1 Fibonacci Number

Problem Introduction

Recall the definition of Fibonacci sequence: $F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$ for $i \geq 2$. Your goal in this problem is to implement an efficient algorithm for computing Fibonacci numbers. The starter files for this problem contain an implementation of the following naive recursive algorithm for computing Fibonacci numbers in `C++`, `Java`, and `Python3`:

```
FIBONACCI(n):
if n ≤ 1:
  return n
return FIBONACCI(n − 1) + FIBONACCI(n − 2)
```

Try compiling and running a starter solution on your machine. You will see that computing, say, F_{40} already takes noticeable time.

Another way to appreciate the dramatic difference between an exponential time algorithm and a polynomial time algorithm is to use the following visualization by David Galles: http://www.cs.usfca.edu/~galles/visualization/DPFib.html. Try computing F_{20} by a recursive algorithm by entering "20" and pressing the "Fibonacci Recursive" button. You will see an endless number of recursive calls. Now, press "Skip Forward" to stop the current algorithm and call the iterative algorithm by pressing "Fibonacci Table". This will compute F_{20} very quickly. (Note that the visualization uses a slightly different definition of Fibonacci numbers: $F_0 = 1$ instead of $F_0 = 0$. This of course has almost no influence on the running time.)

Problem Description

Task. Given an integer n, find the nth Fibonacci number F_n.

Input Format. The input consists of a single integer n.

Constraints. $0 \leq n \leq 45$.

Output Format. Output F_n.

Sample 1.

Input:

```
10
```

Output:

```
55
```

$F_{10} = 55$.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.