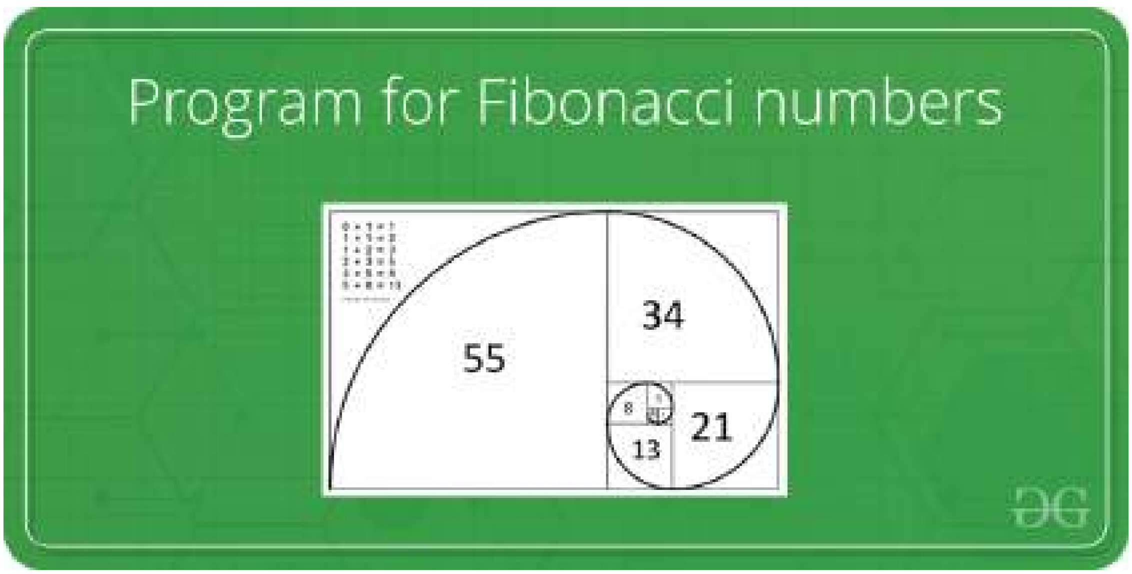
Nth Fibonacci Number

Given a number **n**, print **n-th Fibonacci Number**.

> *The Fibonacci numbers are the numbers in the following integer sequence:* ***0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..***



Examples:

***Input** : n = 1*

***Output** : 1*

***Input** : n = 9*

***Output** : 34*

***Input** : n = 10*

***Output** : 55*

Recommended Problem

Nth Fibonacci Number

Solve Problem

Dynamic Programming Mathematical +2 more

Amazon Microsoft +6 more

Submission count: 2.7L

Iterative Approach to Find and Print Nth Fibonacci Numbers:

In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation: $F_n = F_{n-1} + F_{n-2}$ *with seed values and* $F_0 = 0$ *and* $F_1 = 1$*.*

C++

```
// Fibonacci Series using Space Optimized Method
#include <bits/stdc++.h>
using namespace std;

int fib(int n)
{
    int a = 0, b = 1, c, i;
    if (n == 0)
```

```
        a = b;
        b = c;
    }
    return b;
}

// Driver code
int main()
{
    int n = 9;

    cout << fib(n);
    return 0;
}

// This code is contributed by Code_Mech
```

C

```
// Fibonacci Series using Space Optimized Method
#include <stdio.h>
int fib(int n)
{
    int a = 0, b = 1, c, i;
    if (n == 0)
        return a;
    for (i = 2; i <= n; i++) {
        c = a + b;
        a = b;
        b = c;
    }
    return b;
}

int main()
{
    int n = 9;
    printf("%d", fib(n));
    getchar();
    return 0;
}
```

Java

```java
public class fibonacci {
    static int fib(int n)
    {
        int a = 0, b = 1, c;
        if (n == 0)
            return a;
        for (int i = 2; i <= n; i++) {
            c = a + b;
            a = b;
            b = c;
        }
        return b;
    }

    public static void main(String args[])
    {
        int n = 9;
        System.out.println(fib(n));
    }
};

// This code is contributed by Mihir Joshi
```

Python3

```python
# Function for nth fibonacci number - Space Optimisation
# Taking 1st two fibonacci numbers as 0 and 1


def fibonacci(n):
    a = 0
    b = 1
    if n < 0:
        print("Incorrect input")
    elif n == 0:
        return a
    elif n == 1:
        return b
    else:
        for i in range(2, n+1):
            c = a + b
            a = b
            b = c
        return b
```

```
print(fibonacci(9))

# This code is contributed by Saket Modi
```

C#

```
// C# program for Fibonacci Series
// using Space Optimized Method
using System;

namespace Fib {
public class GFG {
    static int Fib(int n)
    {
        int a = 0, b = 1, c = 0;

        // To return the first Fibonacci number
        if (n == 0)
            return a;

        for (int i = 2; i <= n; i++) {
            c = a + b;
            a = b;
            b = c;
        }

        return b;
    }

    // Driver function
    public static void Main(string[] args)
    {

        int n = 9;
        Console.Write("{0} ", Fib(n));
    }
}
}

// This code is contributed by Sam007.
```

Javascript

```
function fib(n)
{
    let a = 0, b = 1, c, i;
    if( n == 0)
        return a;
    for(i = 2; i <= n; i++)
    {
    c = a + b;
    a = b;
    b = c;
    }
    return b;
}

// Driver code

    let n = 9;

    document.write(fib(n));

// This code is contributed by Mayank Tyagi
```

PHP

```
<?php
// PHP program for Fibonacci Series
// using Space Optimized Method

function fib( $n)
{
    $a = 0;
    $b = 1;
    $c;
    $i;
    if( $n == 0)
        return $a;
    for($i = 2; $i <= $n; $i++)
    {
        $c = $a + $b;
        $a = $b;
        $b = $c;
    }
    return $b;
}
```

```
echo fib($n);

// This code is contributed by anuj_67.
?>
```


Output

```
34
```

Time Complexity: O(n)

Auxiliary Space: O(1)

Recursion Approach to Find and Print Nth Fibonacci Numbers:

> *In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation:* $F_n = F_{n-1} + F_{n-2}$ *with seed values and* $F_0 = 0$ *and* $F_1 = 1$*.*

The Nth Fibonacci Number can be found using the recurrence relation shown above:

- if *n* = 0, then return 0.
- If n = 1, then it should return 1.
- For n > 1, it should return $F_{n-1} + F_{n-2}$

Below is the implementation of the above approach:

C++

```
#include <bits/stdc++.h>
using namespace std;

int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n - 1) + fib(n - 2);
}

int main()
{
    int n = 9;
    cout << n << "th Fibonacci Number: " << fib(n);
    return 0;
}

// This code is contributed
// by Akanksha Rai
```

C

```
// Fibonacci Series using Recursion
#include <stdio.h>
int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n - 1) + fib(n - 2);
}

int main()
{
    int n = 9;
    printf("%dth Fibonacci Number: %d", n, fib(n));
    return 0;
}
```

Java

```
// Fibonacci Series using Recursion
import java.io.*;
class fibonacci {
```

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

    public static void main(String args[])
    {
        int n = 9;
        System.out.println(
            n + "th Fibonacci Number: " + fib(n));
    }
}
/* This code is contributed by Rajat Mishra */
```

Python3

```python
# Fibonacci series using recursion
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)


if __name__ == "__main__":
    n = 9
    print(n, "th Fibonacci Number: ")
    print(fibonacci(n))

 # This code is contributed by Manan Tyagi.
```

C#

```csharp
// C# program for Fibonacci Series
// using Recursion
using System;

public class GFG {
    public static int Fib(int n)
    {
        if (n <= 1) {
            return n;
        }
        else {
            return Fib(n - 1) + Fib(n - 2);
```

```
    // driver code
    public static void Main(string[] args)
    {
        int n = 9;
        Console.Write(n + "th Fibonacci Number: " + Fib(n));
    }
}

// This code is contributed by Sam007
```

Javascript

```
// Javascript program for Fibonacci Series
// using Recursion

function Fib(n) {
  if (n <= 1) {
    return n;
  } else {
    return Fib(n - 1) + Fib(n - 2);
  }
}

// driver code
let n = 9;
console.log(n + "th Fibonacci Number: " + Fib(n));
```

PHP

```
<?php
// PHP program for Fibonacci Series
// using Recursion

function Fib($n)
{
    if ($n <= 1) {
        return $n;
    }
    else {
        return Fib($n - 1) + Fib($n - 2);
    }
}
```

```
// This code is contributed by Sam007
?>
```


Output

```
34
```

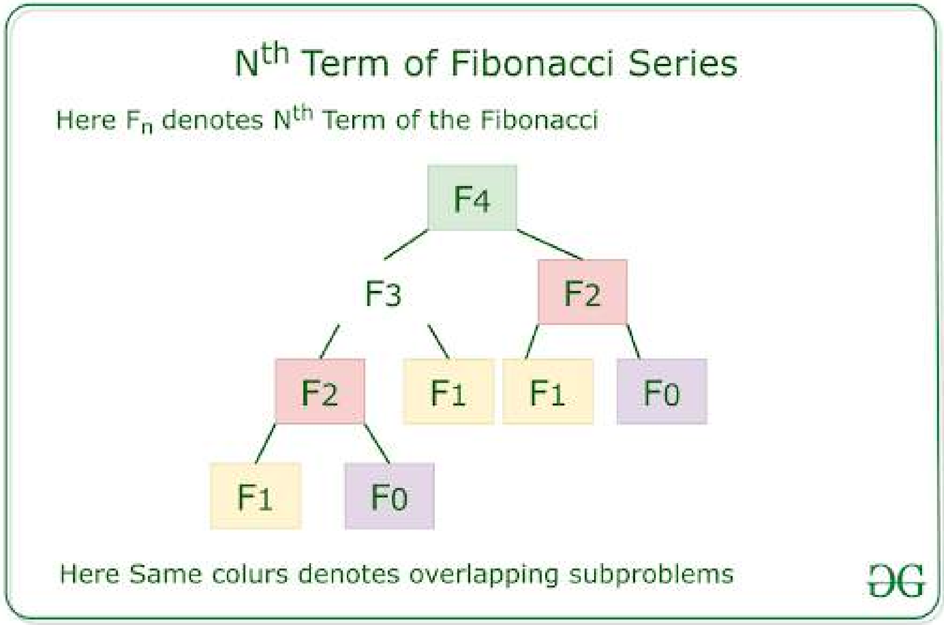
Time Complexity: Exponential, as every function calls two other functions.
Auxiliary space complexity: O(n), as the maximum depth of the recursion tree is n.

Dynamic Programming Approach to Find and Print Nth Fibonacci Numbers:

Consider the Recursion Tree for the 5th Fibonacci Number from the above approach:

```
                         fib(5)
                     /                \
                fib(4)                fib(3)
              /        \              /       \
          fib(3)      fib(2)         fib(2)   fib(1)
         /    \       /    \        /      \
   fib(2)   fib(1)  fib(1) fib(0) fib(1) fib(0)
   /     \
 fib(1) fib(0)
```

If you see, the same method call is being done multiple times for the same value. This can be optimized with the help of Dynamic Programming. We can



Dynamic Programming Approach to Find and Print Nth Fibonacci Numbers:

Below is the implementation of the above approach:

C++

```
// C++ program for Fibonacci Series
// using Dynamic Programming
#include <bits/stdc++.h>
using namespace std;

class GFG {

public:
    int fib(int n)
    {

        // Declare an array to store
        // Fibonacci numbers.
        // 1 extra to handle
        // case, n = 0
        int f[n + 2];
        int i;

        // 0th and 1st number of the
        // series are 0 and 1
        f[0] = 0;
        f[1] = 1;
```

```
            // in the series and store it
            f[i] = f[i - 1] + f[i - 2];
        }
        return f[n];
    }
};

// Driver code
int main()
{
    GFG g;
    int n = 9;

    cout << g.fib(n);
    return 0;
}

// This code is contributed by SoumikMondal
```

C

```
// Fibonacci Series using Dynamic Programming
#include <stdio.h>

int fib(int n)
{
    /* Declare an array to store Fibonacci numbers. */
    int f[n + 2]; // 1 extra to handle case, n = 0
    int i;

    /* 0th and 1st number of the series are 0 and 1*/
    f[0] = 0;
    f[1] = 1;

    for (i = 2; i <= n; i++) {
        /* Add the previous 2 numbers in the series
           and store it */
        f[i] = f[i - 1] + f[i - 2];
    }

    return f[n];
}

int main()
{
```

```
    return 0;
}
```

Java

```
// Fibonacci Series using Dynamic Programming
public class fibonacci {
    static int fib(int n)
    {
        /* Declare an array to store Fibonacci numbers. */
        int f[]
            = new int[n
                      + 2]; // 1 extra to handle case, n = 0
        int i;

        /* 0th and 1st number of the series are 0 and 1*/
        f[0] = 0;
        f[1] = 1;

        for (i = 2; i <= n; i++) {
            /* Add the previous 2 numbers in the series
              and store it */
            f[i] = f[i - 1] + f[i - 2];
        }

        return f[n];
    }

    public static void main(String args[])
    {
        int n = 9;
        System.out.println(fib(n));
    }
};
/* This code is contributed by Rajat Mishra */
```

Python3

```
# Fibonacci Series using Dynamic Programming
def fibonacci(n):

    # Taking 1st two fibonacci numbers as 0 and 1
    f = [0, 1]
```

```
    return f[n]


print(fibonacci(9))
```

C#

```
// C# program for Fibonacci Series
// using Dynamic Programming
using System;
class fibonacci {

    static int fib(int n)
    {

        // Declare an array to
        // store Fibonacci numbers.
        // 1 extra to handle
        // case, n = 0
        int[] f = new int[n + 2];
        int i;

        /* 0th and 1st number of the
           series are 0 and 1 */
        f[0] = 0;
        f[1] = 1;

        for (i = 2; i <= n; i++) {
            /* Add the previous 2 numbers
               in the series and store it */
            f[i] = f[i - 1] + f[i - 2];
        }

        return f[n];
    }

    // Driver Code
    public static void Main()
    {
        int n = 9;
        Console.WriteLine(fib(n));
    }
}

// This code is contributed by anuj_67.
```

Javascript

```
// Fibonacci Series using Dynamic Programming

    function  fib(n)
    {
        /* Declare an array to store Fibonacci numbers. */
        let f = new Array(n+2); // 1 extra to handle case, n = 0
        let i;
        /* 0th and 1st number of the series are 0 and 1*/
        f[0] = 0;
        f[1] = 1;
        for (i = 2; i <= n; i++)
        {
            /* Add the previous 2 numbers in the series
            and store it */
            f[i] = f[i-1] + f[i-2];
        }
        return f[n];
    }
    let n=9;
    document.write(fib(n));

    // This code is contributed by avanitrachhadiya2155
```

PHP

```
<?php
//Fibonacci Series using Dynamic
// Programming

function fib( $n)
{

    /* Declare an array to store
    Fibonacci numbers. */

    // 1 extra to handle case,
    // n = 0
    $f = array();
    $i;
```

```
    $f[0] = 0;
    $f[1] = 1;

    for ($i = 2; $i <= $n; $i++)
    {

        /* Add the previous 2
        numbers in the series
        and store it */
        $f[$i] = $f[$i-1] + $f[$i-2];
    }

    return $f[$n];
}

$n = 9;
echo fib($n);

// This code is contributed by
// anuj_67.
?>
```


Output

```
34
```

Time complexity: O(n) for given n

Auxiliary space: O(n)

Nth Power of Matrix Approach to Find and Print Nth Fibonacci Numbers

> *This approach relies on the fact that if we n times multiply the matrix M = {{1,1},{1,0}} to itself (in other words calculate power(M, n)), then we get*

the (n+1)th Fibonacci number as the element at row and column (0, 0) in the resultant matrix.

- *If n is even then k = n/2:*
  - *Therefore Nth Fibonacci Number = F(n) = [2*F(k-1) + F(k)]*F(k)*
- *If n is odd then k = (n + 1)/2:*
  - *Therefore Nth Fibonacci Number = F(n) = F(k)*F(k) + F(k-1)*F(k-1)*

How does this formula work?

The formula can be derived from the matrix equation.

$$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}^n = \begin{bmatrix}F_{n+1} & F_n\\F_n & F_{n-1}\end{bmatrix}$$

Taking determinant on both sides, we get $(-1)^n = F_{n+1}F_{n-1} - F_n^2$

Moreover, since $A^nA^m = A^{n+m}$ *for any square matrix A, the following identities can be derived (they are obtained from two different coefficients of the matrix product)*

$F_mF_n + F_{m-1}F_{n-1} = F_{m+n-1}$ ———————————(1)

By putting n = n+1 in equation(1),

$F_mF_{n+1} + F_{m-1}F_n = F_{m+n}$ ———————————(2)

Putting m = n in equation(1).

$F_{2n-1} = F_n^2 + F_{n-1}^2$

Putting m = n in equation(2)

$F_{2n} = (F_{n-1} + F_{n+1})F_n = (2F_{n-1} + F_n)F_n$ ———

(By putting Fn+1 = Fn + Fn-1)

To get the formula to be proved, we simply need to do the following

- *If n is even, we can put k = n/2*

- *If n is odd, we can put k = (n+1)/2*

Below is the implementation of the above approach

C++

```
// Fibonacci Series using Optimized Method
#include <bits/stdc++.h>
using namespace std;

void multiply(int F[2][2], int M[2][2]);
void power(int F[2][2], int n);

// Function that returns nth Fibonacci number
int fib(int n)
{
    int F[2][2] = { { 1, 1 }, { 1, 0 } };
    if (n == 0)
        return 0;
    power(F, n - 1);

    return F[0][0];
}

// Optimized version of power() in method 4
void power(int F[2][2], int n)
{
    if (n == 0 || n == 1)
        return;
    int M[2][2] = { { 1, 1 }, { 1, 0 } };

    power(F, n / 2);
    multiply(F, F);

    if (n % 2 != 0)
        multiply(F, M);
}

void multiply(int F[2][2], int M[2][2])
{
    int x = F[0][0] * M[0][0] + F[0][1] * M[1][0];
    int y = F[0][0] * M[0][1] + F[0][1] * M[1][1];
    int z = F[1][0] * M[0][0] + F[1][1] * M[1][0];
```

```
    F[0][0] = x;
    F[0][1] = y;
    F[1][0] = z;
    F[1][1] = w;
}

// Driver code
int main()
{
    int n = 9;

    cout << fib(9);
    getchar();

    return 0;
}

// This code is contributed by Nidhi_biet
```

C

```
#include <stdio.h>

void multiply(int F[2][2], int M[2][2]);

void power(int F[2][2], int n);

/* function that returns nth Fibonacci number */
int fib(int n)
{
    int F[2][2] = { { 1, 1 }, { 1, 0 } };
    if (n == 0)
        return 0;
    power(F, n - 1);
    return F[0][0];
}

/* Optimized version of power() in method 4 */
void power(int F[2][2], int n)
{
    if (n == 0 || n == 1)
        return;
    int M[2][2] = { { 1, 1 }, { 1, 0 } };

    power(F, n / 2);
```

```
        multiply(F, M);
}

void multiply(int F[2][2], int M[2][2])
{
    int x = F[0][0] * M[0][0] + F[0][1] * M[1][0];
    int y = F[0][0] * M[0][1] + F[0][1] * M[1][1];
    int z = F[1][0] * M[0][0] + F[1][1] * M[1][0];
    int w = F[1][0] * M[0][1] + F[1][1] * M[1][1];

    F[0][0] = x;
    F[0][1] = y;
    F[1][0] = z;
    F[1][1] = w;
}

/* Driver program to test above function */
int main()
{
    int n = 9;
    printf("%d", fib(9));
    getchar();
    return 0;
}
```

Java

```
// Fibonacci Series using Optimized Method
public class fibonacci {
    /* function that returns nth Fibonacci number */
    static int fib(int n)
    {
        int F[][] = new int[][] { { 1, 1 }, { 1, 0 } };
        if (n == 0)
            return 0;
        power(F, n - 1);

        return F[0][0];
    }

    static void multiply(int F[][], int M[][])
    {
        int x = F[0][0] * M[0][0] + F[0][1] * M[1][0];
        int y = F[0][0] * M[0][1] + F[0][1] * M[1][1];
        int z = F[1][0] * M[0][0] + F[1][1] * M[1][0];
```

```
        F[0][1] = y;
        F[1][0] = z;
        F[1][1] = w;
    }

    /* Optimized version of power() in method 4 */
    static void power(int F[][], int n)
    {
        if (n == 0 || n == 1)
            return;
        int M[][] = new int[][] { { 1, 1 }, { 1, 0 } };

        power(F, n / 2);
        multiply(F, F);

        if (n % 2 != 0)
            multiply(F, M);
    }

    /* Driver program to test above function */
    public static void main(String args[])
    {
        int n = 9;
        System.out.println(fib(n));
    }
};
/* This code is contributed by Rajat Mishra */
```

Python3

```
# Fibonacci Series using
# Optimized Method

# function that returns nth
# Fibonacci number


def fib(n):

    F = [[1, 1],
         [1, 0]]
    if (n == 0):
        return 0
    power(F, n - 1)
```

```
def multiply(F, M):

    x = (F[0][0] * M[0][0] +
         F[0][1] * M[1][0])
    y = (F[0][0] * M[0][1] +
         F[0][1] * M[1][1])
    z = (F[1][0] * M[0][0] +
         F[1][1] * M[1][0])
    w = (F[1][0] * M[0][1] +
         F[1][1] * M[1][1])

    F[0][0] = x
    F[0][1] = y
    F[1][0] = z
    F[1][1] = w

# Optimized version of
# power() in method 4


def power(F, n):

    if(n == 0 or n == 1):
        return
    M = [[1, 1],
         [1, 0]]

    power(F, n // 2)
    multiply(F, F)

    if (n % 2 != 0):
        multiply(F, M)


# Driver Code
if __name__ == "__main__":
    n = 9
    print(fib(n))

# This code is contributed
# by ChitraNayal
```

C#

```
// Fibonacci Series using
```

```
class GFG {
    /* function that returns
    nth Fibonacci number */
    static int fib(int n)
    {
        int[, ] F = new int[, ] { { 1, 1 }, { 1, 0 } };
        if (n == 0)
            return 0;
        power(F, n - 1);

        return F[0, 0];
    }

    static void multiply(int[, ] F, int[, ] M)
    {
        int x = F[0, 0] * M[0, 0] + F[0, 1] * M[1, 0];
        int y = F[0, 0] * M[0, 1] + F[0, 1] * M[1, 1];
        int z = F[1, 0] * M[0, 0] + F[1, 1] * M[1, 0];
        int w = F[1, 0] * M[0, 1] + F[1, 1] * M[1, 1];

        F[0, 0] = x;
        F[0, 1] = y;
        F[1, 0] = z;
        F[1, 1] = w;
    }

    /* Optimized version of
    power() in method 4 */
    static void power(int[, ] F, int n)
    {
        if (n == 0 || n == 1)
            return;
        int[, ] M = new int[, ] { { 1, 1 }, { 1, 0 } };

        power(F, n / 2);
        multiply(F, F);

        if (n % 2 != 0)
            multiply(F, M);
    }

    // Driver Code
    public static void Main()
    {
        int n = 9;
        Console.Write(fib(n));
    }
```

```
// This code is contributed
// by ChitraNayal
```

Javascript

```
// Fibonacci Series using Optimized Method

// Function that returns nth Fibonacci number
function fib(n)
{
    var F = [ [ 1, 1 ], [ 1, 0 ] ];
    if (n == 0)
        return 0;

    power(F, n - 1);

    return F[0][0];
}

function multiply(F, M)
{
    var x = F[0][0] * M[0][0] + F[0][1] * M[1][0];
    var y = F[0][0] * M[0][1] + F[0][1] * M[1][1];
    var z = F[1][0] * M[0][0] + F[1][1] * M[1][0];
    var w = F[1][0] * M[0][1] + F[1][1] * M[1][1];

    F[0][0] = x;
    F[0][1] = y;
    F[1][0] = z;
    F[1][1] = w;
}

// Optimized version of power() in method 4 */
function power(F, n)
{
    if (n == 0 || n == 1)
        return;

    var M = [ [ 1, 1 ], [ 1, 0 ] ];

    power(F, n / 2);
    multiply(F, F);

    if (n % 2 != 0)
```

```
// Driver code
var n = 9;

document.write(fib(n));

// This code is contributed by gauravrajput1
```

Learn Data Structures & Algorithms with GeeksforGeeks

Output

```
34
```

Time Complexity: O(Log n)
Auxiliary Space: O(Log n) if we consider the function call stack size, otherwise O(1).

Related Articles:
Large Fibonacci Numbers in Java

Feeling lost in the world of random DSA topics, wasting time without progress? It's time for a change! Join our DSA course, where we'll guide you on an exciting journey to master DSA efficiently and on schedule.
Ready to dive in? Explore our Free Demo Content and join our DSA course, trusted by over 100,000 geeks!

- DSA in C++
- DSA in Java
- DSA in Python
- DSA in JavaScript

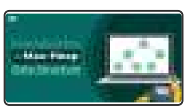
Similar Reads

- Check if a M-th fibonacci number divides N-th fibonacci number
- Check if sum of Fibonacci elements in an Array is a Fibonacci number or not
- C/C++ Program for nth multiple of a number in Fibonacci Series
- Program to find last two digits of Nth Fibonacci number
- Find nth Fibonacci number using Golden ratio
- Nth XOR Fibonacci number
- Nth Fibonacci number using Pell's equation
- Program to find Nth odd Fibonacci Number
- Find Nth Fibonacci Number using Binet's Formula
- GFact | Finding nth Fibonacci Number using Golden Ratio

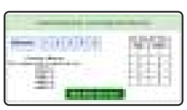
Related Tutorials


Mathematical and Geometric Algorithms - Data Structure and Algorithm Tutorials


Learn Data Structures with Javascript | DSA Tutorial


Introduction to Max-Heap – Data Structure and Algorithm Tutorials


Introduction to Set – Data Structure and Algorithm Tutorials


Introduction to Map – Data Structure and Algorithm Tutorials

Previous

How to check if a given number is Fibonacci number?

Next

C++ Program For Fibonacci Numbers

Article Contributed By :

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : 29AjayKumar, Rajput-Ji, Mithun Kumar, vt_m, ukasp, jit_t, ankthon, GauravRajput1, Akanksha_Rai, RishabhPrabhu, SoumikMondal, humblezero, princiraj1992, Code_Mech, LokeshMohanty, srinam, nidhi_biet, shivanisinghss2110, divyeshrabadiya07, amit143katiyar, 100rahulsingh100, mayanktyagi1709, sujitmeshram, avanitrachhadiya2155, arorakashish0911, rutuja2229, prasun_parate, bhavneet2000, surbhityagi15, rohit5swd, rdtank, mukesh07, sweetyty, dp0p, anikaseth98, cs19b080, gurjotssb2001, moontyagi786, bhawna1997, amartyaghoshgfg, guptaanushree821, rkbhola5, animeshdey, briancarrie765, drea, prasad264, harendrakumar123, anshitbansal, polymatir3j, purisaurabh2885, himanshug9119, anshu_1230, factworx4i2, vikramshirsath177, kartikmoyade0901, rohityadavcdece19, singhchfb3w, rvd31305, harshjainmat20

Article Tags : Amazon , Bloomberg , Fibonacci , MakeMyTrip , MAQ Software , matrix-exponentiation , Modular Arithmetic , series , Snapdeal , DSA , Dynamic Programming , Mathematical

Practice Tags : Amazon, Bloomberg, MakeMyTrip, MAQ Software, Snapdeal, Dynamic Programming, Fibonacci, Mathematical, Modular Arithmetic, series

Improve Article Report Issue

A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

- About Us
- Legal
- Terms & Conditions
- Careers
- In Media
- Contact Us
- Advertise with us
- GFG Corporate Solution
- Placement Training Program
- Apply for Mentor

Explore

- Job-A-Thon Hiring Challenge
- Hack-A-Thon
- GfG Weekly Contest
- Offline Classes (Delhi/NCR)
- DSA in JAVA/C++
- Master System Design
- Master CP
- GeeksforGeeks Videos

Languages

- Python
- Java
- C++
- PHP
- GoLang
- SQL
- R Language
- Android Tutorial

DSA Concepts

- Data Structures
- Arrays
- Strings
- Linked List
- Algorithms
- Searching
- Sorting
- Mathematical
- Dynamic Programming