == null) hi = a.length;
25     while (lo < hi) {
26         let mid = parseInt((lo + hi) / 2);
27         a[mid] < x ? lo = mid + 1 : hi = mid;
28     }
29     return lo;
30 }
31 }
32
33 function TreeSet(elements) {
34     let ts = [], se = new Set(), bisect = new Bisect();
35     initialize();
36     return { add, first, last, poll, pollLast, floor, ceiling, lower, higher, remove, has, size, clear, show };
37     function initialize() {
38         if (elements) {
39             for (const e of elements) {
40                 if (!se.has(e)) {
41                     bisect.insort_right(ts, e);
42                     se.add(e);
43                 }
44             }
45         }
46     }
47     function add(e) {
48         if (!se.has(e)) {
49             bisect.insort_right(ts, e);
50             se.add(e);
51         }
52     }
53     function first() {
54         return ts[0];
55     }
56     function last() {
57         return ts[ts.length - 1];
58     }
59     function poll() {
60         let res = ts[0];
61         ts.splice(0, 1);
62         se.delete(res);
63         return res;
64     }
65     function pollLast() {
66         let res = ts.pop();
67         se.delete(res);
68         return res;
69     }
70     function ceiling(e) { // >= lower_bound
71         let idx = bisect.bisect_right(ts, e);
72         let res = ts[idx - 1] == e ? e : ts[bisect.bisect_right(ts, e)];
73         return res == undefined ? null : res;
74     }
75     function higher(e) { // > upper_bound
76         let idx = bisect.bisect_right(ts, e);
77         let res = ts[idx] > e ? ts[idx] : ts[bisect.bisect_right(ts, e) + 1];
78         return res == undefined ? null : res;
79     }
80     function floor(e) { // <=
81         let idx = bisect.bisect_left(ts, e);
82         let res = ts[idx] == e ? e : ts[bisect.bisect_left(ts, e) - 1];
83         return res == undefined ? null : res;
84     }
85     function lower(e) { // <
86         let idx = bisect.bisect_left(ts, e);
87         let res = ts[idx] < e ? ts[idx] : ts[bisect.bisect_left(ts, e) - 1];
88         return res == undefined ? null : res;
89     }
90     function remove(e) {
91         let idx = bisect.bisect_left(ts, e);
92         if (ts[idx] == e) ts.splice(idx, 1);
93         se.delete(e);
94     }
95     function has(e) {
96         return se.has(e);
97     }
98     function size() {

```

```

99     return ts.length;
100 }
101 function clear() {
102     ts = [];
103     se.clear();
104 }
105 function show() {
106     return ts;
107 }
108 }
109
110 const minOperations = (N) => {
111     let s = N.toString(2), n = s.length, ts = new TreeSet(), cnt = 0, d = cutMaxConsecutive(s), res1 = 0;
112     for (const e of d) {
113         if (e[0] == '1') {
114             res1 += e.length == 1 ? 1 : 2;
115         }
116     }
117     for (let i = 0; i < n; i++) {
118         if (s[i] == '1') ts.add(i);
119     }
120     // pr(s, ts.show());
121     // for (let t = 1; t <= 10; t++) {
122     while (ts.size() > 1) {
123         // pr(ts.show(), "cnt", cnt);
124         while (1) {
125             let cur = ts.last(), pre = ts.lower(cur);
126             // pr("cur", cur, "pre", pre, ts.show());
127             ts.remove(cur);
128             if (cur - 1 == pre) {
129             } else { // 7 4
130                 let last = cur - 1;
131                 if (last - 1 == pre) {
132                     ts.add(last);
133                 } else {
134                     cnt++; // single
135                 }
136                 break;
137             }
138         }
139         cnt++; // merge
140     }
141     let res2 = cnt + ts.size();
142     // pr("222", ts.show(), "cnt", cnt, "res2", res2, "res1", res1);
143     return Math.min(res1, res2);
144 };

```

☐ Custom Testcase☒ Use Example Testcases

Run

Submit

Submission Result: **Accepted** (/submissions/detail/900812220/) ?

More Details &gt; (/submissions/detail/900812220/)

Share your acceptance!

Copyright © 2023 LeetCode

[Help Center \(/support\)](#) | [Jobs \(/jobs\)](#) | [Bug Bounty \(/bugbounty\)](#) | [Online Interview \(/interview/\)](#) | [Students \(/student\)](#) | [Terms \(/terms\)](#) | [Privacy Policy \(/privacy\)](#) [United States \(/region\)](#)