SCIENCE AND ENGINEERING DEPARTMENT

MASTER 1 PROJECT

GCD of polynomials by approximation/interpolation

Authors:
Mouataz El Bouhali
Elina Jankovskaja

Supervisor:
Vincent Neiger

December 2022

Contents

1 Introduction

This project aims to optimize fundamental operations such as the GCD for polynomes.
The project page can be found on github at: https://github.com/ElinaJnK/pgcd$_p$$olynomes(privatef$
$Afulldescriptionofhowtousetheprojectcanbefoundinthefile README.md.$

2 Implementations

2.1 NTL et FLINT

We first sought to measure the performance given to us by existing software. Thus, we can then try to improve them or at least get closer to their performance.

2.1.1 NTL

We started out by familiarizing ourselves with C++ and and the NTL library. After implementing different time measurement functions we found these results for different degrees of polynomials on different operations but with a fixed set of bits at 60 (the number will be on 60 bits) and generating a polynome with FFTInit (which is way faster then GenPrime_long which generates a polynome by using different FFT techniques). The commands we used looked like this

```
./graph <choice of implementation> <choice of operation>
<degree of polynomial> <number of bits> <file to save results (TODO)>
<choice of implementation> can either be
        - 0 : using a FFT generated polynomial
        - 1 : using a bits generated polynomial
<choice of operation> can be:
        - 0 : addition
        - 1 : multiplication
        - 2 : GCD
        - 3 : division  (divided by 2 by default (TODO))
        - 4 : XGCD
For example, you could try ./ntl 0 1 1000 60
Polynome generated with FFT
./ntl 0 1 10000000 60
```

Also, these operations were run on a macOS Monterey with a Apple M1 chip and 16GB of RAM. The computer was plugged in during these computations. Using this from 1000 degrees to 10000000 for addition we get the graph:

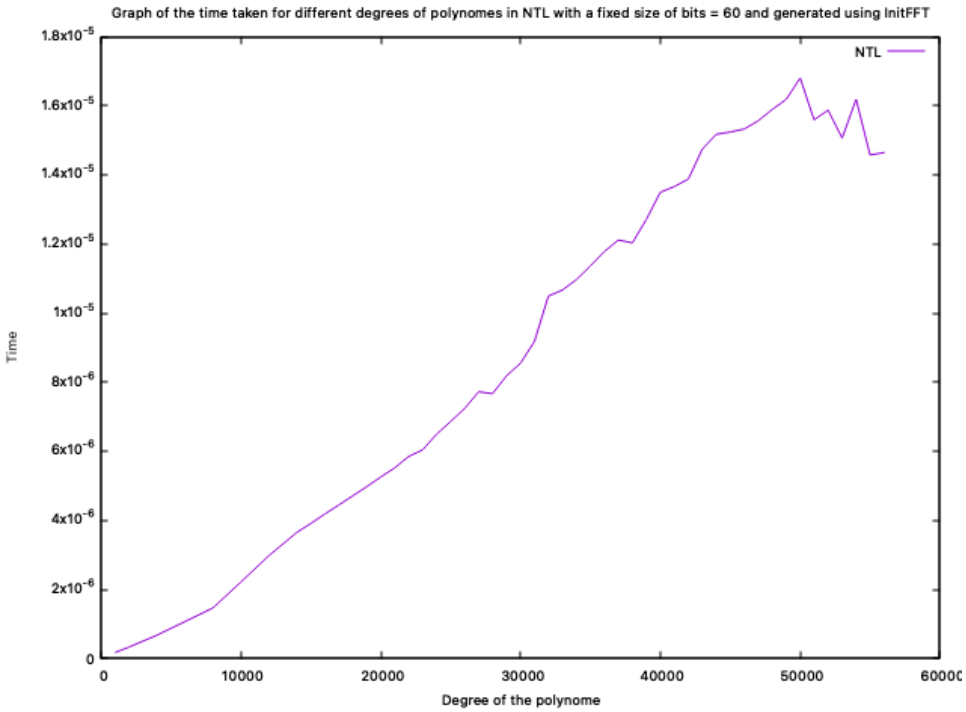

Figure 1: Addition in NTL

For multiplication we get:

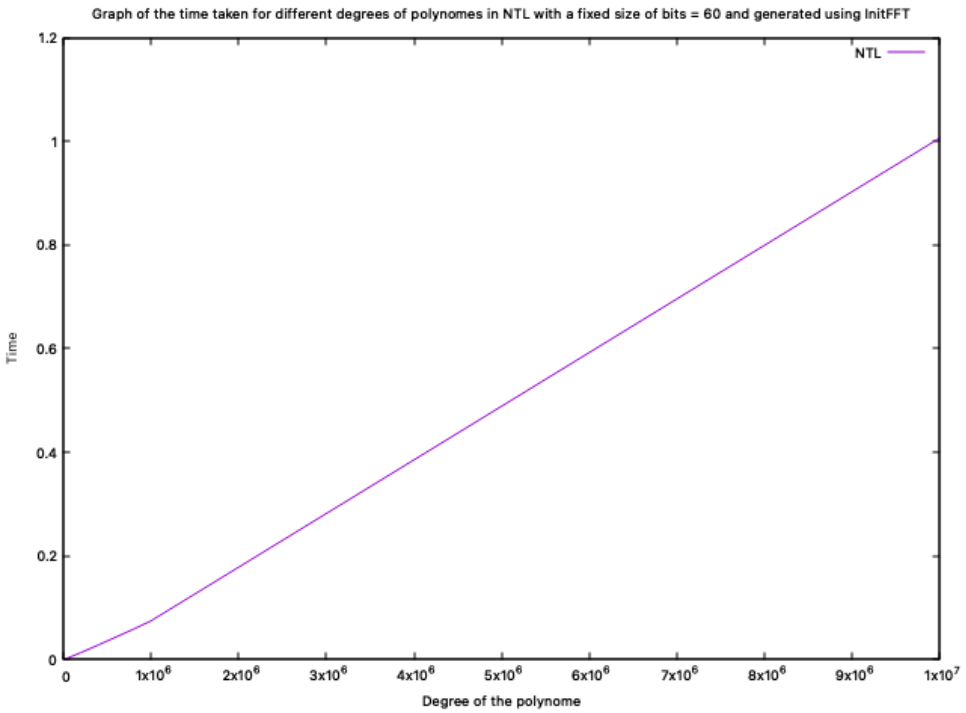

Figure 2: Multiplication in NTL

For the GCD we get:

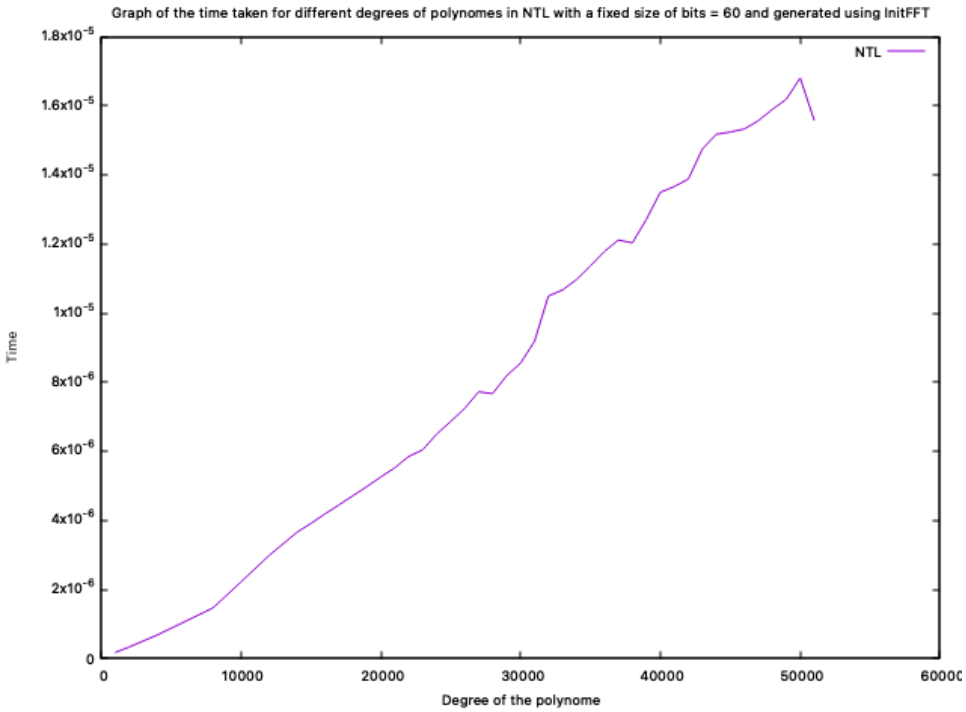

Figure 3: GCD in NTL

For the XGCD we get:

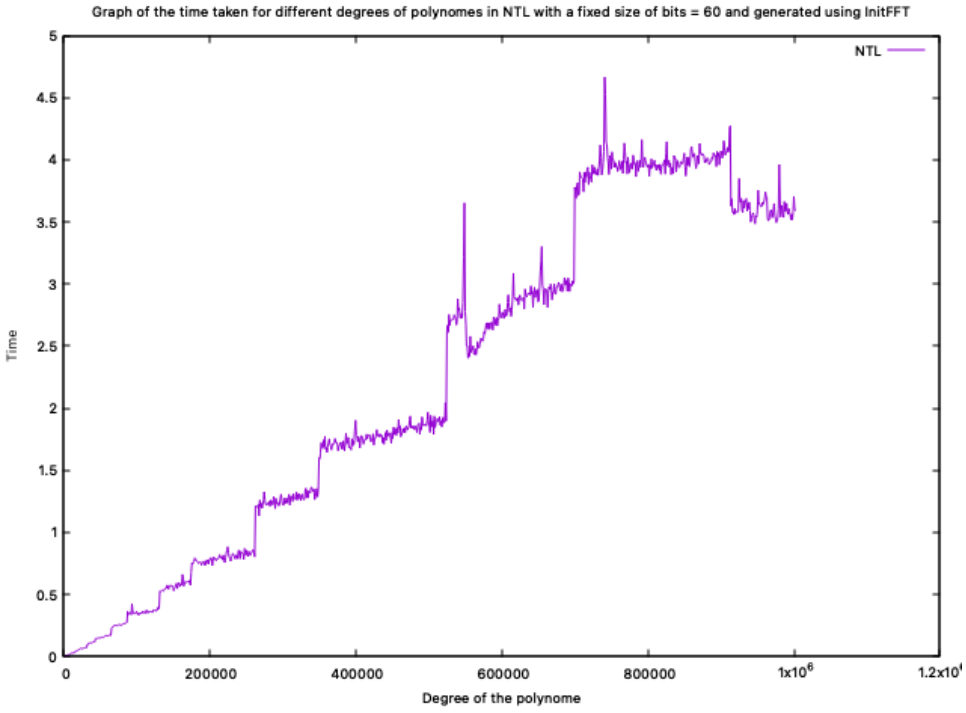

Figure 4: XGCD in NTL (a*u + b*v = p format)

We did not go as 10000000 here because it was taking close to a minute and the other data from the graph was hardly visible with that stretch.

We can see that the computation time is close to linear. But not exactly. More precisely, the complexity is $\mathcal{O}(d \log d)$ d being the degree of the polynome.

2.1.2 FLINT

3 Performances

4 Conclusion