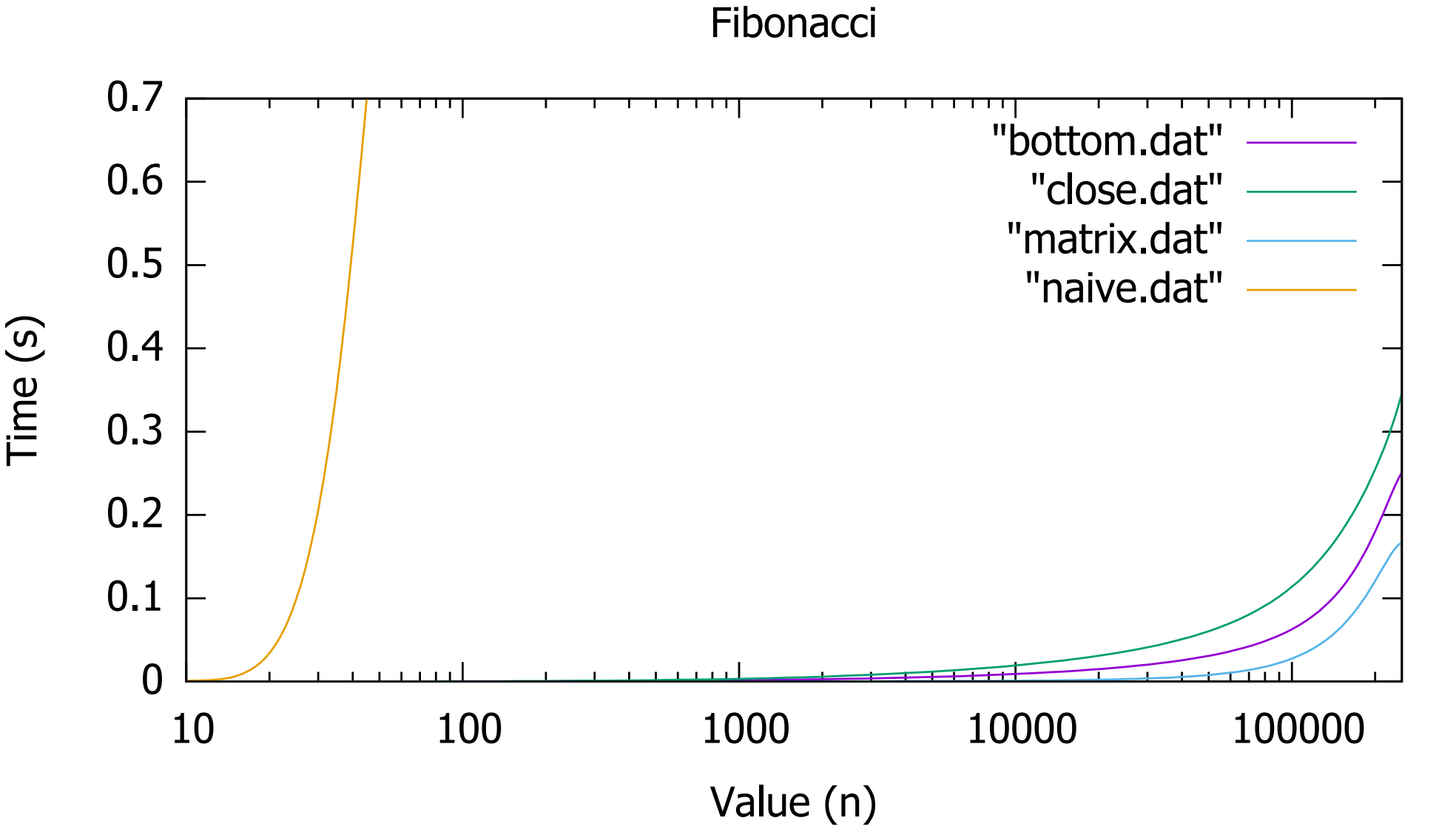
Fibonacci
Time (s)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
100
1000
10000
100000
Value (n)
"bottom.dat"
"close.dat"
"matrix.dat"
"naive.dat"

n	*Naive Algorithm*	*Bottom Up*	*Closed From*	*Matrix Algorithm*
10	0.001	0	0	0
20	0.002	0	0	0
30	0.009	0	0	0
40	0.593	0	0	0
50	1.265	0	0	0
100		0	0	0
500		0.0024	0.0011	0
1000		0.0021	0.0014	0
5000		0.014	0.04	0.001
10000		0.0267	0.03	0.002
50000		0.0294	0.05	0.002
100000		0.05	0.075	0.005
150000		0.062		0.012
200000				0.15

3.1 c) For the same n, do all methods always return the same Fibonacci number?

The naive recursive, bottom up and matrix representation algorithm was always returning the same number but the closed form sometimes did not. This happened because of the overflow and also because of the formula. When the number gets big enough it is out of range and not 100% correct.

3.1 d) Plot your results in a line plot, so that four approaches can be easily compared. Briefly interpret your results.

As we can see from the graph the naïve recursive algorithm is totally slow. It is impossible to continue with greater values than 50 as it requires lot of memory to call the function (itself). The matrix representation algorithm was the fastest as it could compute bigger number with less time and precision. In the middle are closed form and bottom up algorithm which were a bit slower than matrix one, but still better than naïve recursive function.