情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2011-HPC-131 No.3
2011/10/6

高速並列計算用の 状態空間の小さな高品質疑似乱数生成器

斎 藤 睦 夫[†1] 松 本 眞[†2]

著者らは，状態空間 127 ビット，周期 $2^{127}-1$ の疑似乱数生成器 TinyMT を開発した．TinyMT はパラメータ化された疑似乱数生成器であり，パラメータを変えることによって異なる疑似乱数系列を生成することが出来る．パラメータを含めた使用メモリは，28 バイトであり，レジスタや一次キャッシュなどの高速メモリへの格納に適している．出力の品質については，TestU01[4] の BigCrush で検定し，これをパスした．

A high quality pseudo random number generator with small internal state

Mutsuo Saito[†1] and Makoto Matsumoto[†2]

Authors developed a pseudorandom number generator, called TinyMT, which has a period of $2^{127}-1$, with 127-bit internal state. TinyMT is a parameterized pseudorandom number generator, i.e. it can generate distinct pseudorandom number sequences by changing its parameters. Memory used by TinyMT including parameters are 28 bytes, therefore, the memory used by TinyMT can be stored in fast memory like registers or primary cache. The output of TinyMT is tested using BigCrush of TestU01[4] and passed the test.

†1 広島大学
Hiroshima University
†2 東京大学
The University of Tokyo

1. はじめに

高速計算機を利用した科学技術シミュレーションにおいては，しばしば疑似乱数が使用され，その生成速度はシミュレーションの速度に影響を及ぼす．

Mersenne Twister(MT)[6] は高速高品質な疑似乱数生成器であり，科学技術シミュレーションにおいて広く使用されている．また，著者らによる Mersenne Twister for Graphic Processors(MTGP)[11] は GPU に特化した疑似乱数生成器であり，NVIDIA の GPU において MT を上回る性能を示した．

今回報告する Tiny Mersenne Twister(TinyMT) は，プロセッサではなく，メモリアーキテクチャを仮定している．すなわち，小容量で高速なメモリである．そのために，状態空間の小さな疑似乱数生成器を開発した．状態空間 127 ビット，周期 $2^{127}-1$ の疑似乱数生成器である．また，並列計算においてメモリ内容の同期を取ることは速度低下に繋がるため，同期を取らなくて済むように設計した．すなわち，パラメータを変えることによって多数の疑似乱数系列を生成できるようにした．パラメータと状態空間を合わせた使用メモリは 28 バイトである.（計算過程の一時変数を除く）

2. TinyMT

TinyMT は，線形状態遷移関数と出力関数から構成される．MT と同様に欠けた配列を使用するが，違いも多い．

2.1 線形状態遷移関数

$\mathbb{F}_2$ で 0,1 からなる二元体を表すことにする．A, B を $\mathbb{F}_2$ 上の 127×127 行列，s_n, s_{n-1} を $\mathbb{F}_2$ 上の 127 次元横ベクトルとする．

$$s_n = s_{n-1}AB$$

ここで A は排他的論理和とシフト演算による写像，B は MT で使用されている 1 ビットにパラメータを掛けたビット撹乱行列である．後述の出力関数と併せて回路図風に記述したものが**図 1** である．

図の中の $\bigoplus$ はビットごとの排他的論理和であり，$+$ は 2^{32} を法とする算術加算である．

C99 によるプログラムを**図 2** に挙げた．なお，プログラム中の外部変数 mat1, mat2 は合計 64 ビットの状態遷移パラメータであり，外部変数 status は状態空間を保持する配列である．行番号 14 からの if 文が行列 B に該当する処理である．行番号 5 で 0x7fffffff との

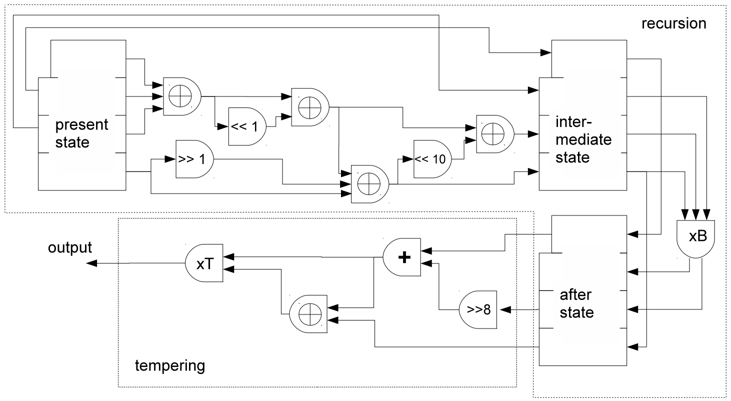

図 1 TinyMT 回路図
Fig. 1 TinyMT Circuit Diagram

論理積を取っているのは，状態空間の大きさを 127 ビットに制限するためである．

2.1.1 状態遷移の周期

線形空間 S の状態遷移関数 $f : S \mapsto S$ の特性多項式 $\varphi(t)$ が原始多項式なら，状態遷移は最大周期 $2^{\dim(S)} - 1$ となる[3)] [§3.2.2]．TinyMT では $\dim(S) = 127$ であり，$2^{127} - 1$ は素数なので，$\varphi(t)$ は既約なら原始である．TinyMTDC (後述 § 3) によって $\varphi(t)$ が既約となるパラメータを求めた．

2.1.2 パラメータと特性多項式

64 ビットパラメータ mat1，mat2 が定まれば，AB の特性多項式 $\in \mathbb{F}_2[t]$ が定まる．B が，状態空間の中の 1 ビットを参照して 1 の場合に定数パラメータとの排他的論理和を取るという写像の場合，パラメータから特性多項式（の係数ベクトル）への写像は Affine 写像となる．定数を引いた線形写像部分のランクを Mathematica で求めると 64 となった．よって，単射であることが確認された．従ってパラメータが異なれば，特性多項式が異なるということが言える．

2.2 出 力 関 数

TinyMT の出力関数は，一つの点で MT と異なっている．出力関数は $\mathbb{F}_2$-線形ではない．これは出力の線形性を低下させるため，より直接的に言えば，TestU01 の Bigcrush の線形性テストをパスさせるためである．

```
void next_state(){
    extern uint32_t status[4];
    extern uint32_t mat1, mat2;

    uint32_t x = (status[0] & 0x7fffffff) ^ status[1] ^ status[2];
    uint32_t y = status[3];
    x ^= (x << 1);
    y ^= (y >> 1) ^ x;
    status[0] = status[1];
    status[1] = status[2];
    status[2] = x ^ (y << 10);
    status[3] = y;
    // xB の部分
    if (y & 1) {
        status[1] ^= mat1;
        status[2] ^= mat2;
    }
}
```

図 2 状態遷移関数
Fig. 2 State Transition Function

```
uint32_t tinymt32_temper() {
    extern uint32_t status[4];
    extern uint32_t tmat;

    uint32_t t0 = status[3];
    uint32_t t1 = status[0] + (status[2] >> 8);
    t0 ^= t1;
    if (t1 & 1) t0 ^= tmat; // xT の部分
    return t0;
}
```

図 3 出力関数
Fig. 3 Output Function

C99 による出力関数プログラムを**図 3** に挙げた．なお，プログラム中の外部変数 tmat はテンパリングパラメータである．

2.2.1 出力列の周期

状態遷移関数が素数の最大周期 $P = 2^{127} - 1$ を持つ場合，状態空間の任意のビットも同じ周期 P を持つといえる．そして出力関数が内部状態を持たないならば，出力列の周期は状態遷移の周期 P の約数であり，同じ数が出力され続けるのでなければ，P そのものである．

Vol.2011-HPC-131 No.3
2011/10/6

表 1 BigCrush 検定結果
Table 1 Result of BigCrush

parameter	seed									
mat1,mat2,tmat	0	10	20	30	40	50	60	70	80	90
8f7011ee,fc78ff1f,3793fdff	74	22		12,22						48
877810ef,fc38ff0f,c7fb7fff				90			51		92	
837c106f,fc18ff07,eeb9bdff			22	47		77				
718e0e31,fb88fee3,11dbffff			45			80				89
50af0a15,fa80fea1,9ddc99ff		25,99	78				86	25		75
14eb029d,f8a0fe29,46f3ebff		78			47			51		
0bf4017e,f858fe17,e8cfecfd		80			67	77	11			23
09f6013e,f848fe13,52a0f5ff										
e51b1ca3,f720fdc9,f8ebffff							91			13
65980cb3,eb38facf,cc3b75ff	51				74					

これは 32 ビット中のどのビットについても言える．TinyMT の出力については，v ビット精度均等分布次元（後述 定義 3.2）において，$k(32) >= 1$ を確認できるので，どのビットについても周期 P が言える．

2.3 性能評価

2.3.1 統計的検定

TinyMT の出力を TesutU01[4] の BigCrush で検定した．BigCrush は 160 種のテストから構成される．BigCrush では p-value が [0.001, 0.9990] の範囲外の場合を fail としている．特に，1.0×10^{-300} 未満を eps，1.0×10^{-15} 未満を eps1 として，eps， eps1， 1-eps，1-eps1 を特に問題としている．**表 1** は TinyMT の BigCrush テストの結果である．

TinyMTDC(§ 3) に ID 0 を指定して生成された最初の 10 組のパラメータと seed を 10 個使用し，計 100 回 BigCrush テストを行った．表中の番号は，fail と判定されたテスト番号である．最悪の p-value は $1 - 2.2 \times 10^{-6}$ であり，eps， eps1， 1-eps，1-eps1 に該当する p-value は観測されなかった．10 組のパラメータのいずれも seed によっては一つか二つのテストに fail になっているが，別の seed ではパスしている．この結果，テストした 10 組のパラメータについて，BigCrush テストにパスしたと言える．

なお，TinyMT は暗号用の疑似乱数として設計されていない．BigCrush に合格することは暗号用として使用出来ることを意味しない．

2.3.2 GPU 上での速度比較

TinyMT を NVIDIA の GPGPU 環境である CUDA[9] で実装し，速度を MTGP， CURAND[8] と比較した．

CUDA は NVIVIA の GPGPU 環境であり，GPU 上での並列実行の単位をスレッドといい，並列に実行されるプログラムをカーネルという．GPU 上のメモリは，高速なローカルメモリ，シェアドメモリとやや低速なデバイスグローバルメモリがある．

MTGP は周期 $2^{11213} - 1$ の MTGP11213 を使用した．MTGP は 256 スレッドでシェアドメモリを状態空間として使用し，ひとつの疑似乱数生成器を構成する．その生成器を最大 108 使用した．異なる生成器は異なる特性多項式を持つ．

CURAND は Marsaglia の xorwow[5] の実装である．つまり，$\mathbb{F}_2$-線形生成器 xorshift と乗数 1 の線形合同法である Weyl 生成器を 2^{32} を法とする加算で結合したものである．その周期は $2^{192} - 2^{32}$ である．速度計測プログラムでは 1 スレッド 1 生成器で，状態空間はローカルメモリにとって，速度を計測した．CURAND の各生成器の特性多項式は同一だが，初期状態は十分離れたものとした．

TinyMT は 1 スレッドで 1 生成器とし，状態空間をローカルメモリにとった．各生成器は特性多項式が異なるようにした．比較に使用した GPU は NVIDIA GeForce GT120 と GeForce GTX 260 である．

表 2 は GPU 上で，5×10^7 個の [0,1) 区間に分布する単精度浮動小数疑似乱数の生成に要した時間をミリ秒単位で計測した結果である．

array は出力された疑似乱数をすべてデバイスグローバルメモリに書き込んだ場合，sum は出力された疑似乱数をスレッド単位で合計し，結果をデバイスグローバルメモリに書き込んだ場合の速度である．arry は疑似乱数生成とアプリケーションプログラムが別 kernel となる場合を想定した測定であり，sum は疑似乱数生成とアプリケーションプログラムが同一 kernel である場合を想定した測定である．MTGP は出力をデバイスグローバルメモリに書き込むような使い方のみを想定しているため sum は計測していない．

デバイスグローバルメモリへの書き込みは相対的に遅いため，TinyMT,CURAND 共に sum の速度が array の速度に勝っている．array では，TinyMT は比較した三つの疑似乱数生成器の中で最も遅く，GT120 で 35.04ms GTX 260 で 5.34ms かかっている．sum では，TinyMT の処理速度は GT120 で 34.21ms， GTX 260 で 4.81ms と向上している．

なお，CURAND は BigCrush 検定に合格しないことが報告されている[1]．私たちも CURAND が，複数の seed で 7 CollisionOver， 27 SimpPoker， 81 LinearComp(r=0) の検定に不合格となることを確認した．

また，MTGP11213 は BigCrush の 80 LinearComp(r=0)， 81 LinearComp(r=29) の検定に不合格となる．これは MTGP が $\mathbb{F}_2$-線形生成器であるためである．

Vol.2011-HPC-131 No.3
2011/10/6

表 2 GPU 上での速度比較
Table 2 Comparison of Speed on GPUs

	MTGP	TinyMT		CURAND	
	array	array	sum	array	sum
GeForce GT 120	32.81ms	35.04ms	34.21ms	20.58ms	19.36ms
GeForce GTX 260	4.75ms	5.34ms	4.62ms	3.03ms	2.88ms

```
void setup_param(uint32_t num) {
    uint32_t work = seq ^ (seq << 15) ^ (seq << 23);
    work <<= 1;
    mat1 = (work & 0xffff0000) | (id & 0xffff);
    mat2 = (work & 0xffff) | (id & 0xffff0000);
    mat1 ^= mat1 >> 19;
    mat2 ^= (mat2 << 18) | 1;
}
```

図 4 状態遷移パラメータ探索
Fig. 4 Serching State Transition Parameters

3. TinyMTDC

大規模並列シミュレーションにおいては，独立性の高い疑似乱数系列を多数使用することが望ましい．松本・西村による Dynamic Creator[7] は，特性多項式の異なる MT を多数生成するアルゴリズムである．

私たちは，特性多項式の異なる TinyMT のパラメータを多数生成する TinyMTDC を作成した．TinyMTDC はユーザの指定した ID に対して指定された個数のパラメータを生成する．

3.1 状態遷移パラメータ探索

状態遷移パラメータ mat1, mat2 は，ユーザ指定の ID と内部カウンタ seq より計算する．seq の初期値は 0x7fffffff であり，指定された個数のパラメータを生成するか，0 になるまでカウントダウンする．C99 によるプログラムを**図 4** に挙げる．

パラメータが決まると，Number Theory Library (NTL)[12] に実装された Berlekamp-Massey 法によって疑似乱数列の最小多項式を計算し，最小多項式が 127 次の既約多項式でない場合は，パラメータを捨てて，seq をカウントダウンし次のパラメータで既約多項式を探す．127 次既約の場合，最小多項式は特性多項式と一致する．

なお，Berlekamp-Massey 法によらずに線形代数によってパラメータから特性多項式を求めることも出来るが，実行速度においてやや劣っていたため，採用しなかった．

3.2 テンパリングパラメータ探索

テンパリングパラメータは，v ビット精度均等分布次元を評価基準として探索を行う．はじめに v ビット精度均等分布次元の定義を示す．

定義 3.1 (k 次元均等分布) 周期 $P = 2^p - 1$ の v ビット整数列

$$\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{P-1}, \mathbf{x}_P = \mathbf{x}_0, \ldots$$

は, 連続する k 個の組

$$(\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+k-1}), \quad i = 0, \ldots, P-1$$

が，すべての可能な kv ビットパターンに渡って均等に分布するとき，k **次元均等分布**するという．ただし，全部ゼロというパターンは一回少なくてよいとする．つまり，v ビットのワードからなる k 個の組が数列に同じ回数だけ出現する（ただし，連続する k 個のゼロは一回少なくてよい）場合である．

定義 3.2 (v-ビット精度 k 次元均等分布) w ビット整数の周期列は，最上位から v ビットの整数が k 次元均等分布するとき，v **ビット精度** k **次元均等分布する**という．

v ビット精度 k 次元均等分布する数列の k の最大値を v **ビット精度均等分布次元** $k(v)$ で表す．$k(v)$ の値が大きいほど v ビット精度での均等分布次元が高いことを示している．

$P = 2^p - 1$ とすると，

$$k(v) \le \lfloor p/v \rfloor.$$

という自明な上限がある．$d(v)$ は上限と $k(v)$ の差である．

$$d(v) := \lfloor p/v \rfloor - k(v) \ge 0,$$

Δ は $d(v)$ の v に渡る合計である．

$$\Delta := \sum_{v=1}^{w} d(v).$$

Δ の値が小さいほど，$k(v)$ $(v = 1, \ldots, w)$ の観点から，疑似乱数生成器が最適に近いことを示している．

上記定義の v ビット精度均等分布次元は，疑似乱数生成器の状態遷移関数および出力関数が $\mathbb{F}_2$-線形写像の場合，Lenstra の lattice reduction によって，計算することが可能である．

TinyMT の出力関数は線形写像でないのでそのままでは v ビット精度均等分布次元を計算することができない．そこで，図 3 の 6 行目を

```
uint32_t t1 = status[0] ^ (status[2] >> 8);
```

と変更すると $\mathbb{F}_2$-線形写像となる．これに対して v ビット精度均等分布次元 $k(v)$ を計算し，Δ が小さくなるようにテンパリングパラメータを探索した.（なお，もともとの出力関数の v ビット精度均等分布次元を計算することは困難であるとおもわれる．私たちは，この二つの出力関数の振る舞いが比較的近いことを仮定して，第 2 の出力関数に関して均等分布次元が高いパラメータを生成し，それを第 1 の出力関数に対して使っている．この点においては数学的裏付けはない）

テンパリングパラメータの探索は，２段階からなる．

- ステップ 1：下位 9 ビットのテンパリングパラメータを探索する．
- 32 ビット変数 tmp に 0 をセットする．

for $j = 0$ to 5 step 5 **do**

- $e = \min(j + 5, 9)$
- tmp の 31-j ビットから 31-e ビットまでのすべてのビットパターンを生成する．
- $k'(1)$ から $k'(e)$ までを計算する．ここで，$k'(v)$ は下位ビットからの v ビット精度均等分布次元である．
- dimension defect $d'(1) + d'(2) + \cdots + d'(e)$ の合計を最小にするビットパターンの中でハミングウェイトが最大のビットパターンに固定する．ここで，$d'(v)$ は下位 v ビットの dimension defect である．

end for

- ステップ 2：上位 23 ビットのテンパリングパラメータを探索する．

for $j = 0$ to 18 step 6 **do**

- $e = \min(j + 6, 23)$
- tmp の j ビットから e ビットまでのすべてのビットパターンを生成する．ここでは下位 9 ビットは変更されない．
- それぞれのビットパターンについて $k(1), k(2), \ldots k(e)$ を計算する．
- dimension defects $d(1) + d(2) + \cdots + d(e)$ の合計を最小にするビットパターンの中でハミングウェイトが最大のビットパターンに固定する．

end for

TinyMTDC の実装において，v ビット精度均等分布次元の計算に使用したアルゴリズムは原瀬の PIS 法[2] であり，これは Lenstra の lattice reduction を改良したものである．

表 3　TinyMTDC による 65536 パラメータ生成
Table 3　65536 parameter generation by TinyMTDC

	$\Delta = 0$	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	time	tried numbers
ID = 0	59744	5711	80	1	0	57min	2078625
ID = 1	59680	5775	80	1	0	57min	2090382
ID = 2	59746	5689	101	0	0	58min	2066520
ID = 3	59645	5801	88	2	0	57min	2088467
ID = 4	59758	5686	92	0	0	57min	2080880

3.3 TinyMTDC の性能

表 3 に TinyMTDC を使用して 65536 個のパラメータを生成した結果を示した．ID はユーザーが任意に指定出来る 32 ビット符号なし整数である．Δ は定義通りに上位ビットから計算した v ビット精度均等分布次元の Δ であって，ステップ 1 で計算した下位ビットからのものではない．

ID = 0 では，65536 個のパラメータ中，59744 個のパラメータが $\Delta = 0$ つまり最大の v ビット精度均等分布次元となったことを示している．他の ID においても，多くのパラメータが $\Delta = 0$ となっている．また，Δ が 4 以上のパラメータは探索されなかった．

このことは大きな Δ を持つパラメータが探索されないことを保証するものではないが，TinyMT web ページ[10] にある TinyMTDC のプログラムでは，Δ の最大値を指定してそれを超えるパラメータは出力しないように指定することができる．

time の欄には 65535 個のパラメータ探索にかかった時間が書かれている（Apple MacPro Intel Xeon 5500 2.26GHz）．65536 個の探索に約 1 時間かかることがわかる，1 秒間では 19 個近くのパラメータを探索したことになる．なお，このパラメータ探索においては 5 プロセスをバックグラウンドで同時に実行しているので，5 プロセス合計では 1 秒間に 94 個のパラメータを探索したことになる．

最後の欄は，65536 個の既約多項式を探索する過程で生成された多項式の数である．別の言い方をすると，内部カウンタの減算された数である．ここから，内部カウンタがゼロになるまでに生成可能な既約多項式を推定すると約 2^{26} 個になる．ユーザーの指定する 32 ビット ID のそれぞれについて 2^{26} 個の既約多項式が生成可能なので，ID を変えることによって TinyMTDC は約 2^{58} の独立なパラメータを生成可能と考えられる．

4. まとめ

状態空間の小さな高性能疑似乱数生成器 TinyMT を作成した．TinyMT には以下の特徴

がある．

- 周期 $2^{127}-1$．
- 状態空間 127 ビット（使用メモリは 16 バイト）．
- パラメータ 12 バイトを含めても 28 バイトしかメモリを使用しない．
- TestU01 の BigCrush 検定に合格する．
- パラメータ生成器 TinyMTDC によって最大 2^{58} 個程度のパラメータを生成できる．

また CUDA における速度計測では以下の結論が得られた．

- デバイスグローバルメモリに疑似乱数を生成する方法では，MTGP や CURAND より遅い．
- スレッド毎に疑似乱数の合計を求める方法では，MTGP より速く，CURAND より遅い．

なお，線形出力関数を使用して，v ビット精度均等分布次元を最適化するようにテンパリングパラメータを定め，実際の疑似乱数生成においては非線形な出力関数を使用するという方法を採用することに理論的な裏付けはない．しかしながら，BigCrush 検定をパスしていることがこの手法の有効性を示していると思われる．

謝辞　本研究は科研費（21654004,23244002,19204002,21654017）の助成を受けたものである．

参　考　文　献

1) Fabien: XORWOW L'Ecuyer TestU01 results (2011). http://chasethedevil.blogspot.com/2011/01/xorwow-lecuyer-testu01-results.html.
2) Harase, S.: An efficient lattice reduction method for $\mathbf{F}_2$-linear pseudorandom number generators using Mulders and Storjohann algorithm, *Journal of Computational and Applied Mathematics*, Vol. 236, No. 2, pp. 141–149 (online), DOI:http://dx.doi.org/10.1016/j.cam.2011.06.005 (2011).
3) Knuth, D. E.: *The Art of Computer Programming. Vol.2. Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 3rd edition (1997).
4) L'Ecuyer, P. and Simard, R.: TestU01: A C library for empirical testing of random number generators, *ACM Transactions on Mathematical Software*, Vol.15, No.4, pp. 346–361 (2006).
5) Marsaglia, G.: Xorshift RNGs, *Journal of Statistical Software*, Vol.8, No.14, pp. 1–6 (2003).
6) Matsumoto, M. and Nishimura, T.: Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, *ACM Trans. on Modeling and Computer Simulation*, Vol.8, No.1, pp.3–30 (1998). http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.
7) Matsumoto, M. and Nishimura, T.: Dynamic Creation of Pseudorandom number generator, *Monte Carlo and Quasi-Monte Carlo Methods 1998*, Springer-Verlag, pp. 56–69 (2000). http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/DC/dc.html.
8) NVIDIA: CURAND Library (2010). http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CURAND_Library.pdf.
9) NVIDIA corp.: *NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Ver 1.0*, NVIDIA, 2701 San Tomas Expressway, Santa Clara, CA 95050, 1.0 edition (2007).
10) Saito, M. and Matsumoto, M.: TinyMT (2011). http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/TINYMT/index.html.
11) Saito, M. and Matsumoto, M.: Variants of Mersenne Twister Suitable for Graphic Processors, *ACM Transactions on Mathematical Software*, Vol.Submitted (online), available from ⟨http://www.math.sci.hiroshima-u.ac.jp/˜m-mat/MT/MTGP/index.html⟩.
12) Shoup, V.: NTL: A Library for doing Number Theory. http://www.shoup.net/ntl/.