BSPro-Comparing different variants of gcd algorithms of large integers

Sunday 12th January, 2020 - 19:39

Licina Esada
University of Luxembourg
Email: esada.licina.001@student.uni.lu

Volker Müller
University of Luxembourg
Email: volker.müller@uni.lu

Abstract

This first semester project of BICS, made by the Student Esada Licina and with the support of hers Tutor Volker Müller. This project is about GCD algorithms.In this report will be explain the scientific and technical deliverables.It will be shown all the results and work.

1. Introduction

For years, humans have successfully tried to solve most of the mathematics problems more easily and quickly with computer languages. This project is about having to compare different GCD algorithms to BigIntegers. To better understand the project, the terms **GCD** and **algorithms** are explained.

If going deeper into the single letter of GCD, it creates the words **Greatest Common Divisor**. This composition means that the project is looking for the highest number that allows to share each of the multiple integers, that are not zero. Here is an example with numbers, the largest number that can be divide with 20 and 15 is 5.

Now comes the term **algorithms**, here one understands a way to find the solution for problems that go in single steps and it only stops until it has the result. Such algorithms end at a given length. Algorithms also have a certain syntax, that means, they have their own strict way of logically building in certain programming languages.

In this project will be to find and compare several ways to solve mathematical calculations more effectively and faster. This ways for solutions are GCD algorithms.It will also compare who has the better power of them with BigInteger.This project has five different types of algorithms the Euclidean algorithm, the extended Euclidean algorithm, the Binary algorithm (Steins algorithm), the extended Binary algorithm and the gcd BigInteger algorithm that was created by Java.

It will also have a practical work where the programming part of the different algorithms in NetBeans (Java) will be, to better observe how they works. To be more focus on the aim she will use Excel to print the time of the algorithms and made a time graph of them. The practical work is just focusing in NetBeans (Java) and Excel. After this it will be explain the scientific and technical deliverable. On the scientific part it's more the speech about the performances and on the technical part more about the codes.

2. Project description

2.1. Domains

The aim of this project is to compare the different algorithm in the programming languages Java and to see the real duration of them in NetBeans. The duration will be print in Excel to have a better look on the different time.

2.1.1. Scientific.

1) **Java:** Many well-known software's were programmed with Java like Minecraft and GeoGebra. The programming languages was published in 1995 by the company Sun Microsysteme and developed by James Gosling. Java is object oriented, distributed, multithreaded, robust and secure.With Java all alone, the project can not achieve much just the basics,it has to use also NetBenas (programmed by Java) to program a more efficient game or app. Minecraft was also programmed with other helpful technological like the program library Lightweight Java Game Library (LWJGL) and Java Runtime environment of the version 6.
2) **NetBeans:** Facilitates the creation of Java Swing application and other programs.NetBeans (also called NetBeans IDE) is a adaptive development process and is a open sources model. It was a simple project named as Xelfi of some students in the Prague university that started 1996 and after it grew to NetBeans that was bought at the company Sun Microsysteme. After some time NetBeans was taken by Oracle and was extended also by Apache Software Foundation and Oracle Corporation.The NetBeans IDE supports Java,C++,PHP,Java Script and other programming languages.In this project will be used NetBeans with the Java language to program the algorithms.
3) **MS Excel:** This spreadsheet program was developed by Microsoft for windows, mac OS, Androids and iOS.Also by Microsoft developed are Word and PowerPoint.Excel

has about 471 different functions and the user can write function on his own with the VBA (Visual Basic for Applications).VBA is an implementation of Microsoft's event-driven programming language. The first Excel 1.0 was introduced as a graphically oriented spreadsheet in 1985.Excel was written by C#, C++, .NET Framework and Microsoft Foundation Classes. It has a useful function where we can put grids of text and numbers to create a diagram.

2.1.2. Technical. Now knowing what the different apps/programs means and what they do,it will be explain in the technical domains what exactly was done that the project run. The goal of the programming part is to have in the end a formal well understanding code that can be easily use for find the gcd of big numbers. To make it more visual it will be shown everything on the NetBeans and Excel windows.

1) **Java classes:** Java has 3 classes, one is BigInteger, where the user can give huge numbers as input,the second is only with integer where the user can enter smaller numbers and the third one where are codes to make a sentence with the results which will be print in excel. The four algorithms have been programmed in each of the two classes and the Java BigInteger gcd algorithm was just programmed in the first class.It was necessary to make the algorithm one time in BigIntegers and one time in Integer, because the bit size of Integers is 32 and that's not enough . So it is needed to program BigInteger that it is possible to work with enormous numbers with no limit of the bit, with this two types it will be compare if the code gives the right results back.When the user takes small numbers then he can calculate with his mind and prove if the result is true then it will be also be true for big number.
2) **NetBeans MainFrame:** The results of the algorithms are shown in NetBeans.Then there's the mainframe where can be create a virtual design and see the GCD as well as the running time of the algorithm. There are 9 buttons for the nine different algorithms, then two button for delete the data, two lists where the results come from, two buttons for clearing the lists, one textfield where the user can choose the number of digits of the random numbers,two other textfields one for the first number and one for the second number and two buttons to save an load the excel file. The window will be also a little bit colored, to have a more interest shine because a world without color a unhappy,bored world. By NetBeans source will be add the three classes and than write a code for the function of the buttons and textfields.The code of the algorithm will be run many time through a loop to have a better result of the time and put it on the list ,this code will be programmed on the algorithms buttons. The lists shows all the results of the tries.In the textfields will be programmed the random code and connect it with the algorithms.
3) **MS Excel sheet:** After finished the Java/NetBeans toy, the data will be saved on Excel.So all the results are showed in the Excel table and the user can easily make graphs or diagrams where he can better see the power of the different algorithms. This graphs or diagrams will be explain better on the scientific part.

2.2. Targeted Deliverables

2.2.1. Scientific deliverable. Scientific presentation to comparison of performance of the several gcd algorithms:

A diagram shows the time of the algorithms.Whit this help it is easier to see the fastest gcd algorithms.So after that it can be explain for what reason it has the best power.

2.2.2. Technical deliverable. A NetBeans/Java program to calculate the gcd of big numbers:

It will be explain how the main codes works and some pictures of the codes will be put inside.Also the design of the NetBeans window will be showed.

3. Prerequisites

To start this project it is necessary to have some knowledge before. So it is more easy to understand what exactly is behind this codes and mathematical problem.

It will have two subtitles one are the scientific prerequisites, where will be put a lot of information/knowledge in the mind. After that comes the technical prerequisites where the scientifically part will be used in a practical way.

3.1. Scientific prerequisites

Eight aspects will be showed that need to be done before beginning the practical part.

1) Make researches about the different gcd algorithms and to go behind the mathematical world. All gcd algorithm is special in their own way.There are not difficult maths formulas for the gcd, the most are basis types.
2) It is necessary to have programs with which are worked and then knowing how to install the different programs Excel, NetBeans and Java and which types, before going deeper on them.
3) Before starting with coding it is important to know how Java work and to understand which function are needed to make classes and lists.
4) Understanding NetBeans will be easier, after knowing Java because they work together.In NetBeans it will be helpful to know how to use Mainframe with the buttons and lists.
5) After the knowledge of NetBeans and Java,it is time to start with the codes.How to write mathematical formulas in computer language.
6) In this project it will be necessary to use the BigInteger function.This function makes it possible to work with enormous numbers.So knowing how to replace the

Integer by the function BigInteger is important and to understand the different methods that are the best to use.
7) Then knowing Excel will be helpful, to get a diagram with the results.To know have to make graphs/diagrams and which possibilities it has.
8) After knowing the basis of Excel,it will be interesting to have information about the connection with NetBeans. To get the results ordered and clearly in a Excel sheet. The are other helpful function which are needed to understand before using.

Without this information's, it will be very difficult to start the project and to get finished on the right time.

3.2. Technical prerequisites

Here are the technical prerequisites, now it will going deeper on the practical side.

It is time to use the several information to produce something beneficial.At first it is important to install all the programs.Then to have experience to coding in Java, that's the most significant part because there without the algorithms the Project has no results to compare.After this comes the using of NetBeans and to create a virtual window with all useful functions like buttons.Knowing to make diagrams and to order all the data is the beginning to start with the scientific deliverable.For this it is important to know using Excel, but to make the diagrams it is necessary to write a code in NetBeans to connect them together.

4. Scientific presentation to comparison of performance of several gcd algorithms

The deliverable is useful for some actors like a Scientist that wants to know why the algorithms has a difference with the time.

4.1. Functional Requirements

Functional requirements define a behaviour or function that the end product is expected to do.

4.1.1. Calculation of the time. The final product needs to have the function to calculate automatically the time of the algorithm's paths. NetBeans has a function which calculate on nanoseconds (ns), ns is a good time unit for the algorithms because some of them are fast and maybe with milliseconds the time can be 0 or equal to a other algorithms then it will be difficult to say which of them has the best time. The end product need to let run the algorithm many times to get a better result of the duration and after take the average of them.

4.1.2. Shows the duration of the algorithms. The goal of this project is to compare the algorithms and in the end to take the algorithms with the best time results.To investigate the power, the time will be showed on the NetBeans window with the help of the code. So that is the task of the final product,after compiling and running the project.

4.1.3. Add the time of the algorithms on excel. After having the time of the different algorithms. The final product should transport the data with a button (save button) click on a Excel sheet.That's very important because of that it is possible to make diagrams and to see better the power of the algorithms.With the help of codes it makes more easier to save and move the results on Excel.

4.2. Non-Functional Requirements

The non-functional requirements are properties of the end product to make it more comfortable to use.

4.2.1. User interface of NetBeans window. That's a good property of the project, it makes it more easier to see the results of the gcd algorithms and it is more readable.After clicked the buttons it comes a sentence where the algorithms name is written , the random numbers,gcd and the time.

4.2.2. User interface of the Excel sheet. The property to have the data of NetBeans easily on a Excel sheet with a button (save button) click is amazing.So it is not necessary to write the time of the algorithms with the hands,that will be slower.It will also be automatically ordered in columns with titles.So starting to create the diagram can be faster done.With all the functions of Excel it will be more understandable and easier.

4.3. Design

Here will be explain the design where the deliverable was done,all the little details about the colors and domains.The design of a product is prominent.It gives something special back, it is the first meeting with the user.The first appearance need to be amazing and to tempt somebody to use the product.The Design will be focused on the Excel sheet where the diagram is set with the columns of time,gcd,digits,numbers and the name of the algorithms.But to begin first it is necessary to say from where the data in excel comes from.

4.3.1. NetBeans. The details of the NetBeans design will be explain on the technical part because the technical deliverable is more specified on NetBeans.Here in this part will be just showed from where the time of the algorithms comes and where it appears.

The program has a code in MainFrame sources for all algorithms button,to calculate the time. The code will run 10000 times for each algorithm to get a better result.The time will be in nanosecond because some algorithm are fast and to take a bigger time unit will not be useful.

After that it will be printed on the NetBeans window with help of the lists, where the algorithms are inside.The window has a little white window for the BigInteger gcd algorithms where the results are printed with a button click.But at first the user need to put on the TextField the length of the numbers that will be chosen randomly from the program. If the user

choose the Euclidean gcd algorithms then it will be printed and it looks like this **The Euclidean GCDalgorithm: 5876,0932 ns for the result 4 with the BigIntegers 9038570076 & 6291289288** .That's a understandable sentence where is everything included.

When the user has tested all the algorithms, he can start to to click on the Excel save button .So he can save his sentence in a almost perfect order in a Excel sheet.He gives a name for the sheet and after he finished with saving, it will come a massage window to say that it was successfully saved.

4.3.2. Excel. In this part will be showed two images from the beginning to the end product in the Excel sheet.

	A	B	C	D	E
1	Algorithms	First_number	Second_number	gcd	Time
2	The ExtendedEuclidean GCDalgorithm	64227	51287	1	7270.99
3	The Euclidean GCDalgorithm	93338	63707	7	3984.8

Fig. 1. Excel window of the results

So that's the beginning when the user open the Excel sheet.How the user can see it will have 5 columns with the name of the algorithm, the first number, the second number,the result and the time.So the user don't need to do much to change things.To get the data in a order,it has a extra code in the class Result.The numbers by the time column are now integer and not any more decimals, Excel remove automatically the decimal places.But to make a diagram and to make it more colored the user need to do it on his own.

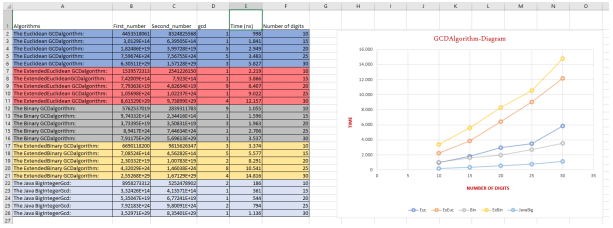

Fig. 2. Excel sheet

The final product will be like on the picture but that depends of the user how he wont to create it.How you can see here are lot of colors and it is intelligible.With a right click on the excel sheet,the user cans do all lot of things and above are lot of function. The colors of the sentences on the left side are connected with the diagram to know which of the algorithms shows the lines on the right side.Excel has a lot of different types of diagram.Here was chosen the Line diagram with the data points, it is more visible because the points shows which data of the y-axis are connected with the data of the x-axis.

As seen in the picture, it is one column more with the number of digits, like this the diagram is more understandable.The user can see how the time goes higher or lower with the length of the numbers.On the y-axis is the time and on the x-axis is the number of digits.There are 3 title , the diagram title and the axis titles. Below of the diagram is the legend, that means it shows the name of the lines.

The power of the algorithms and find out which is the fastest, will be showed on the presentation part.The Excel sheet is a important part of the end product, with this the project is more complete and the main goal is reached.The user can compare easily the different gcd algorithm.

4.4. Production

Here comes the most import information.It is like a answer for the scientific deliverable.Here will be showed which of the algorithms has the most power and what is the reason for the difference.

For the binning it will be showed the code of the time calculation and explain why the code is programmed like this. The second part is to compare the algorithms with help of the diagram.

4.4.1. Time calculation. This will be a little explanation with the help of the code below.It will be explain why the code use is like this.

```
1 double sum = 0;
2 for (int i = 0; i <= 10000; i++) {
    3 long start = System.nanoTime();
    4 BigIntegerGCD.gcdEucBig(a, b);
    5 long end = System.nanoTime();
    6 long diff = end - start;
    7 sum += diff;
  }
```

Fig. 3. Code of the time calculation

It are 7 steps to explain:

1) The variable **sum** will be useful on the end to get the final time. **Sum** is double because time is often a decimal number and after that **sum** will be divided.It is also for a more detailed result of the time.
2) Now begins the **for loop**.How seeing on the picture it will run 10000 times.The reason is that each time when the algorithm run comes a other duration as result and because of that it will run more then one time.
3) Here the code begins to calculate the duration.And this result will be put on a variable **start**.The code takes **long** because the result can have up to 64 bit more then a integer.It will be using for the time unit nanosecond, because some algorithm are faster and so the project get a good time result.
4) The numbers will be put on the algorithm and it start to calculate the gcd.
5) Then it calculated once again the duration on the end.The code makes this because it is necessary.The start time calculate the time from the program itself and the **end** time calculate the program and the algorithm time.
6) So it is needed to subtract the **start** from the **end** to get just the algorithm duration.This will be printed on the variable **diff**.

7) Each time when the **for loop** ends, it will be add the time difference **diff** to the variable **sum**, because in the end the code need to divide the **sum** by 10000 to get the average of all the times.

4.4.2. Comparing gcd algorithms. It will be explain the position of each algorithm and why some are more faster then other.

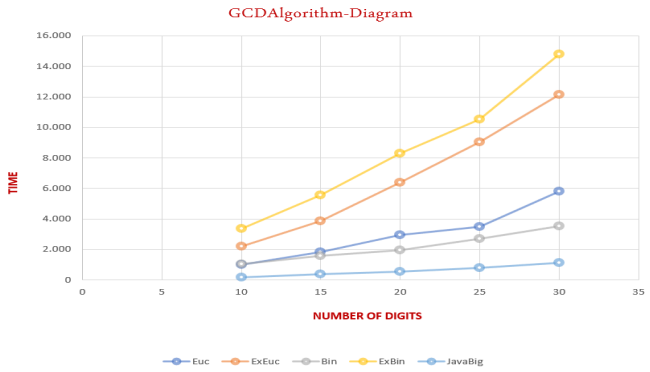

Fig. 4. Excel Diagram

The first line is the **extended Binary** gcd algorithm and he is the slowest algorithm.He begins with the highest time an end with it.The highest line is also the slowest one because the time and the number of digits rise. Then second one is the **extended Euclidean** algorithm and he is the second slowest one.After him comes the **Euclidean algorithm**. Finally the two fastest are the **JavaBigInteger** gcd algorithm and the **Binary** gcd algorithm. So **JavaBigInteger** is more faster then the **Binary**.The diagram shows that when the length of the number rise then the time rise also, they are proportional.

The **JavaBigInteger** algorithm is written by Java and so the project doesn't have the real code of them, also the researches were not successful.The assumption is that maybe it is the fastest because it is no code needed, just insert the function in a method and it will do everything alone.With the other algorithms that are written by a person it is different because Java needs to understand what the way of this algorithm is and then begin to calculate.The **JavaBigInteger** algorithm is already inside the program.Maybe the algorithm is similar as the **Binary** algorithm because it is the second fastest.It can also be that it is a combination of the **Binary** and the Schönhage-Strassen multiplication algorithm because with some researches it is more faster then the **Binary** gcd algorithm.But it is no proof there and no explanation for this assumption.

The **Binary** gcd algorithm works with the binary system.That means the numbers can be just written with ones and zeros.For example the number 2 is written as '10' in binary way.The binary system begins with '0' for the number zero,'1' for the number one, '10' for the number 2 , '11' for the number 3 and '100' for the number 4.So as seen it replace every time the 0 by 1 and when the whole number is written with ones then it will get one bit more.So the algorithm has some codes that helps to calculate the binary side of a number and because of this system is the algorithm faster then the other algorithms.For example the **Euclidean** algorithm is using the **modulo operation** to divide two numbers but here it is just needed to divide the numbers by two with help of the **right shift operation**.This code (**number.shiftRight(1)**) is used to divide the number which moves the bits (that are the digits of the binary number) one position to the right side.One more information is that the user cans just put unsigned BigIntegers , so the code need to take the absolute value of the numbers.The reason for this statement is that it works with binary system and when there is a negative sign then the system get confused.The **Euclidean** algorithms can have negative numbers, that's no problem for the code to calculate the gcd.

So the **Euclidean algorithm** is faster then the **extended Binary** because it has the simplest code.He doesn't need to calculate much just using the **modulo operation** .After comes the **extended Euclidean** algorithm.He is faster then the **extended Binary** algorithm because the **Binary** is more complex on the code, he need to calculate with 4 variable to get the gcd, so he needs more longer to get the result.He works also just with unsigned numbers.The **extended Euclidean** has a more simple construction and he use just 2 variable to calculate the gcd.Also the way of the code is not so long is a little code.The codes of the algorithms will be better explain on the technical part, with some codes images to understand the difference.

4.5. Assessment

The time is coming to see if the requirements are successfully met or not.The first requirement was the calculation of the time.This task was successfully completed.How seeing on the scientific production , the code has given a good result of the time because on the diagram the user can see the difference between the algorithm.Every step of the code was explained and some function of the steps was given from Java, like the function System.nanoTime() that calculated directly the the time in nanosecond. Then the second one was to show the duration of the algorithm, but the report doesn't have any picture of this. But when the project doesn't get the time then the diagram on the scientific part will not exist but the proof is there.With a button click the user gets a clearly sentence on the NetBeans window. The third one was to show the algorithm in a Excel sheet.Taking a look on scientific design and description, there are seeing two pictures,one of the columns with the results and one of the diagram with the different time of the algorithms.There are grateful results with a nice column order. Also the non-functional requirements are successfully used.NetBeans and Excel have helped a lot make the project more easier and understandable.With all the button properties and also the choices that the user have to create something on his own style. How seeing everything that was planned was successfully reached without problems because

NetBeans and Java work good together.Also the work with Excel is amazing because without Excel the report wasn't going to get a diagram.The scientific part will take more longer time to get finished and it will not look professional. The three programs are a good combination to work with algorithms and to make a whole project of them.

5. A NetBeans/Java program to calculate the gcd of big numbers

This deliverable will be useful for teacher, students and for other people who wants to get the gcd of two numbers.For this it exist the integer part where you can input your own two numbers but just until 32 bit.

5.1. Functional Requirements

5.1.1. Calculation of the random numbers. To make after that a diagram it is better to have the two big numbers with the same number of digits, that are an input of the user.But the big numbers are selected at random from the product.This is possible with a code.It is more efficient that the program choose the numbers and so focusing better on the results and comparing them.This code is a little bit confused because it works with binary system.

5.1.2. Calculation of the greatest common divisor. That's the main task without this the final product can't do nothing.The product has 5 codes of the gcd algorithms in a BigInteger class that need to give the right result back.The codes take always the absolute value of the numbers to calculate the gcd without this function it wills give a wrong gcd back.This codes will also be better present on the presentation part.It is just important that the code is right.That's why the product have a second class with Inetger where the user can input small numbers. The end product needs just to calculate the gcd and put it on the window.

5.2. Non-Functional Requirements

5.2.1. The Textfield of the number of digits. This is a property of the final product that makes the life more simple.The user don't need to think about the numbers, he needs just to give the desired length from the number and the program can start. So he checking directly the results and can begin to work with them.It is good that NetBEans has this possibility and function.

5.2.2. The availability of functions. NetBeans has a lot of useful properties, he has a lot of function that are very helpfully.His qualities are to saves the user a lot of work.He has the time function and the MainFrame window with all the properties for example to put icons to understand better what the button means.The program connects things together like buttons and lists. With the buttons the user cans remove and add things on the lists.It has many possibilities, the user reach a lot with simple clicks and moves with the mouse.This functions and properties are fantastic to programming the algorithm , to calculate the gcd and to focus on the end product.

5.3. Design

The time is coming to talk about the NetBeans design.This part is just focusing on the window and the little bit of the different classes of NetBeans.It will be showed two pictures one about the classes and one of the virtual window.

5.3.1. NetBeans construction. This subsection will show where are all the codes and why they are in such order.

Fig. 5. NetBeans with classes and MainFrame

There are 4 windows (3 classes and one MainFrame).

The first class is for the gcd algorithms with the function BigInteger.This codes are not directly usable because there are no values.It needs to be programmed a code in MainFrame which gives the values back to the algorithm.The class has 5 methods where the 5 algorithms are programmed.

Second class is for the gcd algorithm with Integers.They are just to testing if the algorithms gives the right result back or not.The user can it also take to calculate the gcd if he don't have time to do it on his mind.Here is the same thing with the codes like in the first class, they need a code in NetBeans for the values.This class has 4 methods.

In third class are codes for the help in Excel.That class is important because this codes make the order in Excel when the data get saved.In this class are just 2 method, one constructor and 3 attributes.

The forth class is one of the influential parts of the program.Without this the program won't work.There are all codes that makes project run.In Mainframe is the virtual window which we use to get the result and to save everything in Excel.There are a lot of codes that works together and all the classes need to be connected with MainFrame.It is like a mother with his three children.

5.3.2. NetBeans window construction. The window is like a tool, it has many option and things to do.There is a line which separate the Integer from the BigInteger, the two part has also titles.If the user wants to do a such window, he just need to go by JFrame and click on Design.

On the above Integer part of the picture are 4 buttons with 4 different colors for each gcd algorithm.The buttons name are abbreviations of the names of the respective algorithm.When the user clicks two time on one button before the project run then he will be led on the source window where the button code is.Below is present the white window also named

jList for the inputs of the results.Here the user can try and calculate every integer number that has less then 32 bit.To get something on the white board the user need to input a number1 and a number2 at first. Then he can click one of the algorithms buttons and he get a sentence same as on Excel.The program makes the live more easier with the remove and clear buttons.On the buttons are two little images (icons) which are taken from the internet to make the button easier to understand.The remove button take the selected data away, but if the user has nothing selected then it will appear a message window that say 'You did not select any element to remove'.Then it exists also the clear buttons and this is useful if the user wants to delete everything from the jList and to use it again.

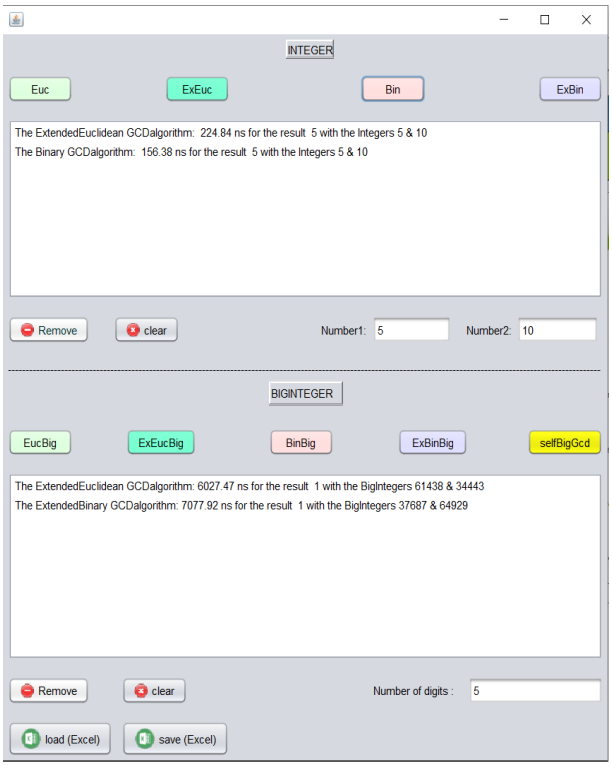

Fig. 6. NetBeans window design

Now comes the explanation of the below BigInteger part. There are 5 buttons with colors.The colors mean that it is the same type of algorithm like above.The yellow button is a algorithm that are written from Java but this gcd algorithm exists just for BigIntegers. Then there is also a jList window that is named as viewList. It has the same function like above and the two buttons, remove and clear. But there are three new thing, one is the 'textfield' of the number of digits and two buttons for Excel.In the textfield the user can input the length of the numbers, the two numbers will have the same length but different values because thy will be chosen randomly of the program.So it will be more easier to focus on the duration of the algorithm and to create the diagram.The window has also a load and save button also with two little images of Excel.If the user clicks the save button, it appears a window where he can save his Excel file. After saving it will comes one more message window to say that the file was successfully saved.The load button is not so important. Clicking on the button means the file that the user saved can bring the data back on the viewList.If one time the user wants to look on the data or to change something.

5.4. Production

5.4.1. Euclidean gcd algorithm. It will start with the easiest algorithm.

```
a=a.abs();
b=b.abs();
if(b.equals(BigInteger.ZERO))
{
    return a;
}
else {
    return gcdEucBig(b,a.mod(b));
}
```

Fig. 7. Euclidean code

At first the code takes the absolute value of the inputs **a** and **b**.Then he looks with the **if** function that **b** equals to zero.If the code returns true, he gives as gcd **a** back,because everything can divide zero.If the the answer is false then the code takes his own the method and replace the **a** by **b** and **b** by **a.mod(b)**, that's the famous **modulo operation**.When a method use it's own method on the code then it is a **recursive method**.So that means that by **else** the code will be repeat with other values of **a** and **b**.The module is a operation that divides two numbers and gives the remainder back. The code repeat it until **b** get zero and the algorithm return as gcd the new number **a**.

5.4.2. Extended Euclidean gcd algorithm. The Extended algorithm is similar just he gets help of two variable **x** an **y**.

```
if (n1.equals(BigInteger.ZERO))
{
    x =BigInteger.ZERO;
    y = BigInteger.ONE;
    return n2;
}
```

Fig. 8. Extended Euclidean code part1

At first the absolute value is taken from the input numbers **n1** and **n2** and the two variables will have as value 1 because **x** is a variable of **n1** and **y** for **n2**.It is like **x** multiply **n1** and **y** multiply **n2**.The code start to look if **n1** is equal to zero. If the answer is true the variable **x** gets the value 0 because 0 multiple **n1** is zero and **y** gets the value 1 because **n2** multiply by 1 stay **n2**.

```
BigInteger x1=BigInteger.ONE;
BigInteger y1=BigInteger.ONE;
BigInteger gcd = gcdExEucBig(n2.mod(n1), n1, x1, y1);
x = y1.subtract((n2.divide(n1)).multiply(x1));
y = x1;

return gcd;
```

Fig. 9. Extended Euclidean code part2

If the answer is false , the code take his own method **(recursive method)** and replace **n1** by **n2.mod(n1)** and **n2** by **n1**.The method gets also two new variable **x1** and **y1** that have on the beginning the value 1.If the code start to repeat the method and finds new values for **x1** and **y1** then he can calculate with them **x** and **y**. It is important to get a value for **x** and **y** because the gcd will be calculate like this: $n1 \cdot x + n2 \cdot y$ =gcd. That are the main explication, but it is not easy to understand without example.The user will better understand if he takes two numbers and try to find the gcd with this way.

5.4.3. Binary gcd algorithm. This algorithm will have three parts to explain.

```
int k=0;
while((!(a.testBit(0))) && (!(b.testBit(0))))
{
    a=a.shiftRight(1);
    b=b.shiftRight(1);

    k++;
}
```

Fig. 10. Binary code part 1

At first the code checks if one of the input numbers **a** or **b** is equal to zero.If no he starts with the **while loop** and looks if the two numbers are even.There exist a special function for binary numbers.The **testBit(0)** function checks if on the position 0 of the binary number sets the number 1.If this is true then it means the numbers are odd and if it is false then they are even.If they are even, the numbers will be divide by 2 with the **right shift operation**.The code has also a variable **k** that will be very important in the ending.Every time when the **while loop** run it will be add 1 by **k**.

```
while (!(a.testBit(0)))
{
a=a.shiftRight(1);
}
```

Fig. 11. Binary code part 2

If one of the numbers is odd then the **while loop** stops running and goes by the other **while loop**. This loop checks if a is even or odd. If it is true(even), **a** will be divided by 2 until **a** gets odd.The **while loop** will stops when it gets false(odd).

So on the picture below is a **do-while loop**, there is no big difference between **while** and **do-while**.The only difference is that the condition is below and if the condition is false the code will run at least one time and then stop.By the **while loop** it will stop directly if the condition is false.In the **do-while loop** is one more **while loop** that checks if **b** is even is true then **b** will be divide by two until he gets odd.Then after leaving the **while loop**, there is one **if** function that checks if **a** is bigger then **b**.If true then **a** and **b** will be swapped because after that **b** need to be subtract by **a**.The subtraction is important because when the code takes the difference of two odd numbers he get a even number and so **b** cans go on the **while loop**.Now the code looks the condition of the **do-while loop**.If **b** is not equal to zero, the code stay on the **do-while loop** if **b** is zero the code leaves the **do-while loop**.In the end **a** will be multiply by the variable **k** with help of the **left shift operation**.**k** has calculated how much the two numbers on the first **while loop** are divided by 2. The last code row is the gcd. That's how the code works.The goal is that **b** gets zero and **a** gets multiply by **k**.

```
do {
    while (!(b.testBit(0)))
    {
        b=b.shiftRight(1);
    }
    if (a.compareTo(b)>0)
    {
        BigInteger swap = a;
        a = b;
        b= swap;
    }

    b = b.subtract(a);
}
while (!b.equals(BigInteger.ZERO));

return  a.shiftLeft(k);
```

Fig. 12. Binary code part 3

5.4.4. Extended Binary gcd algorithm. This algorithm is similar as the above.

```
while((!(a.testBit(0))) && (!(b.testBit(0))))
{
    a=a.shiftRight(1);
    b=b.shiftRight(1);
    shift++;
}

ao=a;
bo=b;

sa=BigInteger.ONE;
sb=BigInteger.ZERO;
ta=BigInteger.ZERO;
tb=BigInteger.ONE;

while(!(a.testBit(0)))
{
    if((!(sa.testBit(0))) && (!(sb.testBit(0))))
    {
        sa=sa.add(bo);
        sb=sb.subtract(ao);
    }
    a=a.shiftRight(1);
    sa=sa.shiftRight(1);
    sb=sb.shiftRight(1);
}
```

Fig. 13. Extended Binary code part 1

The difference here is that this algorithm has 4 variable.The code has two formula for the input number **a** and **b**. The first is $sa \cdot ao + sb \cdot bo$=a and $ta \cdot ao + tb \cdot bo$=b and with this two the user will understand better the code.The code has 3 possibilities the first one is that **a** is even and **b** is odd, the

second is that **b** is even and **a** is odd and the last one is when the two numbers are odd.So the beginning of the code is similar like the above.So the code needs to look with the variable how to get the 3 different possibilities. So when **a** is even and divisible by 2 then the code knows that **sa** and **sb** are also divisible by 2.

```
while(!b.equals(BigInteger.ZERO))
{
    while(!(b.testBit(0)))
    {
        if((!(ta.testBit(0))) && (!(tb.testBit(0))))
        {
            ta=ta.add(bo);
            tb=tb.subtract(ao);
        }
        b=b.shiftRight(1);
        ta=ta.shiftRight(1);
        tb=tb.shiftRight(1);
    }
    if(a.compareTo(b)>0 )
    {
        BigInteger t=a;
        a=b;
        b=t;

        t=sa;
        sa=ta;
        ta=t;

        t=sb;
        sb=tb;
        tb=t;
    }
    b=b.subtract(a);
    ta=ta.subtract(sa);
    tb=tb.subtract(sb);
}
return a.shiftLeft(shift);
```

Fig. 14. Extended Binary code part 2

Here in part 2 is also nothing new. Is everything the same just this algorithm has 4 variable.But this code is also not so important because it is the slowest one.The goal is to look how to write the variables when **a** and **b** are in one of the three possibilities and that **b** needs to get 0 on the end.**a** will be just multiply by **shift**,that's the same like the **k** by the Binary algorithm.

5.4.5. Code to calculate the random numbers. This code will be explain because it is a special code.

```
int x = Integer.valueOf(BigNumTextField.getText());

BigInteger max = (BigInteger) (BigInteger.TEN.pow(x)).subtract(BigInteger.ONE);
BigInteger min = (BigInteger) (BigInteger.TEN.pow(x - 1));
BigInteger delta = max.subtract(min);
Random rnd = new Random();
BigInteger n1 = new BigInteger(delta.bitLength(), rnd);

do {
    n1 = new BigInteger(delta.bitLength(), rnd);
} while (n1.compareTo(delta) > 0);

return n1.add(min);
```

Fig. 15. Random code

At first the code takes the input length from the user with the function **BigNumTextField.getText()**.Then the code put the integer number in a variable **x**. After the **maximum** and **minimum** will be calculated for the random function.For the **max**, the code takes the formula ' 10^x-1 ' and for **min** the formula is ' 10^{x-1}.So if the user puts the number 4 for the length then **max** is 9999 and **min** is 1000.Then the code calculate the difference of **max** and **min** to get a new **delta** **max**.So there is new limit.Then the code calculate the random number **n1** with the **delta** bit length for the max and with the **random** function.So now the code needs to check if **n1** is bigger then **delta**,if it is true then it will be calculate a new **n1**.If **n1** is smaller then **delta** then the **loop** will stop and the code will return **n1+min**. The length by binary numbers is different because they works with 0 and 1.The code need to choose a such way because BigInteger has no standard random function that calculate with **min** and **max**.

5.5. Assessment

All the requirement are successfully met.The first one was the calculation of the random number.When the user put a length for the number on the textfield, the code gave him to random numbers back.That was a good code because with them the user can easier make a diagram and he didn't need to think about a big number.The second one was the calculation of the gcd.On the technical production are some images of the algorithm and also a explanation of the codes how they work.The project is being a good work.Then main task works and the user cans begin to calculate the gcd because without the algorithms the project can't start.But how seeing the codes gives good results back without any problem.

The non-functional requirements were very helpfully.All the buttons on NetBeans have a significant role on the program.Without one of them buttons the program will not be complete.The buttons, textfields and list are also easy to use and the user can also make colors or pictures on them.It is recommended to work with NetBeans and how the most people know, many famous programs are programmed by using NetBeans.With NetBeans the people can make wonderful things, they need just to have some creativity. Here is not much to say because everything was going good and the project is finish now.

Acknowledgment

I,Esada Licina would to thank my project tutor (pat) Volker Müller for this inspired idea with the algorithm.Thanks a lot for his time and help.With his help I get all my answers and learned a lot about the programs.My mathematical knowledge is also rise.

The thanks also get on the BICS team.Without this project the student doesn't have the possibilities get more experience in other domains that they choose and like.We learn a lot in one semester with this project.The students learn to be organised,expand their language skills,scientific skills and also programming skills.

Finally I thanks a lot my family and friends for their support and help in this semester.

6. Conclusion

The goal of this project is to compare different gcd algorithms of large numbers.To program this algorithm was used

Java with help of NetBeans.The program get a virtual window where are showed the results.It is also programmed the time, to know which of the algorithms has a better power.That was goal of the scientific deliverable.The answer is that the Java gcd BigInteger algorithm was the fastest but this algorithm couldn't be explain because it is a method from Java.So it was decided to take the second fastest that was programmed by hand.It is the Binary gcd algorithm.It was explained why it is the fastest and the answer was because of the binary system and the shift operation.It was also a diagram with the results to see the difference of the duration.

So in technical part was the codes explain of the algorithms and the random code.The design of the NetBeans window was carefully showed and explained.All the requirement ere reached and successfully finished.

References

[1] BiCS Bachelor Semester Project Report Template: https://github.com/nicolasguelfi/lu.uni.course.bics.global University of Luxembourg, BiCS - Bachelor in Computer Science (2019).

[2] Extended Euclidean gcd algorithm explanation: https://brilliant.org/wiki/extended-euclidean-algorithm/

[3] Binary gcd algorithm explanation: http://www.cse.unt.edu/∼tarau/teaching/PP/NumberTheoretical/GCD/Binary

[4] Euclidean gcd algorithm explanation: https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm

[5] BigInteger function for the algorithms code: https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html

[6] Scientific vs. Technical: https://keydifferences.com/difference-between-science-and-technology.html

[7] Data save to Excel help: mauricemuteti.info/how-to-export-jtable-data-to-excel-in-java-netbeans-gui-tutorial/

[8] What is a algorithm and for what are they used: https://www.bbc.co.uk/bitesize/topics/z3tbwmn/articles/z3whpv4

[9] Excel: https://www.computerhope.com/jargon/e/excel.htm

[10] NetBeans: https://netbeans.org/

[11] Java: https://www.journaldev.com/32975/what-is-java-programming-language

7. Appendix

Source of the image used in NetBeans.

1) Icons for the NetBeans buttons: www.iconarchive.com/tag/java-netbeans