Is Fibo

All topics

Fibonacci Numbers

Fibonacci Numbers are

$$0, 1, 1, 2, 3, 5, 8, 13 \ldots$$

Fibonacci numbers are generated using the following recurrence relation

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ &\vdots \\ F_i &= F_{i-1} + F_{i-2} \ for\ i \geq 2 \end{aligned}$$

It is interesting to note that the n^{th} Fibonacci number grows so fast that F_{47} exceeds the 32-bit signed integer range.

The fastest way to accurately compute Fibonacci numbers is by using a matrix-exponentiation method.

$$\begin{pmatrix} F_{k+2} \\ F_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{k+1} \\ F_k \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

We need to calculate M^N to calculate the Nth fibonacci number. We can calculate it in $O(log(N))$ using fast exponentiation.

Simple Recursive Solution

```
fibonacci(n)
    if(n=0)
        return 0
    if (n=1)
        return 1
    return (fibonacci(n-1)+fibonacci(n-2))
```

Time complexity:
$T(n) = T(n-1) + T(n-2)$

Go to Top

$T(n) = O(2^n)$

Optimization using Dynamic Programming

There are only n overlapping subproblems which can be stored to reduce the time complexity to $O(n)$

```
//Initialize all elements in dp to -1
fibonacci(n)
    if(dp[n]!=-1)
        return dp[n]
    if(n=0)
        dp[n]=0
    else if (n=1)
        dp[n]=1
    else
        dp[n]=fibonacci(n-1)+fibonacci(n-2)
    return dp[n]
```

Time Complexity = $O(n)$
Space Complexity = $O(n)$

Using Matrix Exponentiation

```
//Calculating A^p in O(log(P))
Matrix_pow ( Matrix A,int p )
    if(p=1)
        return A
    if(p%2=1)
        return A*Matrix_pow(A,p-1)

    Matrix B = Matrix_pow(a,p/2);
    return B * B

fibonacci(n)
    if(n=0)
        return 0;
    if(n=1)
        return 1;
    Matrix M[2][2]={{1,1},{1,0}}
    Matrix res=matrix_pow(M,n-1);
    return res[0][0];
```

Time Complexity = $O(log(n))$

Related challenge for **Fibonacci Numbers**

Fibonacci Finding (easy)

Success Rate: 35.74% **Max Score: 30** **Difficulty:**

Solve Challenge

Binet's Forumula

According to Binet's formula, a Fibonacci number is given by

$$F_n = \frac{\varphi^n - \psi^n}{\varphi - \psi} = \frac{\varphi^n - \psi^n}{\sqrt{5}}$$

where, $\varphi = \frac{1+\sqrt{5}}{2}$ and $\psi = \frac{1-\sqrt{5}}{2}$

solving for n gives

Go to Top

$$n = \log_{\varphi}\left(\frac{F_n\sqrt{5} + \sqrt{5F_n^2 \pm 4}}{2}\right)$$

This formula must return an integer for all n, so the expression under the radical must be an integer (otherwise, the logarithm does not even return a rational number).

Hence, for x to be a Fibonacci number, $5x^2 + 4$ or $5x^2 - 4$ must be a perfect square.

Precomputation

Precomputation is a technique where we try to use memory to store solutions in advance, such as values that are computed multiple times in a challenge, so that they can be converted to a memory look up.

Clearly lookups are $O(1)$ in complexity and hence reduce the time of computation overall to a great extent.

For example, suppose there are 10,000 queries on finding $\binom{n}{r}$. If we precalculate factorials this task becomes a O(1) as we just have to multiply and divide. Otherwise we have to do it in O(n) every time.