Prática 2 – MS

Grupo:

Andressa Alvilino Ferreira Silva

Matheus Cunha Reis

Weuler Borges

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? (b) What is the smallest multiplier?

$g(x) = a * x \; mod \; 127 \qquad m - 1 = 127 - 1 = 126 = 2 * 3^2 * 7$

Primos de m-1 : 2, 3, 7

Multiplicadores de período completo:

$$\frac{(p1-1)*(p2-1)*\ldots*(pr-1)}{(p1*p2*\ldots*pr)}*(m-1) =$$
$$= \frac{(2-1)*(3-1)*(7-1)}{2*3*7}*(2^2*3*7)$$
$$= 1*2*6*3$$
$$= 36$$

a) Tem-se 36 multiplicadores de período completo

b) O menor multiplicador de período completo é o 3 (solução achada pelo código 2_1_1.c)

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

a) O maior primo no intervalo é 32749 (solução achada pelo código 2_1_6.c)

b) $g(x) = a * x \; mod \; 32749$

$m - 1 = 32749 - 1 = 32746 = 2^2 * 3 * 2729$

Primos de m-1 : 2, 3, 2729

Multiplicadores de período completo:

$$\frac{(p1-1)*(p2-1)*\ldots*(pr-1)}{(p1*p2*\ldots*pr)} * (m-1) =$$

$$= \frac{(2-1)*(3-1)*(2729-1)}{2*3*2729} * (2^2 * 3 * 2729)$$

$$= 1 * 2 * 2728 * 2 \qquad 8$$

$$= 10912$$

Tem-se 10912 multiplicadores de período completo

c) O menor multiplicador de período completo é o 2 (solução achada pelo código 2_1_1.c)

2.1.8) (a) Evaluate 7^i mod 13 and 11^i mod 13 for i = 1, 5, 7, 11. (b) How does this relate to Example 2.1.5?

a) $7^1 \; mod \; 13 = 7$ $\qquad$ $11^1 \; mod \; 13 = 11$

$7^5 \; mod \; 13 = 11$ $\qquad$ $11^5 \; mod \; 13 = 7$

$7^7 \; mod \; 13 = 6$ $\qquad$ $11^7 \; mod \; 13 = 2$

$7^{11} \; mod \; 13 = 2$ $\qquad$ $11^{11} mod \; 13 = 6$

b) No exemplo 2.1.5 é usado o mesmo valor de m, ou seja, temos os mesmos primos relativos e também temos os mesmos 4 multiplicadores de período completo. A diferença é que no exemplo tem a informação de somente um multiplicador e com o teorema 2.1.4 os outros são descobertos. Já no exercício, temos como informações outros 2 multiplicadores diferentes e com eles também conseguimos descobrir os outros, o que prova o teorema 2.1.4 que diz que se elevarmos um multiplicador de período completo aos primos relativos de m-1 conseguimos achar os outros.

2.1.9) (a) Verify that the list of five full-period multipliers in Example 2.1.6 is correct. (b) What are the next five elements in this list?

a) Sim, está correto

b) $7^{23}\ mod\ 2147483647 = 680742115$

$7^{25}\ mod\ 2147483647 = 1144108930$

$7^{29}\ mod\ 2147483647 = 373956417$

$7^{37}\ mod\ 2147483647 = 655382362$

$7^{41}\ mod\ 2147483647 = 1615021558$

Os próximos 5 multiplicadores de período completo são 680742115, 1144108930, 373956417, 655382362, 1615021558

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.

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

(solução achada pelos códigos 2_1_11.cpp e 2_1_1.c)

2.2.9) You have been hired as a consultant by XYZ Inc to assess the market potential of a relatively inexpensive hardware random number generator they may develop for high-speed scientific computing applications. List all the technical reasons you can think of to convince them this is a bad idea.

O autor do livro cita 5 aspectos fundamentais para um gerador de números aleatórios: aleatoriedade, controlabilidade, portabilidade, eficiência e documentação.

Nesse contexto ele argumenta que apenas o gerador de software tem potencial de satisfazer todos esses critérios simultaneamente.

A geração de números aleatórios através de um hardware pode, dependendo da implementação, gerar números satisfatoriamente aleatórios. A eficiência e a documentação de um RNG de harware também podem atingir níveis aceitáveis.

Apesar disso, a controlabilidade, que é um fator essencial para experimentos cientificios, é questionável. Por exemplo um gerador que utiliza “White noise” como input pode gerar uma sequência de números diferentes em um contexto diferente. Isso tornaria os resultados não reprodutíveis, pois seria impossível rodar o experimento com uma mesma sequência de números 10 vezes.

A portabilidade também é pior do que a do RNG de software já que como o gerador depende de um contexto físico, as sequências de números podem variar caso sejam o RNG físico seja utilizado em diferentes localidades ou sistemas.

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

a) Com m = 32749, temos 116 multiplicadores de período completo modulo compatível. São eles:

2 6 7 10 13 17 23 24 28 29 30 31 32 33 35 44 50 52 53 54 57 61 63 65 67 68 72 74 76 79 82 84 85 89 92 94 97 98 99 115 116 117 118 120 124 128 129 131 139 140 142 145 150 151 153 155 156 159 160 162 165 167 171

172 175 176 182 189 191 193 202 204 207 218 219 227 229 237 242 244 249 257 261 268 270 279 292 314 317 321 337 348 372 376 380 394 409 419 425 442 448 528 606 617 668 779 798 861 963 1309 1488 2046 3274 4678 5458 16374

(solução achada pelo código 2_2_11.cpp)

b) O resultado se relaciona com a geração de números aleatórios porque como o sistema só aguenta até 2^15 – 1, quando o calculo de a*x é feito, o resultado não pode ultrapassar esse valor. Ou seja, não é possível usar qualquer número de multiplicador até 2^15 – 1 e por isso é necessário achar os multiplicadores de período completo que são módulos compatíveis, pois eles nos garantem que quando calculado a*x não ocorrerá overflow.