UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU

Graduação em Ciência da Computação

**Atividade Prática 02**

GBC065 – Modelagem e Simulação

Uberlândia

2018



**Atividade Prática 02**

Trabalho apresentado à disciplina de Modelagem e Simulação (GBC065), ministrada pelo professor Anderson Rodrigues dos Santos, para o curso de Bacharelado em Ciência da Computação, no período 2018-2, na Universidade Federal de Uberlândia.

**Grupo 02 – Integrantes:**

Antonio Carlos Neto

11611BCC054

Ronistone Gonçalves dos Reis Júnior

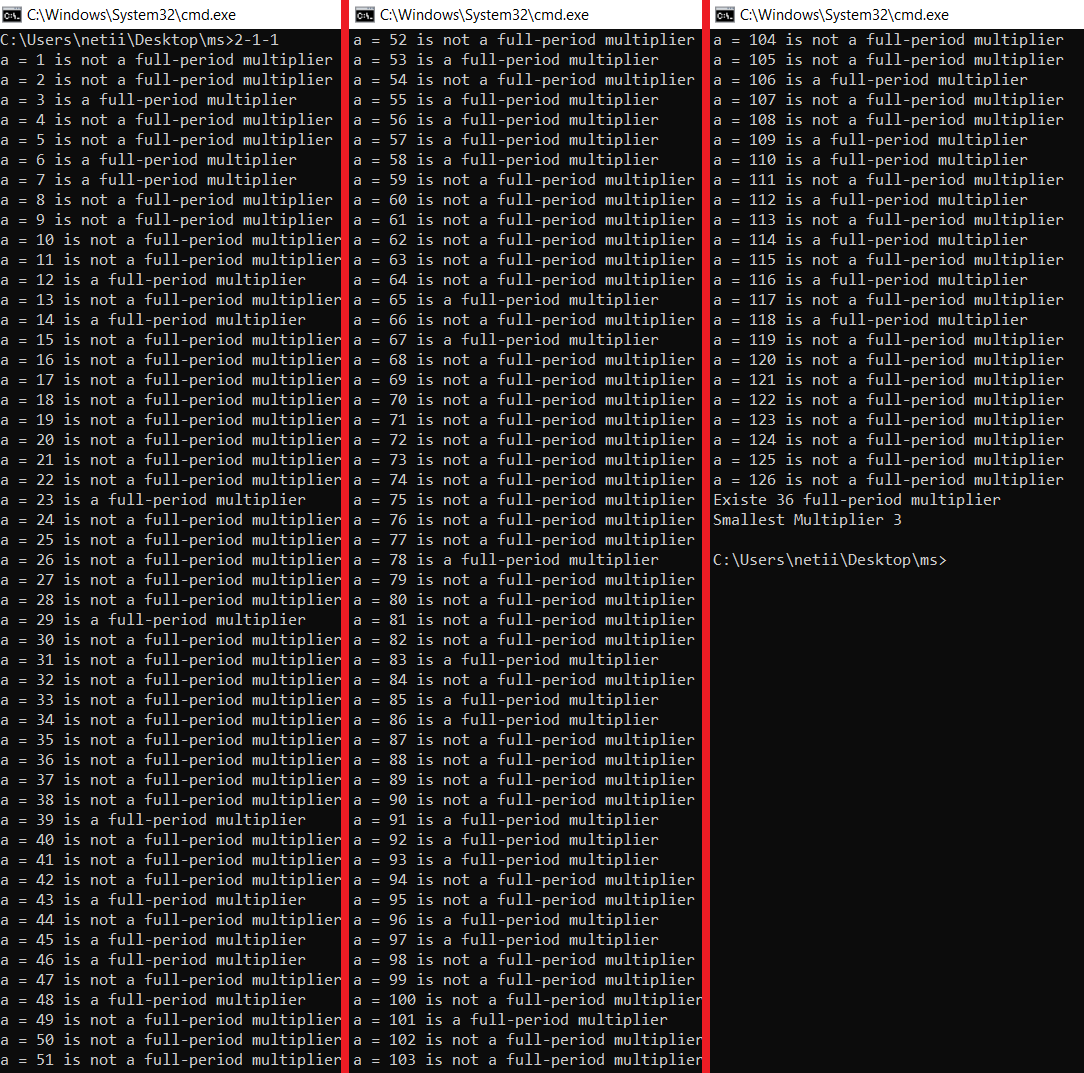
11521BCC018

Uberlândia

2018

**Exercise 2.1.1:**

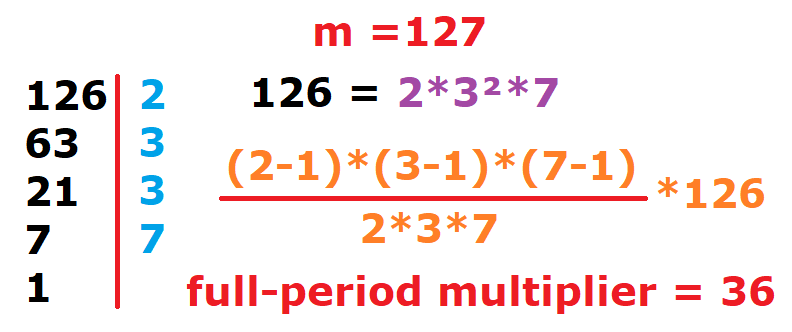
**For the tiny Lehmer generator defined by g(x) = ax mod 127, find all the full-period multipliers.**

****

**(a) How many are there?**

**R:** De acordo com o programa feito e o Teorema 2.1.3, temos 36 full-period multiplier.

* Teorema 2.1.3 aplicado com m = 127:



**(b) What is the smallest multiplier?**

**R:** De acordo com o programa feito(1-2-2.c), baseado no algoritmo 2.1.1, disponibilizado pelo livro texto, temos que 3 é o smallest multiplier.

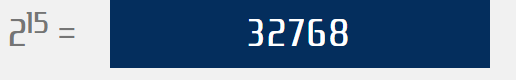
**Exercise 2.1.6:**

**In ANSI C an int is guaranteed to hold all integer values between −(2^15 −1) and 2^15 −1 inclusive.**

**(a) What is the largest prime modulus in this range?**

**R:** O largest prime modulus nesse intervalo é 32.749. Calculado pelo programa feito(1-2-6-a) e olhando na tabela disponível online.

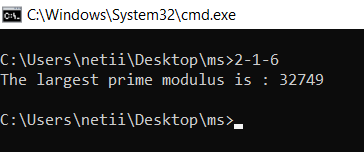
* Temos que 2^15 - 1 é 32.767.



* Temos pela tabela o largest prime modulus é 32.749.



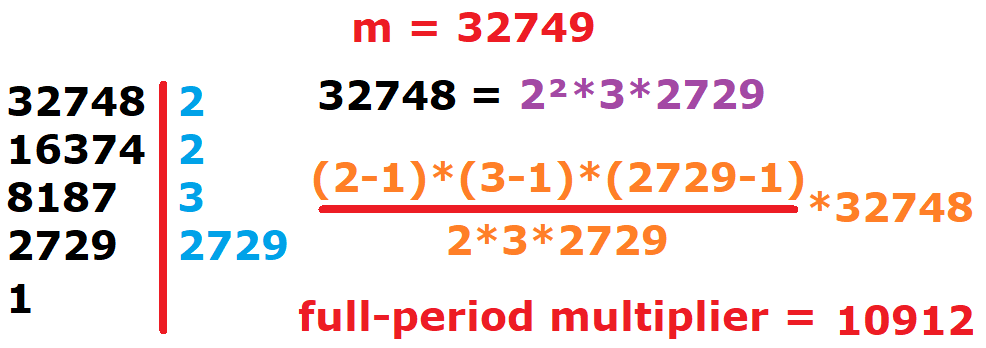
* Temos pelo programa o largest prime modulus é 32.749.



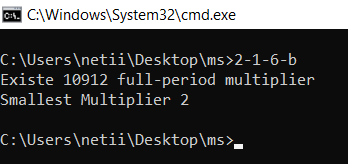
**(b) How many corresponding full-period multipliers are there and what is the smallest one?**

**R:** De acordo com o programa feito e o Teorema 2.1.3, temos 10912 full-period multiplier e temos que 2 é o smallest multiplier.

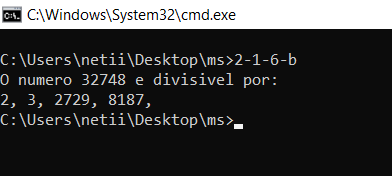
* Teorema 2.1.3 aplicado com m = 32.748:



* Pelo programa feito(2-1-6-b(2).c) com m = 32.748:



* Pelo programa feito(2-1-6-b(1).c), achamos os divisores do número 32.748, para auxiliar no teorema 2.1.3.



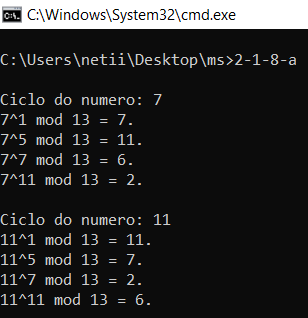
**Exercise 2.1.8:**

**(a) Evaluate 7^i mod 13 and 11^i mod 13 for i = 1, 5, 7, 11.**

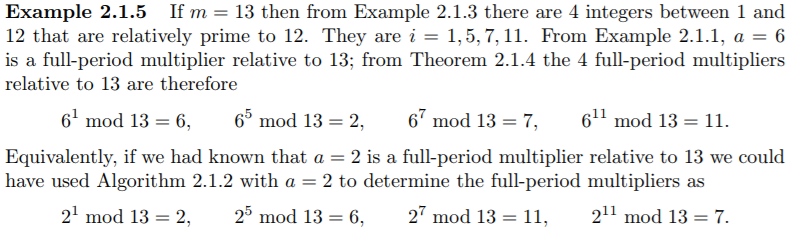
**R:** A tabela a seguir contém a resposta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | **i** | **a^i** | **m** | **a^i mod m** |
| **7** | **1** | **7** | **13** | **7** |
| **7** | **5** | **16.807** | **13** | **11** |
| **7** | **7** | **823.543** | **13** | **6** |
| **7** | **11** | **1.977.326** | **13** | **2** |
| **11** | **1** | **11** | **13** | **11** |
| **11** | **5** | **161.051** | **13** | **7** |
| **11** | **7** | **19.487.171** | **13** | **2** |
| **11** | **11** | **285.311.670.611** | **13** | **6** |

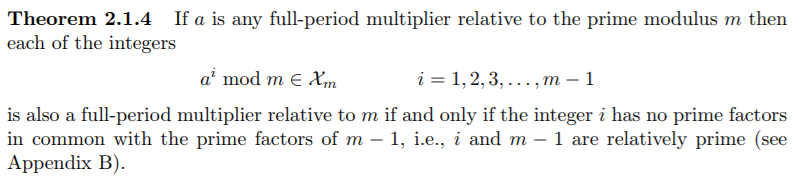
* As respostas da tabela e do programa feito(2-1-8-a.c) correspondem .



**(b) How does this relate to Example 2.1.5**

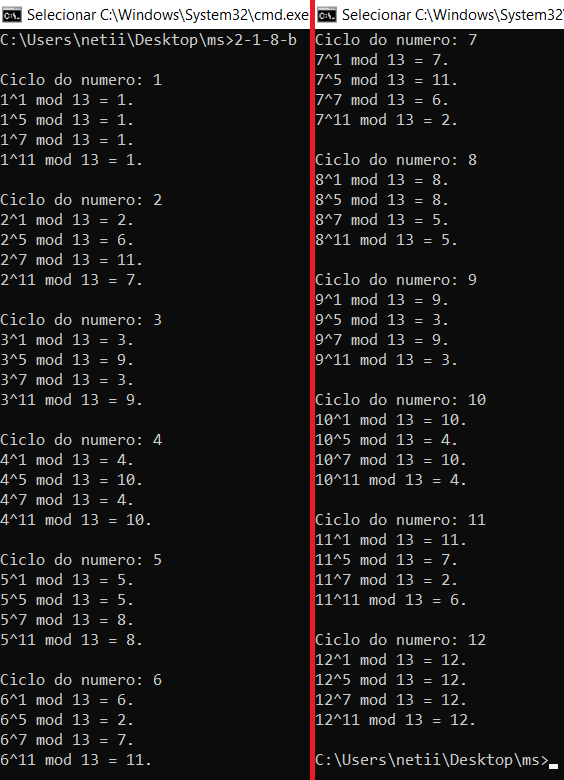
****

****

****

**R:** Estamos calculando as duas sequências de números restantes, utilizando os números 7, 11, que são full-period multiplier, da mesma forma que o Example 2.1.5. Utilizando i como 1, 5, 7, 11 que são primos entre si de 12, de forma que a é full-period multiplier do m = 13, gerando uma nova sequência de número aleatórios(a(s) que geram full-period), sendo que o tamanho é n(quantidades de número primos entre si de m-1).

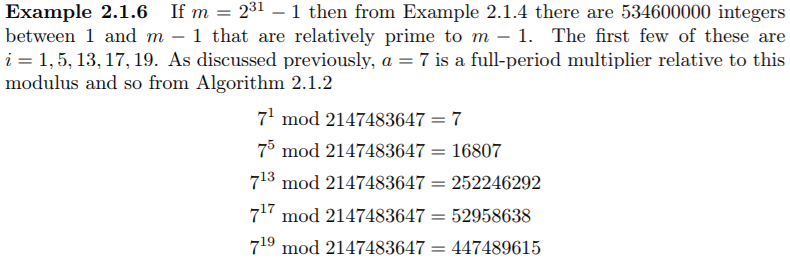
* Utilizando o programa(2-1-8-b.c) feito, provamos que os únicos números(valores de a) que possuem full-period utilizando a fórmula a^i mod m, são 2, 6, 7, 11, sendo que eles são full-period utilizando a fórmula (a\*xi) mod m.



**Exercise 2.1.9:**

**(a) Verify that the list of five full-period multipliers in Example 2.1.6 is correct.**

**R:** O Example 2.1.6 está correto, confira na tabela abaixo.



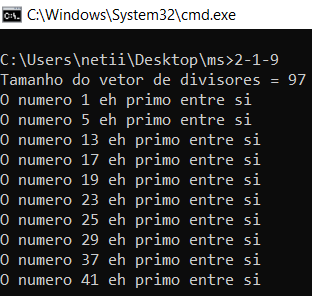
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | **i** | **a^i** | **m** | **a^i mod m** |
| **7** | **1** | **7** | **2.147.483.647** | **7** |
| **7** | **5** | **16.807** | **2.147.483.647** | **16.807** |
| **7** | **13** | **96.889.010.407** | **2.147.483.647** | **252.246.292** |
| **7** | **17** | **232.630.513.987.207** | **2.147.483.647** | **52.958.638** |
| **7** | **19** | **11.398.895.185.373.144** | **2.147.483.647** | **447.489.616** |

**(b) What are the next five elements in this list?**

**R:** Com o programa(2-1-9.c) feito, calculamos os 10 primeiros primos entre si com o número **2.147.483.646**, assim calculamos o restante da tabela.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | **i** | **a^i** | **m** | **a^i mod m** |
| **7** | **23** | **27.368.747.340.080.914.000** | **2.147.483.647** | **1.367.079.193** |
| **7** | **25** | **1.341.068.619.663.964.900.807** | **2.147.483.647** | **1144108930** |
| **7** | **29** | **3.219.905.755.813.179.726.837.607** | **2.147.483.647** | **373956417** |
| **7** | **37** | **18.562.115.921.017.574.302.453.163.671.207** | **2.147.483.647** | **655382362** |
| **7** | **41** | **44.567.640.326.363.195.900.190.045.974.568.007** | **2.147.483.647** | **1615021558** |

* Programa 2-1-9.c:

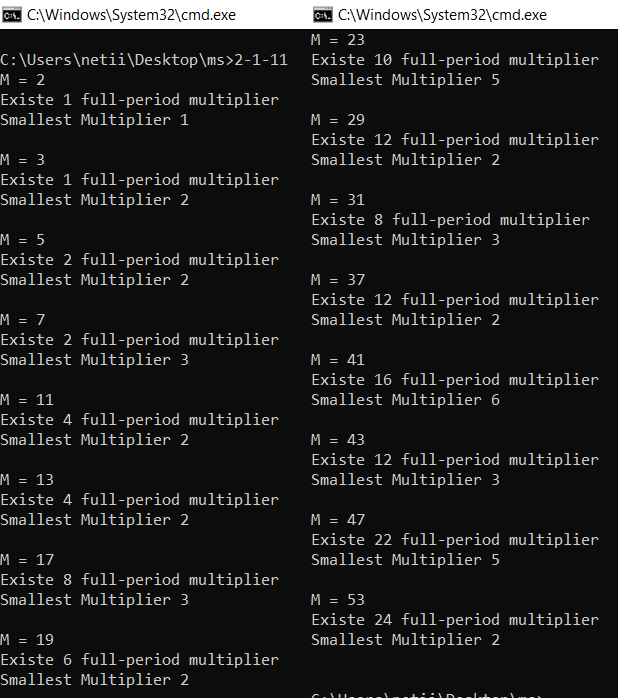


**Exercise 2.1.11:**

**For the first few prime moduli, this table lists the number of full-period multipliers and the smallest full-period multiplier. Add the next 10 rows to this table**

**R:** A tabela foi completada com o programa 2-1-11.c.

|  |  |  |
| --- | --- | --- |
| prime modulus m | number of full-period multipliers | smallest full-period multiplier a |
| 2 | 1 | 1 |
| 3 | 1 | 2 |
| 5 | 2 | 2 |
| 7 | 2 | 3 |
| 11 | 4 | 2 |
| 13 | 4 | 2 |
| 17 | 8 | 3 |
| 19 | 6 | 2 |
| 23 | 10 | 5 |
| 29 | 12 | 2 |
| 31 | 8 | 3 |
| 37 | 12 | 2 |
| 41 | 16 | 6 |
| 43 | 12 | 3 |
| 47 | 22 | 5 |
| 53 | 24 | 2 |

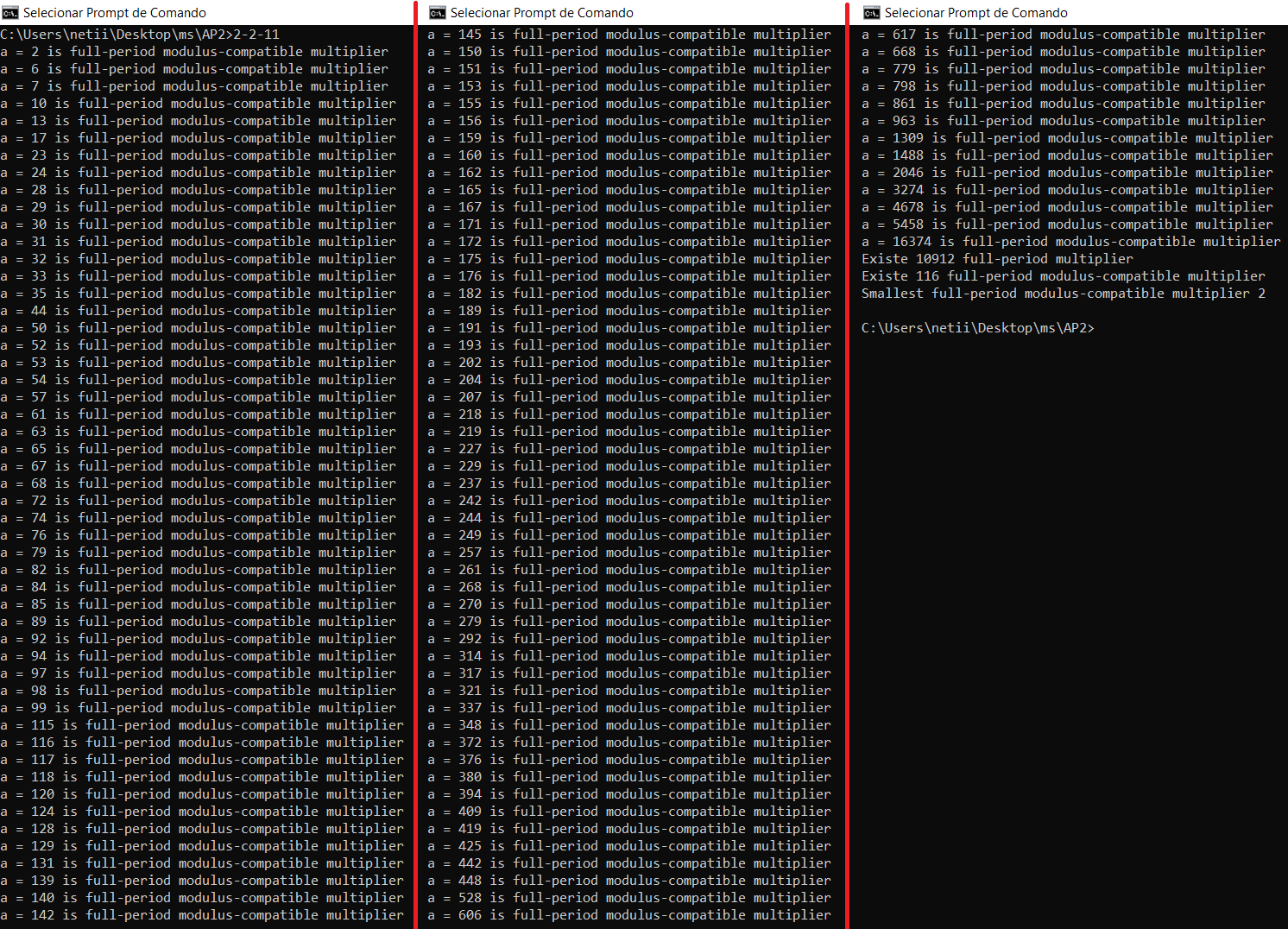


**Exercise 2.2.11:**

**Let m be the largest prime modulus less than or equal to 2^15 −1 (see Exercise 2.1.6).**

**(a) Compute all the corresponding modulus-compatible full-period multipliers.**

**R:** Considerando m = 32749, temos que 116 modulus-compatible full-period multipliers. Sendo que um modulus-compatible full-period multipliers é um full-period multipliers e o resto(m%a) é menor que o quociente(m/a);



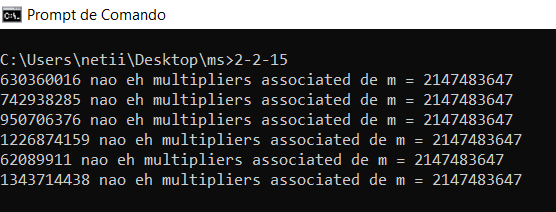
**(b) Comment on how this result relates to random number generation on systems that support 16-bit integer arithmetic only.**

**R:** Se escolhermos o valor de “a” como um módulo compatível, calculado acima, temos que um algoritmo, como exemplo 2.2.1 do livro texto, é suportado em uma sistem 16-bit, gerando assim número aleatórios.

**Exercise 2.2.15:**

**Determine whether the multipliers associated with m = 2^31 −1 given by Fishman (2001): a = 630 360 016, a = 742 938 285 a = 950 706 376, a = 1 226 874 159,a = 62 089 911, and a = 1 343 714 438 are modulus-compatible.**

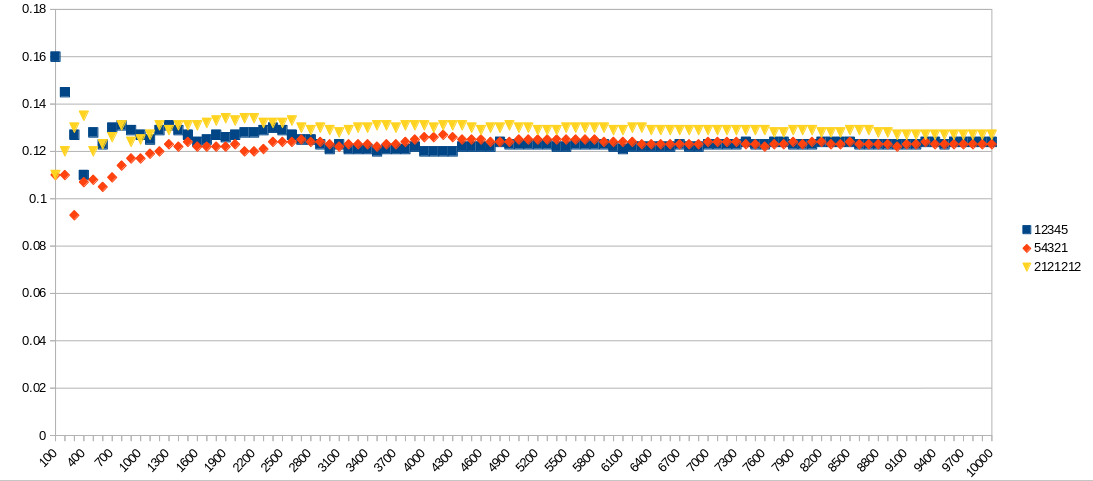
**R:** Temos que nenhum desses valores para a são módulo compatível.



**Exercise 2.3.6:**

**According to slides number seven and eight from section 2.3, example 2.3.6, construct a graph similar to slide eight but Pr(X=9).**

**R:** Como podemos ver no gráfico abaixo a medida que cresce o número de repetições temos a probabilidade variando entre 0.123 e 0.127 nos 3 seeds usados.



**Exercise 2.4.1:**

**Modify program det so that all 2^31 −1 possible matrices associated with the random number generator with (a, m) = (48271, 2^31 −1) are generated.**

**R:** Para gerar as matrizes temos apenas (2^31-1)/9 matrizes pois na próxima execução temos uma repetição devido a cada matriz utilizar 9 valores, uma solução para isto é realizar a permutação dos valores dentro da matriz, mas teríamos 10^9 operações o que ficou inviável para testes.