Leetcode-Question-330: Patching Array

原文链接：
https://leetcode.com/problems/patching-array/

题目描述：
330. Patching Array
Difficulty: Medium

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:
nums = [1, 3], n = 6
Return 1.

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:
nums = [1, 5, 10], n = 20
Return 2.
The two patches can be [2, 4].

Example 3:
nums = [1, 2, 2], n = 5
Return 0.

解析：参考https://leetcode.com/discuss/83272/share-my-thinking-proces

> *The question asked for the "minimum number of patches required". In other words, it asked for an optimal solution. Lots of problems involving optimal solution can be solved by dynamic programming and/or greedy algorithm. I started with greedy algorithm which is conceptually easy to design. Typically, a greedy algorithm needs selection of best moves for a subproblem. So what is our best move?*
>
> *Think about this example: nums = [1, 2, 3, 9]. We naturally want to iterate through nums from left to right and see what we would discover. After we encountered 1, we know 1...1 is patched completely. After encountered 2, we know 1...3 (1+2) is patched completely. After we encountered 3, we know 1...6 (1+2+3) is patched completely. After we encountered 9, the smallest number we can get is 9. So we must patch a new number here so that we don't miss 7, 8. To have 7, the numbers we can patch is 1, 2, 3 ... 7. Any number greater than 7 won't help here. Patching 8 will not help you get 7. So we have 7 numbers (1...7) to choose from. I hope you can see number 7 works best here because if we chose number 7, we can move all the way*

> *up to 1+2+3+7 = 13. (1...13 is patched completely) and it makes us reach n as quickly as possible. After we patched 7 and reach 13, we can consider last element 9 in nums. Having 9 makes us reach 13+9 = 22, which means 1...22 is completely patched. If we still did't reach n, we can then patch 23, which makes 1...45 (22+23) completely patched. We continue until we reach n.*

```
class Solution {
public:
    int minPatches(vector<int>& nums, int n) {
        int length = nums.size();
        unsigned sum = 0, i = 0, patchCnt = 0;
        while (sum < n) {
            if (i >= length || sum + 1 < nums[i]) {
                sum += sum + 1;
                patchCnt++;
            } else {
                sum += nums[i++];
            }
        }
        return patchCnt;
    }
};
```

目前投票最高的答案：

```
class Solution {
public:
    int minPatches(vector<int>& nums, int n) {
        int length = nums.size();

        int patchCnt = 0;
        unsigned missInt = 1;

        int i = 0;
        while (missInt <= n) {
            if (i < length && nums[i] <= missInt) {
                missInt += nums[i++];
            } else {
                missInt += missInt;
                patchCnt++;
            }
        }
        return patchCnt;
    }
};
```

解释如下：

> *Let miss be the smallest sum in [0,n] that we might be missing. Meaning we already know we can build all sums in [0,missInt). Then if we have a number num <= missInt in the given array, we can add it to those smaller sums to build all sums in [0, missInt+num). If we don't, then we must add such a number to the array, and it's best to add missInt itself, to maximize the reach.*
>
> *Example: Let's say the input is nums = [1, 2, 4, 13, 43] and n = 100. We need to ensure that all sums in the range [1,100] are possible.*
>
> *Using the given numbers 1, 2 and 4, we can already build all sums from 0 to 7, i.e., the range [0,8). But we can't build the sum 8, and the next given number (13) is too large. So we insert 8 into the array. Then we can build all sums in [0,16).*
>
> *Do we need to insert 16 into the array? No! We can already build the sum 3, and adding the given 13 gives us sum 16. We can also add the 13 to the other sums, extending our range to [0,29).*
>
> *And so on. The given 43 is too large to help with sum 29, so we must insert 29 into our array. This extends our range to [0,58). But then the 43 becomes useful and expands our range to [0,101). At which point we're done.*

打赏

Liu Jiaming / 2016年3月19日 / 算法、编程 / Leetcode