

330. Patching Array

Dec. 11, 2016 | 5.3K views

PreviousNext
★★★★★
Average Rating: 4.76 (33 votes)

Given a sorted positive integer array *nums* and an integer *n*, add/patch elements to the array such that any number in range `[1, n]` inclusive can be formed by the sum of some elements in the array. Return the minimum number of patches required.

Example 1:

Input: *nums* = [1,3], *n* = 6
Output: 1
Explanation:
Combinations of *nums* are [1], [3], [1,3], which form possible sums of: 1, 3, 4.
Now if we add/patch 2 to *nums*, the combinations are: [1], [2], [3], [1,3], [2,3], [1,2,3].
Possible sums are 1, 2, 3, 4, 5, 6, which now covers the range [1, 6].
So we only need 1 patch.

Example 2:

Input: *nums* = [1,5,10], *n* = 20
Output: 2
Explanation: The two patches can be [2, 4].

Example 3:

Input: *nums* = [1,2,2], *n* = 5
Output: 0

Solution

Intuition

For any missing number, if we want to cover it, we have to add at least one number smaller or equal to that number. Otherwise, it will not be covered. Imagine you want to give someone change of *x* cents, and you don't have enough coins. You will need to ask for coin less than or equal to *x*.

Algorithm

Suppose *miss* is the smallest missing number, then we know that `[1, miss)` (left-closed, right-open) is already covered. In order to cover *miss*, we have to add something smaller than or equal to *miss*. Otherwise, there is no way we can cover it.

For example, you have any array *nums* = [1,2,3,8] and *n* = 16. The numbers already covered is in the ranges `[1, 6]` and `[8, 14]`. In other words, 7, 15, 16 are missing. If you add patches larger than 7, then 7 is still missing.

Suppose the number we added is *x* then, the ranges `[1, miss)` and `[x, x + miss)` are both covered. And since we know that *x* <= *miss*, the two ranges will cover the range `[1, x + miss)`. We want to choose *x* as large as possible so that the range can cover as large as possible. Therefore, the best option is *x* = *miss*.

After we covered *miss*, we can recalculate the coverage and see what's the new smallest missing number. We then patch that number. We do this repeatedly until no missing number.

Here is the recipe of this greedy algorithm:

- Initialize the range `[1, miss) = [1, 1) = empty`
- While *n* is not covered yet
- if the current element *nums*[*i*] is less than or equal to *miss*
 - extends the range to `[1, miss + nums[i])`
 - increase *i* by 1
- otherwise
 - patch the array with *miss*, extends the range to `[1, miss + miss)`
 - increase the number of patches
- Return the number of patches

Example:

nums = [1,2,3,8] and *n* = 80

iteration	miss	covered range	i	nums[i]	patches	comment
0	1	[1, 1)	0	1	0	
1	2	[1, 2)	1	2	0	
2	4	[1, 4)	2	3	0	
3	7	[1, 7)	3	8	0	
4	14	[1, 14)	3	8	1	patch 7
5	22	[1, 22)	4	none	1	
6	44	[1, 44)	4	none	2	patch 22
7	88	[1, 88)	4	none	3	patch 44

Correctness

One may ask how do you know this is correct? In this section, we will prove that the greedy algorithm always gives the smallest possible number of patches:

Lemma

If the greedy algorithm needs to patch *k* numbers to cover `[1, n]`, it is impossible to patch less than *k* numbers to do the same.

Proof by contradiction

For a given number *n* and array *nums*, suppose the *k* patches found by greedy algorithm is $X_1 < X_2 < \dots < X_k \leq n$. If there is another set of patches $Y_1 \leq Y_2 \leq \dots \leq Y_{k'} \leq n$, with $k' < k$, then we have $Y_1 \leq X_1$, otherwise X_1 is not covered. And since adding X_1 cannot cover X_2 which means the sum of all the elements including X_1 is smaller than X_2 . Thus adding Y_1 will also not cover X_2 . And so we have:

$$Y_2 \leq X_2$$

otherwise X_2 will not be covered and so on so forth.

$$Y_i \leq X_i, i = 1, 2, \dots k'$$

Finally, we can see that since $X_1, X_2, \dots X_{k'}$ is not enough to cover X_k , therefore $Y_1, Y_2, \dots, Y_{k'}$ is also not enough to cover $X_k \leq n$. This contradicts the fact that $Y_1 \leq Y_2 \leq \dots \leq Y_{k'} \leq n$ covers `[1, n]`. **This completes the proof.**

Thus, the greedy algorithm will always return the fewest patches possible. Even through for particular cases, there could be many different ways to patch. But none of them will have fewer patches than the greedy algorithm does.

JavaCopy

```
1 public class Solution {
2     public int minPatches(int[] nums, int n) {
3         int patches = 0, i = 0;
4         long miss = 1; // use long to avoid integer overflow error
5         while (miss <= n) {
6             if (i < nums.length && nums[i] <= miss) // miss is covered
7                 miss += nums[i++];
8             else { // patch miss to the array
9                 miss += miss;
10                patches++; // increase the answer
11            }
12        }
13        return patches;
14    }
15 }
```

Complexity Analysis


- Time complexity: $O(m + \log n)$. In each iteration, we either increase the index *i* or we double the variable *miss*. The total number of increment for index *i* is *m* and the total number of doubling *miss* is $\log n$
- Space complexity: $O(1)$. We only need three variables, *patches*, *i* and *miss*.

Rate this article: ★★★★★



PreviousNext


Comments: 4

Sort By ▾







Type comment here... (Markdown is supported)


 Preview  Post



ywen1995 ★395 · May 17, 2019 10:03 AM

One of the best and most complete solutions on leetcode, deserves the 5.00 rating :D

7   |  Share |  Reply







coder42 ★6 · September 1, 2017 2:36 AM


Can anyone help to explain more about this statement "suppose the k patches found by greedy algorithm is $X_1 < X_2 < \dots \leq X_k \leq n$. If there is another set of patches $Y_1 \leq Y_2 \leq \dots \leq Y_{k'} \leq n$, with $k' < k$, then we have $Y_1 \leq X_1$, otherwise X_1 is not covered."

More specifically, how to deduce inequality of " $Y_1 \leq X_1$ " ? Given *n*, *k'* patches, we could only know $X_k <= n$ and $Y_{k'} <= n$ and $k' < k$, but I can't see where we could get " $Y_1 \leq X_1$ ".

[Read More](#)

0   |  Share |  Reply

[SHOW 1 REPLY](#)








coder42 ★6 · September 1, 2017 2:36 AM

Can anyone help to explain more about this statement "suppose the k patches found by greedy algorithm is $X_1 < X_2 < \dots \leq X_k \leq n$. If there is another set of patches $Y_1 \leq Y_2 \leq \dots \leq Y_{k'} \leq n$, with $k' < k$, then we have $Y_1 \leq X_1$, otherwise X_1 is not covered."

More specifically, how to deduce inequality of " $Y_1 \leq X_1$ " ? Given *n*, *k'* patches, we could only know $X_k <= n$ and $Y_{k'} <= n$ and $k' < k$, but I can't see where we could get " $Y_1 \leq X_1$ ".





[Read More](#)

0   |  Share |  Reply



rinass ★-1 · September 13, 2017 10:29 PM

I think one more logic would be excellent:
that any number could be formed by sum of powers of 2. This implies if we just check if the *nums* array contains powers of 2, or are derivable from the given numbers. It would be linear time.

-1   |  Share |  Reply