Chapter 13

Fibonacci numbers

The Fibonacci numbers form a sequence of integers defined recursively in the following way. The first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

\[ F_n = \begin{cases} 0 & \text{for } n = 0, \\ 1 & \text{for } n = 1, \\ F_{n-1} + F_{n-2} & \text{for } n > 1. \end{cases} \]

The first twelve Fibonacci numbers are:

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
<td>34</td>
<td>55</td>
<td>89</td>
</tr>
</tbody>
</table>

Notice that recursive enumeration as described by the definition is very slow. The definition of \( F_n \) repeatedly refers to the previous numbers from the Fibonacci sequence.


```python
1 def fibonacci(n):
2     if (n <= 1):
3         return n
4     return fibonacci(n - 1) + fibonacci(n - 2)
```

The above algorithm performs \( F_n \) additions of 1, and, as the sequence grows exponentially, we get an inefficient solution.

Enumeration of the Fibonacci numbers can be done faster simply by using a basis of dynamic programming. We can calculate the values \( F_0, F_1, \ldots, F_n \) based on the previously calculated numbers (it is sufficient to remember only the last two values).

13.2: Finding Fibonacci numbers dynamically.

```python
1 def fibonacciDynamic(n):
2     fib = [0] * (n + 2)
3     fib[1] = 1
4     for i in xrange(2, n + 1):
5         fib[i] = fib[i - 1] + fib[i - 2]
6     return fib[n]
```

The time complexity of the above algorithm is \( O(n) \).

© Copyright 2015 by Codility Limited. All Rights Reserved. Unauthorized copying or publication prohibited.
13.1. Faster algorithms for Fibonacci numbers

Fibonacci numbers can be found in $O(\log n)$ time. However, for this purpose we have to use matrix multiplication and the following formula:

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}, \text{ for } n \geq 1.$$  

Even faster solution is possible by using the following formula:

$$Fib_n = \left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n \sqrt{5}$$  \hspace{2cm} (13.1)

These algorithms are not trivial and it will be presented in the future lessons.

13.2. Exercise

**Problem:** For all the given numbers $x_0, x_1, \ldots, x_{n-1}$, such that $1 \leq x_i \leq m \leq 1000000$, check whether they may be presented as the sum of two Fibonacci numbers.

**Solution:** Notice that only a few tens of Fibonacci numbers are smaller than the maximal $m$ (exactly 31). We consider all the pairs. If some of them sum to $k \leq m$, then we mark index $k$ in the array to denote that the value $k$ can be presented as the sum of two Fibonacci numbers.

In summary, for each number $x_i$ we can answer whether it is the sum of two Fibonacci numbers in constant time. The total time complexity is $O(n + m)$.

---

Every lesson will provide you with programming tasks at [http://codility.com/programmers](http://codility.com/programmers).