CoderCareer: Discussing Coding Interview Questions from Google, Amazon, Facebook, Microsoft, etc

Thursday, October 27, 2011

No. 15 - Fibonacci Sequences

Problem: Please implement a function which returns the n^{th} number in Fibonacci sequences with an input n. Fibonacci sequence is defined as:

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f(n-1) + f(n-2) & n > 1 \end{cases}$$

Analysis: It is a classic interview questions to get numbers in Fibonacci sequences. We have different solutions for it, and their performance varies a lot.

Solution 1: Inefficient recursive solution

Fibonacci sequences are taken as examples to lecture recursive functions in many C/C++ textbooks, so most of candidates are familiar with the recursive solution. They feel confident and delighted when they meet this problem during interviews, because the can write the following code in short time:

```
long long Fibonacci(unsigned int n)
{
    if(n <= 0)
        return 0;

    if(n == 1)
        return 1;

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

Our textbooks take Fibonacci sequences as examples for recursive functions does not necessarily mean that recursion is a good solution for Fibonacci sequences. Interviewers may tell candidates that the performance of this recursive solution is quite bad, and ask them to analyze root causes.

Let us take f(10) as an example to analyze the recursive process. We have to get f(9) and f(8) before we get f(10). Meanwhile, f(8) and f(7) are needed before we get f(9). The dependency can be visualized in a tree as shown in Figure 1:

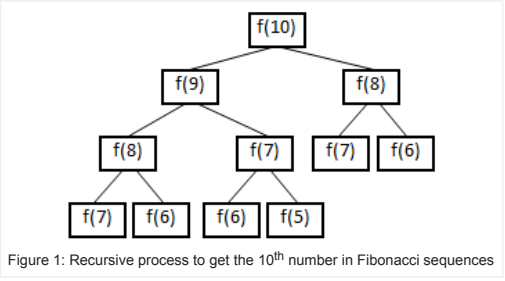


Figure 1: Recursive process to get the 10^{th} number in Fibonacci sequences

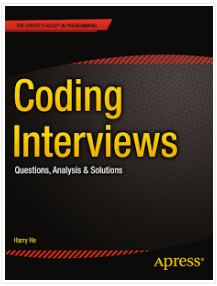
It is not difficult to notice that there are many duplicate nodes in the tree in Figure 1. The number of duplicated nodes increases dramatically when n increases. Readers may have a try on the 100^{th} number if Fibonacci sequences to have intuitive ideas about how slow this recursive solution is.

Solution 2: Practical Solution with O(n) efficiency

It is easy to optimize performance fortunately if we calculate from bottom. That is to say, we get f(2) based on f(0) and f(1), and get f(3) based on f(1) and f(2). We follow this pattern till we get f(n). It is obvious that its time complexity is O(n). Its corresponding code is shown below:

```
long long Fibonacci(unsigned n)
{
    int result[2] = {0, 1};
    if(n < 2)
```

My Book Has Been Published



Coding Interviews: Questions, Analysis & Solutions. Click the cover image to view the details on Amazon.com.

Favarite links

- TuringCamp: Online algorithm training

Pages

- Blog Home
- Coding Interviews: Questions, Analysis & Solutions
- Data Structure Interview Questions
- Linked List Interview Questions
- Binary Tree Interview Questions
- String Interview Questions
- Array and Matrix Interview Questions
- Stack and Queue Interview Questions
- Algorithm Interview Questions
- Dynamic Programming Interview Questions
- Search and Sort Interview Questions
- Backtracking Interview Questions
- Bit Manipulation Interview Questions
- Programming Interview Questions
- Amazon Interview Questions
- Google Interview Questions
- Facebook Interview Questions
- Microsoft Interview Questions

Blog Archive

```
        return result[n];

    long long  fibNMinusOne = 1;
    long long  fibNMinusTwo = 0;
    long long  fibN = 0;
    for(unsigned int i = 2; i <= n; ++ i)
    {
        fibN = fibNMinusOne + fibNMinusTwo;

        fibNMinusTwo = fibNMinusOne;
        fibNMinusOne = fibN;
    }

     return fibN;
}
```

Solution 3: O(logn) solution

Usually interviewers expect the O(n) solution above. However, there is an O(*log*n) solution available, which is based on an uncommon equation as shown below:

$$\begin{bmatrix} f(n) & f(n-1) \\ f(n-1) & f(n-2) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1}$$

It is not difficult to prove this equation vithmathematical induction. Interested readers may have try.

Now the only problem is how to calculate power of a matrix. We can calculate power with exponent n in O(*log*n) time with the following equation:

$$a^n = \begin{cases} a^{n/2} * a^{n/2} & n \text{ is even} \\ a^{(n-1)/2} * a^{(n-1)/2} * a & n \text{ is odd} \end{cases}$$

The source code to get power of a matrix looks complicated, which is listed below:

```
#include <cassert>

struct Matrix2By2
{
    Matrix2By2
    (
        long long m00 = 0,
        long long m01 = 0,
        long long m10 = 0,
        long long m11 = 0
    )
    :m_00(m00), m_01(m01), m_10(m10), m_11(m11)
    {
    }

    long long m_00;
    long long m_01;
    long long m_10;
    long long m_11;
};

Matrix2By2 MatrixMultiply
(
    const Matrix2By2& matrix1,
    const Matrix2By2& matrix2
)
{
    return Matrix2By2(
        matrix1.m_00 * matrix2.m_00 + matrix1.m_01 * matrix2.m_10,
        matrix1.m_00 * matrix2.m_01 + matrix1.m_01 * matrix2.m_11,
        matrix1.m_10 * matrix2.m_00 + matrix1.m_11 * matrix2.m_10,
        matrix1.m_10 * matrix2.m_01 + matrix1.m_11 * matrix2.m_11);
}

Matrix2By2 MatrixPower(unsigned int n)
{
    assert(n > 0);

    Matrix2By2 matrix;
    if(n == 1)
    {
        matrix = Matrix2By2(1, 1, 1, 0);
    }
    else if(n % 2 == 0)
    {
        matrix = MatrixPower(n / 2);
        matrix = MatrixMultiply(matrix, matrix);
    }
    else if(n % 2 == 1)
    {
        matrix = MatrixPower((n - 1) / 2);
        matrix = MatrixMultiply(matrix, matrix);
        matrix = MatrixMultiply(matrix, Matrix2By2(1, 1, 1, 0));
    }
```

No. 11 - Print Binary Trees from Top to Bottom
No. 10 - K-th Node from End
No. 09 - Numbers with a Given Sum
No. 08 - Calculate 1+2+...+n
► September (7)

About Me

Harry He
Follow 0
View my complete profile

```
    return matrix;
}

long long Fibonacci(unsigned int n)
{
    int result[2] = {0, 1};
    if(n < 2)
        return result[n];

    Matrix2By2 PowerNMinus2 = MatrixPower(n - 1);
    return PowerNMinus2.m_00;
}
```

Even though it cost only O(*log*n) time in theory, its hidden constant parameter is quite big, so it is not treated as a practical solution in real software development. Additionally, it is not a recommended solution during interviews since its implementation code is very complex.

The discussion about this problem is included in my book <Coding Interviews: Questions, Analysis & Solutions>, with some revisions. You may find the details of this book on Amazon.com, or Apress.

Posted by Harry He at 11:48 AM

Labels: Algorithm, Amazon, Interview Question, Microsoft

10 comments:

buddystripes February 1, 2013 at 1:53 AM

Hey! Thanks for posting all of these cool problems. I wanted to suggest an ever faster solution. It's constant time if you solve out the closed form solution for the nth fibonacci number.

http://math.stackexchange.com/questions/55922/fibonacci-recurrence-relations

Reply

Replies

Harry He February 4, 2013 at 3:40 PM

Thanks for your reply. The equation listed in the URL you provided is cool, but is not better than the solution in this post: (1) Precision issue. The equation is based on the value of sqrt(5). The calculation on double values is not as accurate as calculation on integer values. (2) The time efficiency is not constant. We can only calculate [(1 + sqrt(5)) / 2]^n and [(1 - sqrt(5)) / 2]^n in O(lgn) time.

Sahil Rally October 16, 2013 at 4:55 PM

Hi Harry

Why to consider Precision Issue iff it is giving accurate answer ?

Reply

Satish Dixit September 25, 2015 at 12:29 AM

very nice explanation.One of the different fibonacci problem is mentioned in this link http://techno-terminal.blogspot.in/2015/09/fibonacci-problem-calculate-sum-fx-fx-1.html

Reply

djax February 8, 2016 at 6:26 AM

Binet's formula solves this instantaneously.

Reply

Replies

djax February 8, 2016 at 6:27 AM

Accuracy is maintained, all the floating values cancel in the numerator and denominator

djax February 8, 2016 at 6:28 AM

"The calculation on double values not as accurate..." Who would ever ask for the 2.2th fibonacci number? That is meaningless.

djax February 8, 2016 at 6:35 AM

Accuracy is maintained, all the floating values cancel in the numerator and denominator

Reply

djax February 8, 2016 at 6:37 AM

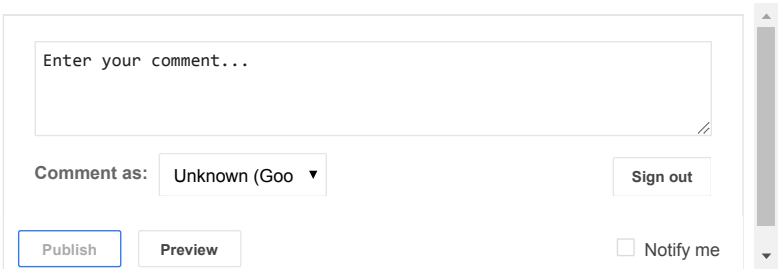
Binet's formula solves this instantaneously.

Reply

Unknown May 19, 2017 at 4:48 AM

Solution one is not only inefficient it is also incorrect.

Reply



Newer Post Home Older Post

Subscribe to: Post Comments (Atom)