Simulación de Sistemas

grado en ingeniería informática

PRÁCTICA 2

Modelos de Monte Carlo. Generadores de datos

Autor
Adrián Acosa Sánchez

Rama
Computación y Sistemas Inteligentes

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2022-2023

Índice general

Capítulo 1

Mi Segundo Modelo de Simulación de Monte Carlo

1.1. Modelización del modelo por Monte Carlo

Una vez que tenemos el problema modelado por Monte Carlo es momento de ver qué tan bueno es éste haciendole pruebas sobre cómo se comporta para distintas combinaciones de x e y y comparándolas con la expresión analítica del problema que viene en el guión.

Las combinaciones de x e y que usaré para ello serán:

- $x = 10$ / $y = 1$
- $x = 10$ / $y = 5$
- $x = 10$ / $y = 8$
- $x = 15$ / $y = 10$

Cada una de las combinaciones serán simuladas con números de veces 100, 1000, 5000, 10000 y 100000 veces para ver si cambian los resultados en función de las veces simuladas. También hay que probar cada una de las distribuciones que se menciona en el guión.

Para el primer tipo de distribución obtenemos los siguientes resultados:

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	1	100	87	452.600000	0.001296
10	1	1000	96	416.030000	0.013941
10	1	5000	89	404.830000	0.062063
10	1	10000	93	405.324000	0.125970
10	1	100000	90	401.903400	1.241647
10	5	100	60	152.100000	0.001286
10	5	1000	54	132.020000	0.012689
10	5	5000	50	124.414000	0.063354
10	5	10000	54	123.520000	0.127698
10	5	100000	48	123.101100	1.254433
10	8	100	30	29.500000	0.001276
10	8	1000	20	21.380000	0.012423
10	8	5000	16	19.664000	0.064112
10	8	10000	22	20.024000	0.127483
10	8	100000	20	19.091300	1.260559
15	10	100	34	119.300000	0.001311
15	10	1000	35	88.825000	0.012580
15	10	5000	35	81.895000	0.063079
15	10	10000	34	82.023500	0.135397
15	10	100000	34	81.319700	1.256663

Cuadro 1.1: Resultados de ejecutar el modelo de Monte Carlo con distribución uniforme.

Para poder realizar correctamente el estudio de si ha sido buena modelización o no, primero debemos conocer el valor óptimo de s para los diferentes casos. Para ello usaré la expresión que aparece en el guión de prácticas:

- Para $x = 10$ e $y = 1$, $s^* = 89.5$
- Para $x = 10$ e $y = 5$, $s^* = 49.5$
- Para $x = 10$ e $y = 8$, $s^* = 19.5$
- Para $x = 15$ e $y = 10$, $s^* = 32.83$

Se puede observar cómo los valores obtenidos por la simulación de Monte Carlo son bastante cercanos al valor óptimo del modelo. Los valores obtenidos cuando simulamos con bajas repeticiones varían mucho el resultado con respecto al valor óptimo. Sin embargo, a medida que repetimos el proceso con un número de veces mucho mayor, nos acercamos más a este valor óptimo debido a que por mucho que estemos usando valores aleatorios, de media siempre obtendremos mejores resultados cuanto mayor sea el número de la muestra.

Podemos concluir que el modelo es bastante preciso para representar el modelo siempre y cuando usemos un número de ejecuciones altas.

Ahora vamos a ver cómo se comporta el modelo cuando usamos una distribución proporcional:

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	1	100	85	590.700000	0.001600
10	1	1000	96	573.800000	0.015891
10	1	5000	97	567.134000	0.079397
10	1	10000	94	566.568000	0.157738
10	1	100000	93	564.064400	1.566322
10	5	100	74	275.800000	0.001523
10	5	1000	71	248.740000	0.015133
10	5	5000	69	236.150000	0.078578
10	5	10000	70	233.457000	0.154835
10	5	100000	70	232.022300	1.566630
10	8	100	59	79.400000	0.001511
10	8	1000	44	61.970000	0.015133
10	8	5000	46	59.062000	0.076884
10	8	10000	43	58.036000	0.154246
10	8	100000	46	58.102200	1.541667
15	10	100	54	213.000000	0.001563
15	10	1000	55	195.515000	0.015857
15	10	5000	60	191.262000	0.077931
15	10	10000	58	189.306500	0.154304
15	10	100000	58	189.472250	1.533207

Cuadro 1.2: Resultados de ejecutar el modelo de Monte Carlo con distribución proporcional creciente.

En éste caso nos pasa igual que en el primero. Si nos fijamos podemos observar cómo cuando usamos un número de repeticiones muy bajos, los valores obtenidos varían mucho y a medida que nos vamos acercando a números de repeticiones mucho más altos obtenemos menos variación en los resultados. Como no tenemos una expresión analítica que nos calcule el valor óptimo de s, concluiremos que el modelo obtiene unos valores muy cercanos al óptimo cuando usamos el número de repeticiones más alta para cada caso.

Una diferencia observable a simple vista entre ésta distribución y la de la tabla 1.1 es que las entradas de la columna "Mejor s"son mucho más altas aquí. Ésto se debe a que la distribución proporcional es creciente y ésto provoca que se den como mejores resultados aquellas demandas que sean más altas ya que tienen más probabilidad de salir. En cuanto al tiempo de ejecución la diferencia es mínima y no es muy relevante.

Y por último vamos a ver la simulación mediante una distribución "triangular":

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	1	100	69	441.200000	0.001238
10	1	1000	78	421.850000	0.012166
10	1	5000	78	416.576000	0.061793
10	1	10000	74	417.297000	0.123098
10	1	100000	78	415.443100	1.222110
10	5	100	52	192.500000	0.001265
10	5	1000	53	169.810000	0.012635
10	5	5000	50	169.358000	0.066643
10	5	10000	52	168.650000	0.132698
10	5	100000	48	166.742900	1.261132
10	8	100	41	47.900000	0.001276
10	8	1000	37	43.730000	0.012694
10	8	5000	30	42.630000	0.062987
10	8	10000	33	42.908000	0.125408
10	8	100000	33	42.368300	1.226529
15	10	100	44	155.200000	0.001270
15	10	1000	38	137.320000	0.012593
15	10	5000	38	137.305000	0.063753
15	10	10000	41	137.159500	0.125660
15	10	100000	41	136.963300	1.233677

Cuadro 1.3: Resultados de ejecutar el modelo de Monte Carlo con distribución "triangular".

En éste último caso podemos ver que salen muchos valores rondando 50. Ésta distribución "triangular"lo que provoca es que sean más probables los valores medios que los valores altos o bajos. Aquí se ve reflejado en que en la primera simulación vemos como en las tablas 1.1 y 1.2 salían valores mucho más altos y ahora han salido unos valores más bajo. Ésto es porque, como acabo de mencionar, los valores altos de demanda se ven penalizados por ésta distribución ya que tienen menor probabilidad que los valores que se encuentran en mitad de la tabla.

Como conclusión podemos decir que éstos modelos se acercan muy bien a la realidad en el caso de que los ejecutemos un número de veces relativamente alto, para así ser más precisos con la estimación (como hemos podido comprobar empíricamente en el caso de la tabla 1.1).

1.2. Modificaciones del modelo

1.2.1. Primera modificación

Ahora se da la opción al establecimiento de poder devolver las unidades no vendidas pero que haya que pagar una cantidad fija (z) como gastos de devolución, siendo esta cantidad independiente del número de unidades que se devuelvan.

Para ésta primera modificación he hecho pruebas con los siguientes datos ejecutándolos 10000 y 100000 veces cada una:

- $x = 10$ / $z = 1$
- $x = 10$ / $z = 5$
- $x = 10$ / $z = 10$
- $x = 10$ / $z = 100$

Y con estos datos he obtenido los siguientes resultados para cada una de las distribuciones:

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Coste por devolución (z)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	0	1	10000	97	499.369900	0.123301
10	0	1	100000	97	495.345140	1.280890
10	0	5	10000	99	493.390000	0.128021
10	0	5	100000	99	490.786500	1.254347
10	0	10	10000	98	488.275000	0.125137
10	0	10	100000	96	485.424300	1.253742
10	0	100	10000	91	412.304000	0.123278
10	0	100	100000	87	400.858700	1.243047

Cuadro 1.4: Resultados de ejecutar la primera modificación del modelo con distribución uniforme.

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Coste por devolución (z)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	0	1	10000	99	661.257400	0.151678
10	0	1	100000	99	659.731950	1.593401
10	0	5	10000	95	656.408500	0.162619
10	0	5	100000	98	655.351650	1.638641
10	0	10	10000	97	651.690000	0.152098
10	0	10	100000	95	650.302700	1.586894
10	0	100	10000	88	572.008000	0.154631
10	0	100	100000	89	570.320000	1.574889

Cuadro 1.5: Resultados de ejecutar la primera modificación del modelo con distribución proporcional creciente.

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Coste por devolución (z)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	0	1	10000	98	502.639000	0.121352
10	0	1	100000	96	499.394580	1.228234
10	0	5	10000	84	498.231000	0.122613
10	0	5	100000	93	496.194550	1.275088
10	0	10	10000	97	492.063000	0.127478
10	0	10	100000	96	491.285000	1.291043
10	0	100	10000	82	406.606000	0.125063
10	0	100	100000	80	404.619600	1.307969

Cuadro 1.6: Resultados de ejecutar la primera modificación del modelo con distribución "triangular".

En el caso de esta modificación, no tenemos que tener en cuenta la pérdida por unidad no vendida ya que siempre será peor pagar por las que no se vendan en el caso de que podamos pagar una cantidad fija por devolverlas. Independientemente de la distribución utilizada, vemos como los resultados son muy similares para las tres.

Se puede apreciar que realizando esta modificación la demanda crece muchísimo, ya que pedir más unidades de las que vendemos incluso en el caso de que el coste por devolución sea de 100, merece la pena pagar esa cantidad que pagar por cada unidad no vendida en caso de que se queden bastantes sin vender.

1.2.2. Segunda modificación

En este caso la modificación consiste en que la devolución sigue siendo a costo pero dando a elegir entre la opción de asumir los costes de las unidades no vendidas si es un número pequeño de unidades y la opción de pagar los gastos de devolución si son grandes y le merece más la pena que pagar por cada unidad no vendida.

Aplicando el mismo procedimiento que antes, he simulado 10000 y 100000 veces las tres distribuciones con $z = 200$ y con los siguientes datos:

- $x = 10$ / $y = 5$
- $x = 10$ / $y = 10$
- $x = 20$ / $y = 5$
- $x = 20$ / $y = 10$

Los resultados de ésta simulación son los siguientes para las tres distribuciones:

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Coste por devolución (z)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	5	200	10000	82	359.795000	0.122845
10	5	200	100000	79	356.288550	1.247951
10	10	200	10000	79	342.371000	0.127144
10	10	200	100000	81	336.210000	1.265307
25	5	200	10000	91	1094.779000	0.126661
25	5	200	100000	87	1086.849500	1.275876
25	10	200	10000	87	1075.621000	0.126274
25	10	200	100000	91	1070.513500	1.324787

Cuadro 1.7: Resultados de ejecutar la segunda modificación del modelo con distribución uniforme.

Ganancia por venta (x)	**Pérdida por unidad no vendida (y)**	**Coste por devolución (z)**	**Veces ejecutadas**	**Mejor s**	**Mejor ganancia media**	**Tiempo (en seg.)**
10	5	200	10000	85	558.283000	0.156210
10	5	200	100000	87	552.674400	1.604674
10	10	200	10000	77	526.622000	0.154310
10	10	200	100000	81	525.970800	1.518530
25	5	200	10000	96	1531.620500	0.151326
25	5	200	100000	94	1531.736700	1.527295
25	10	200	10000	93	1515.604000	0.152044
25	10	200	100000	93	1503.659450	1.526752

Cuadro 1.8: Resultados de ejecutar la segunda modificación del modelo con distribución proporcional creciente.

Ganancia por venta (x)	Pérdida por unidad no vendida (y)	Coste por devolución (z)	Veces ejecutadas	Mejor s	Mejor ganancia media	Tiempo (en seg.)
10	5	200	10000	60	391.133000	0.122836
10	5	200	100000	61	389.268250	1.259295
10	10	200	10000	61	363.009000	0.128892
10	10	200	100000	59	358.476200	1.277941
25	5	200	10000	75	1117.495000	0.139728
25	5	200	100000	73	1111.467550	1.286411
25	10	200	10000	80	1086.311000	0.127489
25	10	200	100000	77	1076.797550	1.290287

Cuadro 1.9: Resultados de ejecutar la segunda modificación del modelo con distribución "triangular".

En este caso he aumentado el coste por devolución para que sea un caso más exigente a la hora de elegir qué hacer con las unidades no vendidas.

Para el caso de la distibución uniforme se piden muchas más unidades en comparación con las otras dos pero la ganancia media obtenida es menor. En la distribución triangular se piden bastantes menos unidades que en las otras dos (debido a que tienen más probabilidad los valores intermedios) y sin embargo se le saca bastante ganancia teniendo en cuenta las unidades solicitadas. Pero la distribución que más beneficio medio tiene es la distribución proporcional creciente debido a que hay más probabilidad para las demandas más altas, por lo que como podemos observar, obtenemos una mayor demanda que en el resto de las tablas y aún así se consigue un mayor beneficio.

La conclusión final de ésta modificación es que es mucho más rentable en esta situación que en el resto el arriesgarse a pedir muchas más unidades (como podemos ver con la distribución proporcional creciente) ya que la pérdida que vayamos a obtener puede minimizarse en función del menor valor entre z y pagar por cada una de las unidades no vendidas una cierta cantidad.

Capítulo 2

Generadores de datos

2.1. Mejorando los generadores

En este apartado se pide que se realicen tres mejoras sobre los generadores de datos dados para la práctica. Las mejoras propuestas son las siguientes:

1. Ordenarlos valores de la variable en orden decreciente de probabilidad antes de construir la tabla de búsqueda.
2. Implementar una busqueda binaria para buscar en la tabla.
3. Para el generador (a), deducir un método para el que su tiempo de ejecución sea constante.

Para las comparaciones usaré la distribución "triangular"que es la que da tiempos de ejecución más altos. Sin más preámbulos, vamos con la primera mejora.

2.1.1. Reordenación de valores

Vamos a comparar con el generador base cuánto tiempo se mejora.

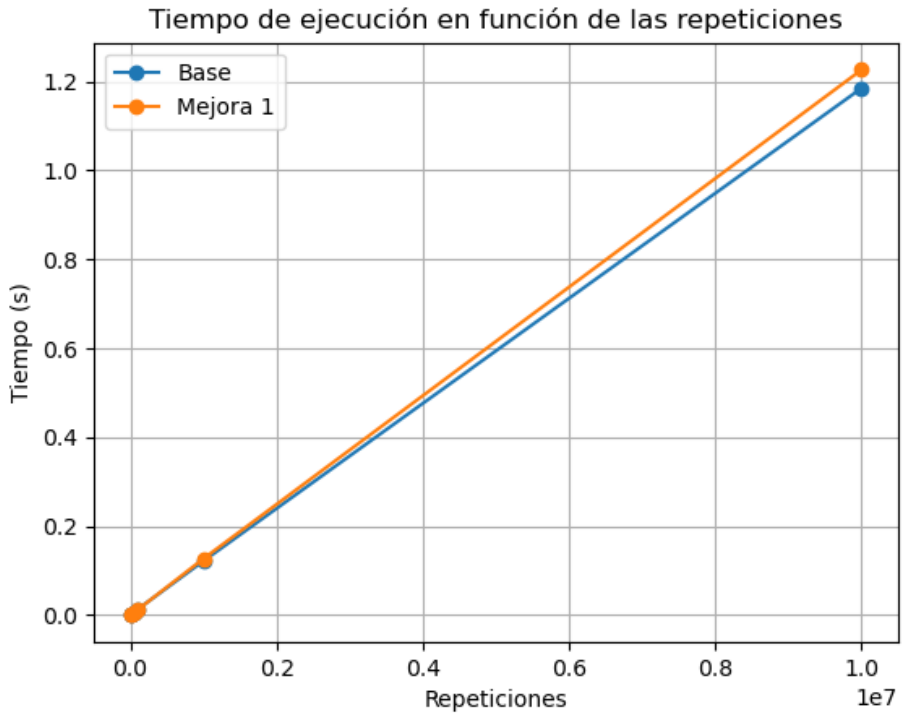

Figura 2.1: Generador base vs. la primera mejora

Podemos apreciar en la imagen como se consigue una pequeña mejora únicamente reordenando el vector para que las probabilidades más altas esten al principio. Aún así, se sigue teniendo el problema de que la búsqueda en el vector es lineal como se puede apreciar, y por lo tanto, en el peor caso posible, tendríamos que buscar en las últimas casillas del vector hasta encontrar el valor con la probabilidad baja.

2.1.2. Búsqueda binaria

En esta mejora se pide que cuando tengamos que generar la demanda, los valores en vez de ser encontrados mediante una búsqueda lineal, los buscaremos mediante una búsqueda binaria que es mucho mejor. Por lo tanto vamos a compararlo con el generador base también y con el generador de la mejora 1 para comprobar si mejoramos o empeoramos con respecto a ésta última:

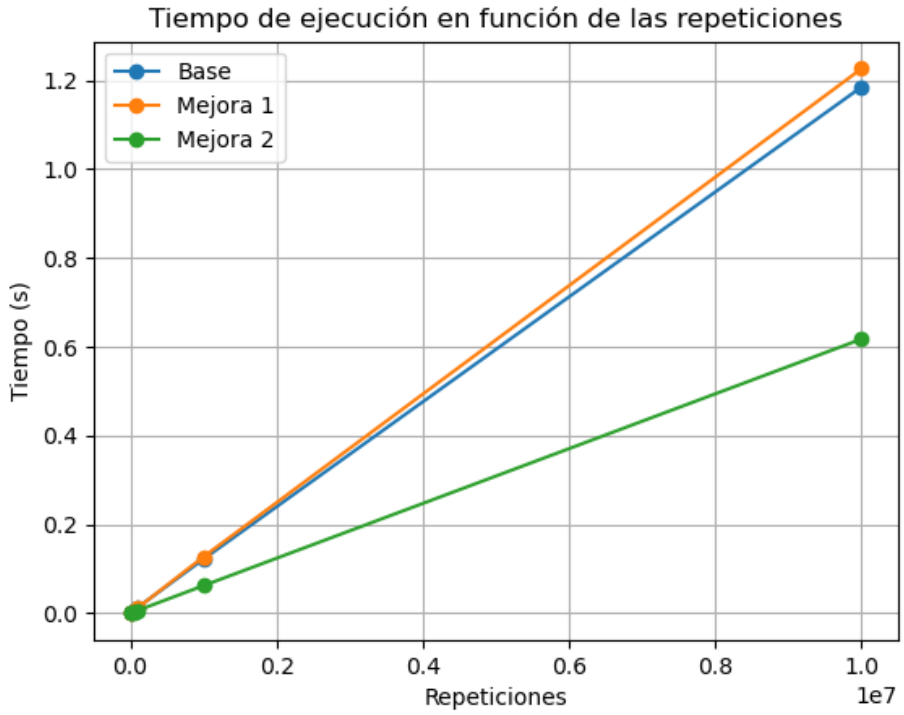

Figura 2.2: Generador base vs. mejora 1 vs. mejora 2

En ésta mejora, como era de esperar, obtenemos una mejora muy buena con respecto al generador base y a la primera mejora. Esto es debido a que la búsqueda de los valores en éste caso es mucho más eficiente, y por tanto tarda muchísimo menos tiempo (Estamos hablando que hemos mejorado la eficiencia de $\mathcal{O}(n)$ a $\mathcal{O}(\log n)$).

2.1.3. Tiempo de ejecución constante

Para poder deducir el método de que la tabla (a) nos de un tiempo de ejecución constante a la hora de generar números hay que pensar en que el valor que devuelve el generador es siempre un valor decimal que luego buscamos en nuestra tabla de probabilidades para que se devuelva el índice. Si nos fijamos en los índices que nos devuelven, tienen mucho parecido al número que estamos buscando dentro del array, y es que podemos obtener el índice directamente si multiplicamos la probabilidad por 100 y nos quedamos con la parte entera. Así podríamos mejorar la eficiencia a un orden de $\mathcal{O}(1)$, ya que simplemente tendríamos que hacer una multiplicación al número generado para poder obtener el índice que ocupa en el array.

Implementando esta mejora en nuestro modelo y comparado con todas las mejoras y el generador base obtenemos los siguientes resultados (recordando que todos se ejecuten con la tabla (a)):

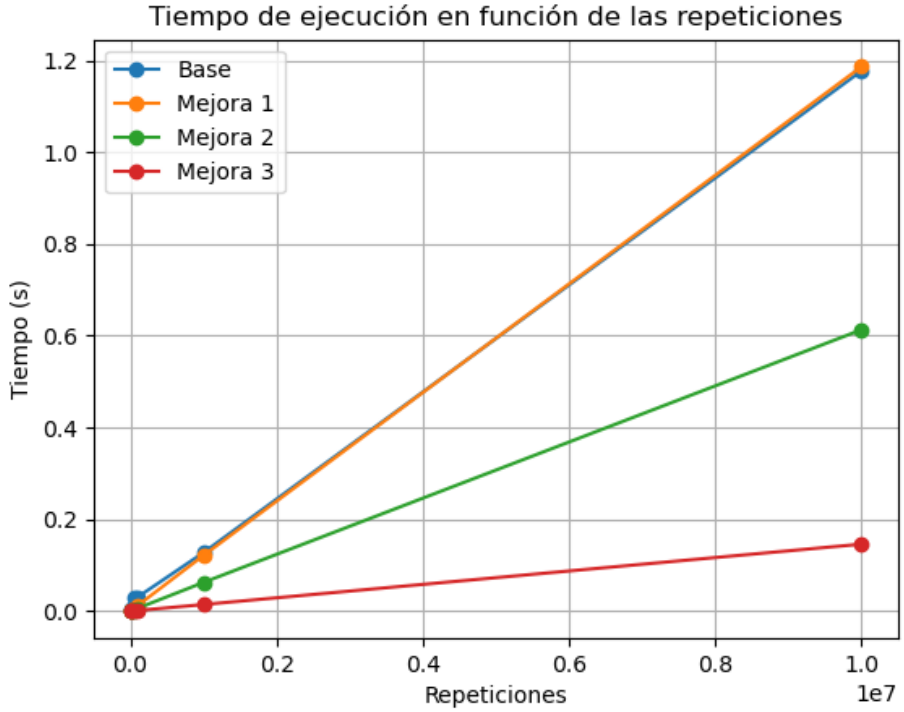

Figura 2.3: Comparación de todas las mejoras y el generador base

Y como era de esperar, el generador con la mejora 3 es muchísimo mejor que el generador con la búsqueda lineal de los valores.

2.2. Generador congruencial lineal multiplicativo

En este apartado se pide que se implemente el generador congruencial lineal multiplicativo que se indica a continuación

$$x_{n+1} = (1013x_n) \quad \text{mód } m \tag{2.1}$$

siendo $m = 2^{12}$ con diferentes implementaciones:

- Aritmética entera
- Aritmética real "artesanal"
- Aritmética real "artesanal" corregida
- Aritmética real con *fmod*

Vamos a ver qué periodos nos da cada una de las implementaciones:

Semilla	**Aritmética entera**	**Aritmética real "artesanal"**	**Aritmética real "artesanal" corregida**	**Aritmética real con *fmod***
1	1024	1024	1024	1024
24	128	128	128	128
200	128	128	128	128
321	1024	1024	1024	1024

Todas son implementaciones son correctas, debido a que un generador congruencial multiplicativo lineal por definición tiene como periodo máximo $\frac{m}{4}$ y lo alcanza cuando x_0 es impar (como es nuestro caso). En caso contrario, no alcanza su periodo máximo como se puede observar en la tabla para los valores pares escogidos y para cualquiera que se escoja.