

Homework 3

CSC 1351, Spring 2006

Due: 20 April 2006

1. Solve Exercises R18.2, R18.6, and R18.10 on page 694 in the textbook. You do not need to implement and run them. It is good enough, if you show the code structure on paper. What is the complexity of these algorithms ($O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, etc.)?
2. Implement and test Exercise P18.16 on page 700 in the textbook. Compare the running time of four different implementations of the Fibonacci function: the recursive $O(2^n)$ algorithm (page 679), the iterative $O(n)$ algorithm (page 682), the *memoizing*¹ $O(n)$ algorithm described in Exercise P18.16, and the following recursive $O(n)$ algorithm:

```
public static int fib(int n) {
    return fib(n, 0, 1);
}

private static int fib(int n, int f1, int f2) {
    if (n <= 1) return f2;
    else return fib(n-1, f2, f1+f2);
}
```

Use `System.currentTimeMillis()` or class `StopWatch` from pages 708–709² to time these four implementations. What's the largest Fibonacci number that you can compute in under 10 seconds with each one of these algorithms on your machine? What's the overhead in percent of the two recursive $O(n)$ algorithms compared to the iterative algorithm?

BTW, for an short explanation of how and why Fibonacci came up with this sequence, check out

<http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Recn/Binary/>

¹This is not a typo, it is a technical term. A memoizing algorithm is an algorithm that stores (partial) results it already computed once for later use.

²You can get the source code from the publisher's Student Companion Site.