Kipling Gillespie
CS315 - 003

Home Word 4

Algorithms

Division

For the division algorithm I used the algorithm from our text book which uses recursion. The calling BinVec is the numerator. It takes two BinVec reference arguments y(denominator), and r(remainder). It returns a BinVec q which is the resulting quotient. The algorithm checks whether the calling BinVec is odd by checking if it's Least Significant Bit is 1 and stores this in a bool for later use. If the decimal value for the calling BinVec is zero, the algorithm returns 0 for q and r. Otherwise, the value of x is divided by two by deleting the Least Significant Bit and recursively calling Div on x. The return value is stored in q and r. Both q and r are multiplied by 2 by adding zero as the new Least Significant Bit. We check our save bool that says if x was odd and if it was odd we add one to r. If r is now equal to or greater than the denominator we subtract the denominator from r and add one to the quotient. The quotient q and remainder r are then returned.

The effect this algorithm has is successive divisions by 2 of x until x equals zero. As the recursive stack unwinds, it performs modulus addition on the remainder whenever x is odd at that level, and increments q by 1 whenever the remainder wraps around. This method is faster than Successive subtractions, and has a quadratic run time.

ExpMod

For the exponential modulus algorithm, I used the non-recursive version that we learned in class. The return value is set to 1 and a BinVec w is set to the input value *this. It algorithm loops through each element of the BinVec y starting from least significant bit. If that bit is one, we calculate the new value for z by multiplying it by w and using the result as the input of my division algorithm, N as the input for the numerator, and z as the reference return value of the remainder. Whether the current bit of y is 0 or 1, we square the value of W and divide it by N and store the remainder back in w. Once we have checked every bit of y we will have the correct answer and we return z.

Run Time of Exponential Modulus

I used a function to generate binary values with n 0's and 1's. The chrono library was used to get the time before and after each run to calculate the run time. The recorded times are located at the bottom of this report with my input values. The run time was determined to be cubic based on inputs and results.

Build Info

My program was made with Code::Blocks 13.12 with the following build arguments.

```
mingw32-g++.exe -Wall -fexceptions -std=gnu++11 -O2  -c "C:\Users\Kipling Gillespie\Google Drive\School\CS 315\HW4\binvec.cpp" -o obj\Release\binvec.o
mingw32-g++.exe -Wall -fexceptions -std=gnu++11 -O2  -c "C:\Users\Kipling Gillespie\Google Drive\School\CS 315\HW4\main.cpp" -o obj\Release\main.o
mingw32-g++.exe  -o "bin\Release\CS315 HW4.exe" obj\Release\binvec.o obj\Release\main.o  -s
Output file is bin\Release\CS315 HW4.exe with size 478.00 KB
```

Run Instructions

The executable takes input through the command arguments. I just drag the exe into the console and follow it with the arguments. The first argument determines which algorithm it runs. 3 runs division, and r runs exponential modulus. The second argument is the value for x, and the third argument is the value for y, the fourth argument is N, but N is only used when running the exponential modulus algorithm. It takes

in binary values for x, y, and N with the left most bit the least significant bit and the right most value the most significant bit.

Inputs

Problem 1

Test 1

x = 011000111, y = 111101101, Quotient = 1

Remainder = 1110101

Test 2

x = 10011000111, y = 01101, Quotient = 0100101

Remainder = 1011

Test 3

x = 111000111000111000111, y = 101010101010101, Quotient = 1010101

Remainder = 0111000100111

Problem 2

Test 1

x = 01, y = 0101, N = 00101

Result = 001

Test 2

x = 1000110001, y = 1011, N = 0000110001

Result = 1

Test 3

x = 1100110011001100110011001100110011, y = 1100011

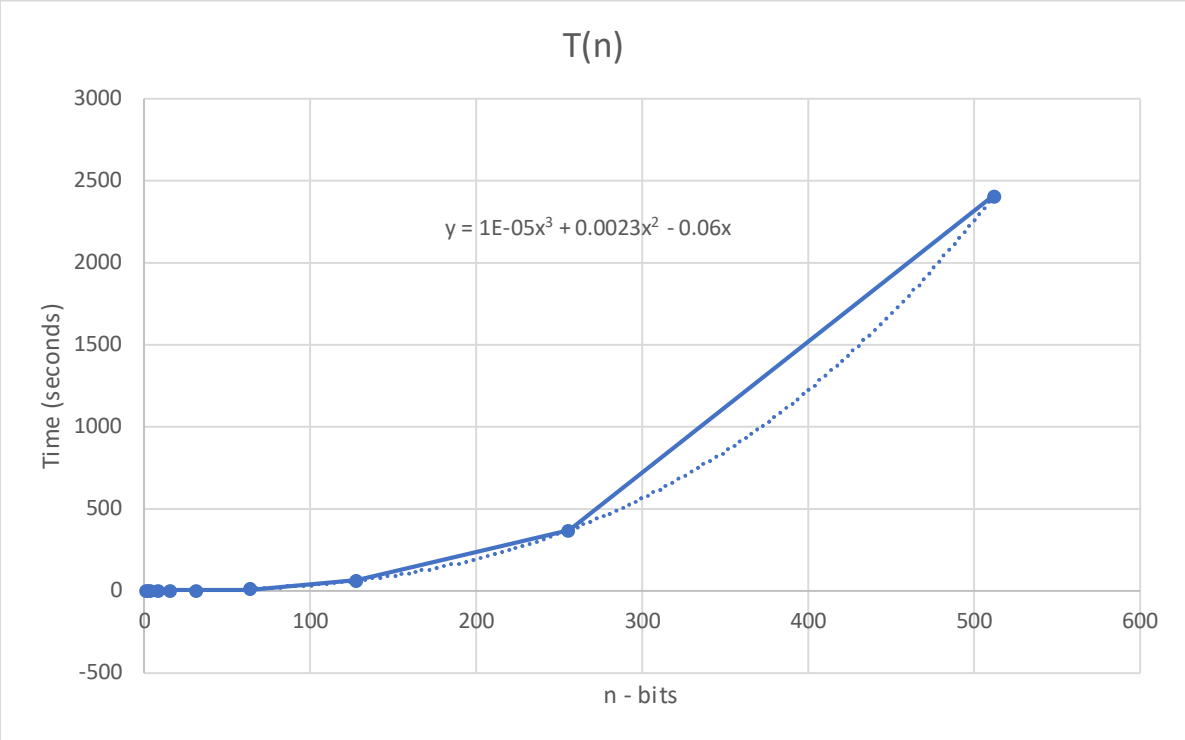
N = 11100011100011100011100011100011111

Result = 01001000111011011000011111111100011

Kipling Gillespie
CS315 - 003

Problem 3

N	Time(seconds)
1	0.00299
2	0.006003
4	0.017002
8	0.051005
16	0.148549
32	0.720124
64	5.18961
128	62.299
256	364.248
512	2404.58



Kipling Gillespie
CS315 - 003

I estimated the next step and found that it would take over 11 hours to complete.

	Forecast
1024	16500