Assignment-1

SM404

Generate Digits of π to any given Precision

Keshav Goyal
IMT2020101
31 March 2022
Instructor: Prof. G.Srinivasaraghavan

Index

Aim

Implement a programme which can calculate the value of π to any given precision and verify this value for its correctness.

Algorithms and References

1. Basic Integer Algorithms: shoup.net
   - Addition
   - Subtraction
   - Multiplication
   - DivMod
2. Division Algorithm: Repeated DivMod
3. Toom-Cook (Toom3) Multiplication:
   - https://en.wikipedia.org/wiki/Toom–Cook_multiplication
   - https://gmplib.org/manual/Toom-3_002dWay-Multiplication
   - https://www.cryptologie.net/article/121/toom-cook-multiplication-for-dummies/
4. Base Conversion Algorithms (In order to get result in decimal form):
   - https://youtube.com/watch?v=R5v3FmG5qus
5. Special Shift Algorithm (To facilitate multiplication and exact division by powers of 2)
6. Newton-Raphson Iterative Algorithm
7. Borwein's exp-2 Algorithm
8. Verification Tools:
   - https://www.rapidtables.com/calc/math/base-calculator.html
   - https://wims.univ-cotedazur.fr/wims/wims.cgi
   - https://ncalculators.com/digital-computation/binary-multiplication-calculator.htm
   - https://text-compare.com
   - https://wordcounter.net/character-count
   - https://apod.nasa.gov/htmltest/gifcity/sqrt2.1mil
   - https://github.com/eneko/Pi

Instructions to run

- Commands:
  - make calculate
  - ./calculate
- Follow the instructions given and input the required numbers in decimal form.

Procedure

- I have implemented these algorithms by utilising the OOP functionality of C++. I have created an *Integer* class which represents any integer of base 2^m. This class was then used as a data member in another class called *Float*. *Float* class represents any rational number(upto finite precision) having base 2^m. All the basic arithmetic and algorithms have then been defined for these classes using methods like operator overloading.

- When run, the programme prompts the user to choose if they want value of π or √2. Then it asks for the required precision and the base to be used for calculations. For calculation of π , user will also get the option of choosing Toom3 as multiplication method. Programme then calculates the value of π or √2 and prints it to stdout in decimal form. The programme then goes on to verify this calculated value using the pre-stored 1 million digit precision value of π as well as for √2. It then informs the user whether the value calculated was correct or not. The total time taken for the whole process is then shown in seconds(Time taken for Calculating in given base, Converting to decimal and finally verifying).

- I have implemented Toom3 algorithm but it hasn't been extensively used in my implementation. This is because both Newton-Raphson method and Borwein's Algorithm only need multiplications with 0.5 (Division by 2). For this simple multiplication was found to be faster than Toom3. Anyway, the option to use Toom3 for calculation has been included in the main menu.

- Anyhow, my Toom3 implementation did give decent results when separately tested for huge multiplications(In the order of 10^5 digits).

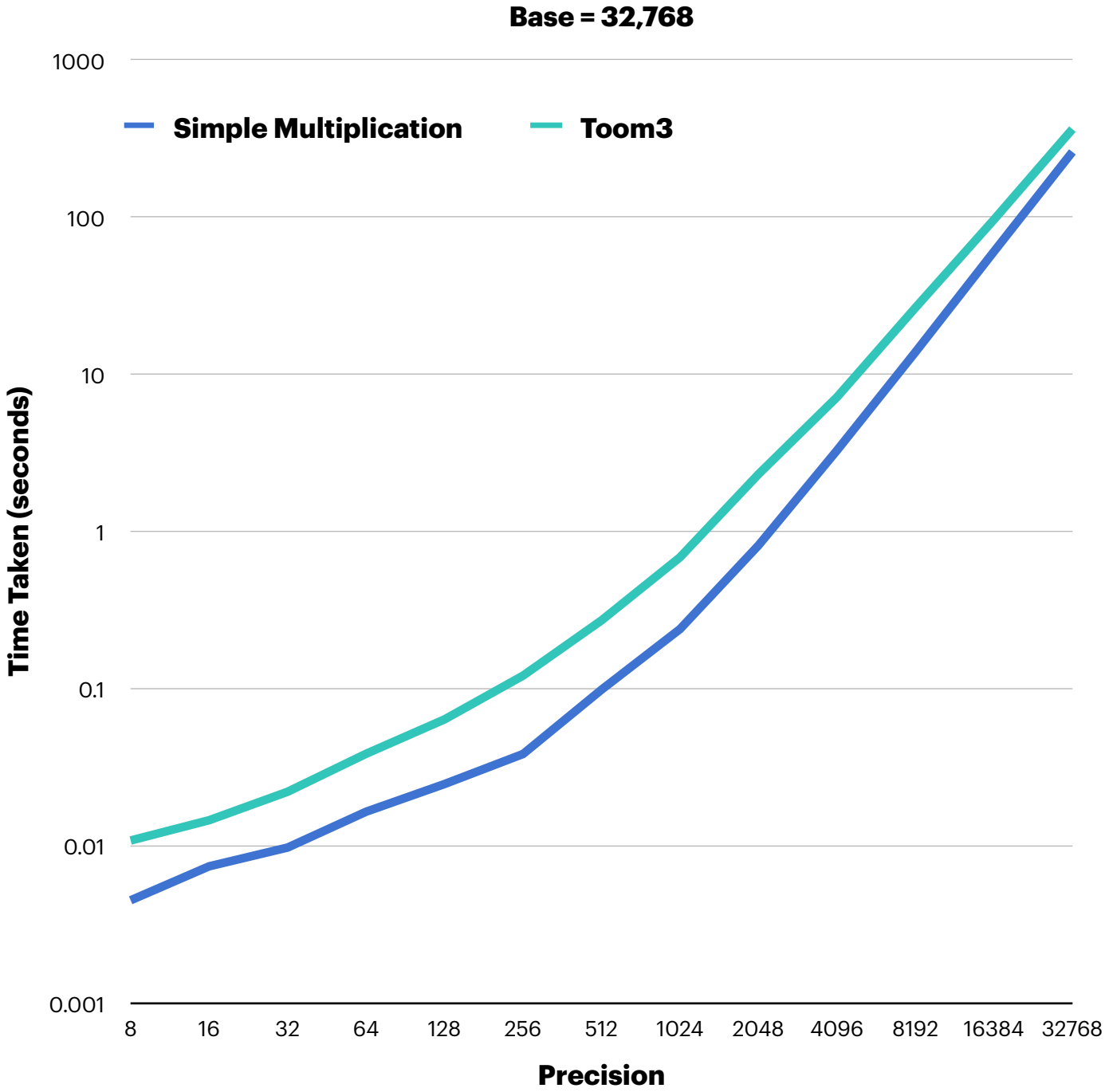
BASE	NUMBER OF DIGITS IN EACH MULTIPLICAND (Each Digit < Base)		TOOM3		SIMPLE MULTIPLICATION	
			TIME TAKEN	STORAGE USED	TIME TAKEN	STORAGE USED
2^8	500,000	500,000	12.32 minutes	40 GB	26.5 minutes	17 MB

- Both, Borwein's Algorithm and Newton-Raphson Algorithm stop iterating when the difference between two consecutive estimates becomes less than the required precision.

- The intermediate results such as square roots and divisions are calculated upto a precision of [(Actual Precision) + 8]. This maintains the accuracy of the digits at higher precision places.

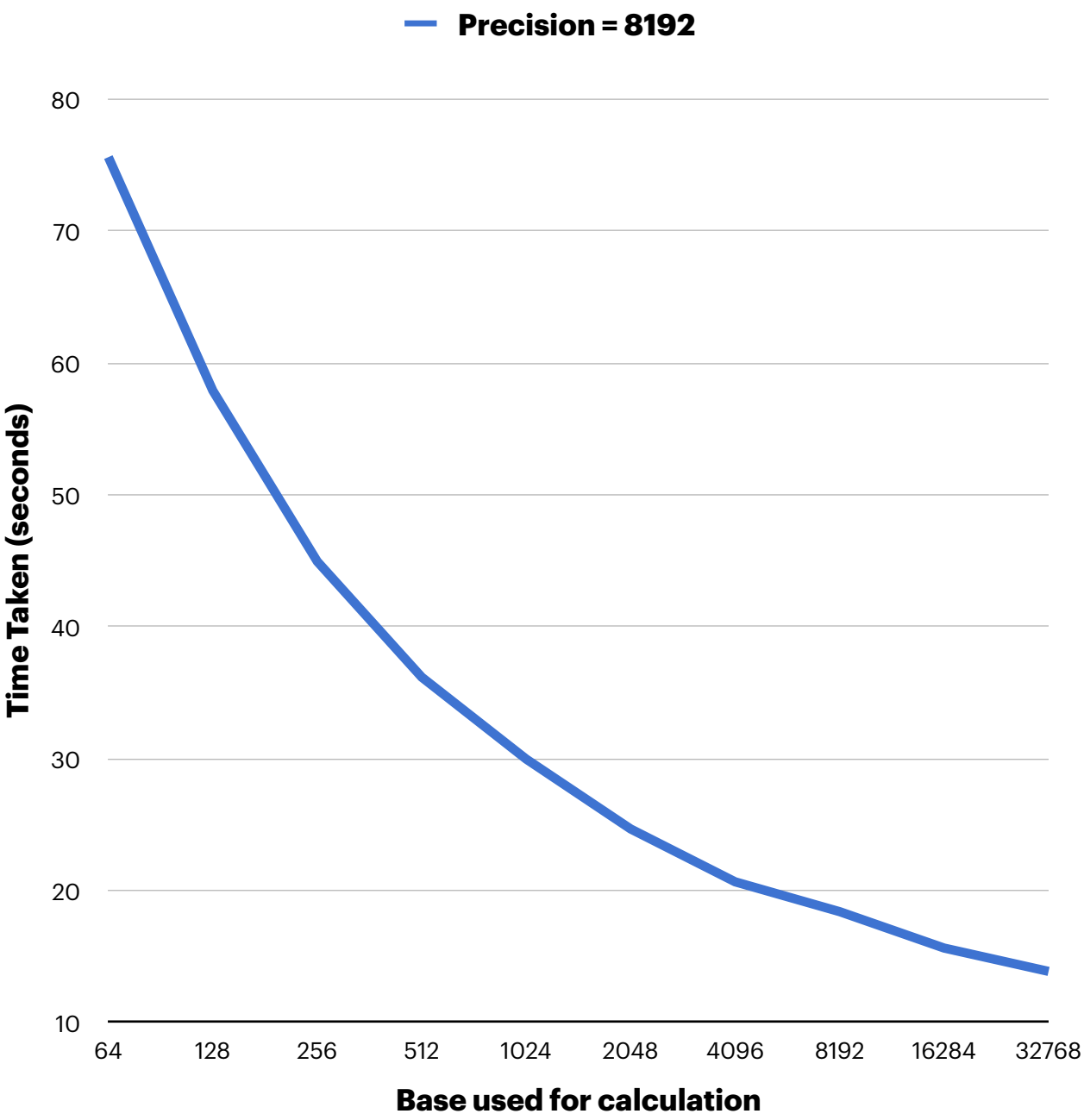
Observations and Analysis

Relation between Precision and Time taken for calculation of π



Thus, we can verify that more precision requires more amount of time. And looking at the trend it is also evident that Toom3 implementation will take lesser time as the precision increases further from 32,768.

Relation between Base and Time taken for calculation of π



Also, as the base used for calculation increases, the time for the process reduces(this effect is significant for larger precisions such as 8192).

Future Scope

- My laptop kills the running programme when the swap utilised by it exceeds 55 GB(This is unalterable in MacOS). To reduce the utilised swap, I can better implement the concept of destructors. This would allow me to get precisions of more than 35,000.
- This would also allow me to use much smaller bases for higher precisions.
- My Toom3 implementation can still be optimised further, which would allow me to incorporate it in the implementation of Newton-Raphson method and Borwein's Algorithm