

LeetCode

Explore

Problems

Mock

Contest

Articles

Discuss

Store

Articles

50. Pow(x, n)

Previous

Next

50. Pow(x, n)

Dec. 27, 2017 | 94.9K views

Average Rating: 4.02 (50 votes)

Implement `pow(x, n)`, which calculates  $x$  raised to the power  $n$  ( $x^n$ ).

**Example 1:**

**Input:** 2.00000, 10

**Output:** 1024.00000

**Example 2:**

**Input:** 2.10000, 3

**Output:** 9.26100

**Example 3:**

**Input:** 2.00000, -2

**Output:** 0.25000

**Explanation:**  $2^{-2} = 1/2^2 = 1/4 = 0.25$

**Note:**

- $-100.0 < x < 100.0$
- $n$  is a 32-bit signed integer, within the range  $[-2^{31}, 2^{31} - 1]$

Approach 1: Brute Force

Intuition

Just simulate the process, multiply `x` for `n` times.

If  $n < 0$ , we can substitute  $x, n$  with  $\frac{1}{x}, -n$  to make sure  $n \geq 0$ . This restriction can simplify our further discussion.

But we need to take care of the corner cases, especially different range limits for negative and positive integers.

Algorithm

We can use a straightforward loop to compute the result.

C++

Java

```
1 class Solution {
2 public:
3     double myPow(double x, int n) {
4         long long N = n;
5         if (N < 0) {
6             x = 1 / x;
7             N = -N;
8         }
9         double ans = 1;
10        for (long long i = 0; i < N; i++)
11            ans = ans * x;
12        return ans;
13    }
14};
```

Copy

Complexity Analysis

- Time complexity :  $O(n)$ . We will multiply `x` for `n` times.
- Space complexity :  $O(1)$ . We only need one variable to store the final product of `x`.

Approach 2: Fast Power Algorithm Recursive

Intuition

Assuming we have got the result of  $x^n$ , how can we get  $x^{2*n}$  ? Obviously we do not need to multiply `x` for another `n` times. Using the formula  $(x^n)^2 = x^{2*n}$ , we can get  $x^{2*n}$  at the cost of only one computation. Using this optimization, we can reduce the time complexity of our algorithm.

Algorithm

Assume we have got the result of  $x^{n/2}$ , and now we want to get the result of  $x^n$ . Let `A` be result of  $x^{n/2}$ , we can talk about  $x^n$  based on the parity of `n` respectively. If `n` is even, we can use the formula  $(x^n)^2 = x^{2*n}$  to get  $x^n = A * A$ . If `n` is odd, then  $A * A = x^{n-1}$ . Intuitively, We need to multiply another  $x$  to the result, so  $x^n = A * A * x$ . This approach can be easily implemented using recursion. We call this method "**Fast Power**", because we only need at most  $O(\log n)$  computations to get  $x^n$ .

C++

Java

```
1 class Solution {
2 public:
3     double fastPow(double x, long long n) {
4         if (n == 0) {
5             return 1.0;
6         }
7         double half = fastPow(x, n / 2);
8         if (n % 2 == 0) {
9             return half * half;
10        } else {
11            return half * half * x;
12        }
13    }
14    double myPow(double x, int n) {
15        long long N = n;
16        if (N < 0) {
17            x = 1 / x;
18            N = -N;
19        }
20        return fastPow(x, N);
21    }
22};
```

Copy

Complexity Analysis

- Time complexity :  $O(\log n)$ . Each time we apply the formula  $(x^n)^2 = x^{2*n}$ ,  $n$  is reduced by half. Thus we need at most  $O(\log n)$  computations to get the result.
- Space complexity :  $O(\log n)$ . For each computation, we need to store the result of  $x^{n/2}$ . We need to do the computation for  $O(\log n)$  times, so the space complexity is  $O(\log n)$ .

Approach 3: Fast Power Algorithm Iterative

Intuition

Using the formula  $x^{a+b} = x^a * x^b$ , we can write `n` as a sum of positive integers,  $n = \sum_i b_i$ . If we can get the result of  $x^{b_i}$  quickly, the total time for computing  $x^n$  will be reduced.

Algorithm

We can use the binary representation of `n` to better understand the problem. Let the binary representation of `n` to be  $b_1, b_2, \dots, b_{length\_limit}$ , from the Least Significant Bit(LSB) to the Most Significant Bit(MSB). For the `i`th bit, if  $b_i = 1$ , it means we need to multiply the result by  $x^{2^i}$ .

It seems to have no improvement with this representation, since  $\sum_i b_i * 2^i = n$ . But using the formula  $(x^n)^2 = x^{2*n}$  we mentioned above, we can see some differences. Initially  $x^1 = x$ , and for each  $i > 1$ , we can use the result of  $x^{2^{i-1}}$  to get  $x^{2^i}$  in one step. Since the number of  $b_i$  is at most  $O(\log n)$ , we can get all  $x^{2^i}$  in  $O(\log n)$  time. After that, for all `i`s that satisfy  $b_i = 1$ , we can multiply  $x^{2^i}$  to the result. This also requires  $O(\log n)$  time.

Using fast power recursively or iteratively are actually taking different paths towards the same goal. For more information about fast power algorithm, you can visit its wiki<sup>1</sup>.

C++

Java

```
1 class Solution {
2 public:
3     double myPow(double x, int n) {
4         long long N = n;
5         if (N < 0) {
6             x = 1 / x;
7             N = -N;
8         }
9         double ans = 1;
10        double current_product = x;
11        for (long long i = N; i ; i /= 2) {
12            if ((i % 2) == 1) {
13                ans = ans * current_product;
14            }
15            current_product = current_product * current_product;
16        }
17        return ans;
18    }
19};
```

Copy

Complexity Analysis

- Time complexity :  $O(\log n)$ . For each bit of `n`'s binary representation, we will at most multiply once. So the total time complexity is  $O(\log n)$ .
- Space complexity :  $O(1)$ . We only need two variables for the current product and the final result of `x`.

Footnotes

1. [https://en.wikipedia.org/wiki/Exponentiation\\_by\\_squaring](https://en.wikipedia.org/wiki/Exponentiation_by_squaring)

Rate this article:

Previous

Next

Comments: 18

Sort By

myproudname

46

August 30, 2018 3:30 AM

Report

Approach 1 is NOT accepted for submission! There is a red warning: Time Limit Exceeded! Why? Thank you!

33

Share

Reply

SHOW 5 REPLIES

calvinchankf

2917

April 24, 2019 3:11 PM

[Just for fu] besides the approaches above, u can also do it with a **dynamic programming** approach

- by splitting the n, we can get into subproblems.
- cache to avoid redundant calculation

Read More

18

Share

Reply

SHOW 2 REPLIES

BryanBo-Cao

1434

July 7, 2018 12:44 AM

Thanks for explanation. But why is there a `;` after a class in Java solution? That's for C++ class, but a typical Java class coding style doesn't need that.

11

Share

Reply

matonglidewazi

9

October 5, 2018 11:51 PM

Suggestion for Approach 2: Fast Power Algorithm Recursive:

Try not to use `n=-n` or it would cause int overflow on INT\_MIN, instead, try this:

```
double myPow(double x, int n){
```

Read More

9

Share

Reply

SHOW 1 REPLY

tzl00166

9

May 14, 2018 10:22 PM

Report

I know maybe it is a stupid question, but why need to transfer n from int to long?

7

Share

Reply

SHOW 7 REPLIES

mission\_2020

97

May 7, 2020 11:18 PM

this is tricky.dont think I can solve without knowing before !!!.practice and repeated practice needed

3

Share

Reply

katruskin

4

May 23, 2020 6:29 PM

zero in power zero is undefine, not 1 as in accepted testcase.

1

Share

Reply

deva402

11

April 20, 2019 3:55 AM

Report

I am having hard time understanding description of approach # 3. Can someone help me with some pointers? Thanks

1

Share

Reply

SHOW 3 REPLIES

kosmur

142

February 14, 2019 1:57 PM

There is one more solution - math solution.  
We can use natural logarithm rule ->  $y = E^{Ln(y)}$  - where **E** is Euler's number and **Ln** is logarithm.  
Now let's substitute **y** with our case  $x^n$ , we will get  $x^n = E^{Ln(x^n)}$ .  
After we get last expression, we can apply **logarithm power rule**  $Log(a^b) = b * Log(a)$ , so we can cast our last expression from  $x^n = E^{Ln(x^n)}$  to  $x^n = E^{n * Ln(x)}$ .

Read More

1

Share

Reply

SHOW 4 REPLIES

ProfNandaa

36

December 17, 2018 10:48 PM

JS function (approach #2):

```
var myPow = function(x, n) {
  const fastPow = (x, n) => {
    if (n == 0) return 1
```

Read More

1

Share

Reply

<

1

2

>