«Պատահական թվերի գեներատորներ»

Նշանյան Արեգ

Նախաբան

Պատահականությունները և դրանց դերը մեր կյանքում, ինչպես նաև համակարգչային գիտության մեջ շատ հետաքրքիր և ինչ-որ չափով հակասական թեմա է, որը տասնամյակներ շարունակ ֆիլտրվել է:

Համակարգչային գիտության մեջ պատահականությունը նույնքան կարևոր դեր է խաղում, ինչքան մեր առօրյայում: Շատ ալգորիթմներ և համակարգչային ծրագրեր հիմնվում են պատահական թվերի վրա որպես այսպես ասած «պատահականության» աղբյուր: Օրինակ՝ կրիպտոգրաֆիայում պատահական թվերն օգտագործվում են հաղորդագրությունների գաղտնագրման և վերծանման համար՝ ապահովելով հաղորդակցության ապահով միջոց: Խաղերում պատահականությունն օգտագործվում է խաղի իրադարձությունները իրական կյանքին հնարավորինս մոտեցնելու համար:

Չնայած դրա լայն կիրառմանը, պատահականության հայեցակարգը դեռևս բանավեճի առարկա է: Որոշ փիլիսոփաներ պնդում են, որ իրական պատահականություն գոյություն չունի, և որ բոլոր իրադարձությունները, ի վերջո, որոշվում են հիմքում ընկած պատճառներով և օրինաչափություններով: Մյուսները, սակայն, պատահականությունը տեսնում են որպես տիեզերքի հիմնարար ասպեկտ, նրա բնորոշ անորոշությամբ և անկանխատեսելիության արտացոլումով:Որոշ մասնագետների կողմից վեճի առարկա է դարձել նաև այն, թե իրականում լինում են պատահական թվեր թե նրանք բոլորը ինչոր օրինաչափությունների արդյունքեն:

Համակարգչային գիտության մեջ դիտարկվում են երկու տիպի պատահակն թվերի գեներատորներ

1) (PRNG) - Կեղծ պատահական թվերի գեներատորների ընտանիք
2) (TRNG)- իրական պատահական թվերի գեներատորներ:

PRNG-ները ալգորիթմներ են, որոնք ստեղծում են պատահական թվերի հատկություններ ունեցող հաջորդականություն՝ հիմնված սկզբնական սերմի արժեքի վրա:

TRNG-ները պատահական թվեր են ստեղծում՝ հիմնված ֆիզիկական գործընթացների վրա, ինչպիսիք են մթնոլորտային աղմուկը կամ ռադիոակտիվ քայքայումը, այդ թվերը համարվում են իսկապես պատահական և հաճախ օգտագործվում են անվտանգային բարձր պահանջներ ունեցող ծրագրերում, օրինակ ծածկագրության մեջ:

Այս հոդվածում մենք կքննարկենք յուրաքանչյուր տեսակի գեներատորի ուժեղ և թույլ կողմերը, ինչպես նաև դրանց օգտագործումը տարբեր ծրագրերում:

Առաջին գլուխ

PRNG-ի էությունը և պարզագույն օրինակ

Տարբեր PRNG ալգորիթմներ ունեն իրենց լավ ու վատ կողմերը, որոնցով որոշվում է այն ոլորտը որտեղ այդ գեներատորներ կօգտագործվեն: Բոլոր ալգորիթմները ունեն իրենց առանձնահատկությունները: Դիտարկենք Linear Congruential prngի օրինակը, որը պարզ բայց արդյունավետ գեներատոր է, ընդհանուր պատկեր ստեղծելու համար:

Դիտարկենք $X_{i+1}=X_i*2+1$ ալգորիթմը, այն Linear Congruential ալգորիթմի պարզ տեսակն է: ամեն մի ալգորիթմի հիմքում ընկած է այսպես կոչված սերմը, մեր նշած օրինակում այն ինֆորմատիցոն էնտրոպիան է(էնտրոպիայի մասին կխոսենք ապագա գլուխներում): Կարելի է նկատել որ այս ալգորիթմի տրամաբանությունը շատ հեշտ է գուշակել, որովհետև թվերը ցույց են տալիս բացահայտ էքսպոնենցիալ աճ: Այդ թերությունից ազատվելու համար կարելի է ավելացնել թվի մոդուլը ըստ որևէ թվի, Օրինակ սա՝

$$X_{i+1}=(X_i*2+1)\mod 13$$

Այս փոփոխությունը ավելի է դժվարացնում հաջորդ գեներացված թվի գուշակելու հնարավորությունը, բայց միևնույն ժամանակ կրճատում է պարբերությունը, ինչը նշանակում է որ գեներացվող թվերի բանակն է պակասում:

Ընդհանրացնելով կստանանք ալգորիթմ որը կունենա այսպիսի տեսք

$$X_{i+1}=(X_i*a+c)\text{mod}M$$

Այժմ Ժամանակակից կիրառական մաթեմատիկայի խնդիրներից մեկը ճիշտ թվեր գտնելն է a, c և M-ի համար, որոնք հնարավոր կդարձնեն մաքսիմալ երկարացնել կրկնման ցիկլը և իջացնել թվերի կանխատեսելիությունը: Այս պահի դրությամբ Java ծրագրավորման լեզվում որպես հիմնական օգտագործվում են a=25214903917 c=11 $M=2^{48}$ թվերը: Որոնք ըստ լեզվի ստեղծողների այս պահին ամենահաջողն են:

Երկրորդ գլուխ

Մի քիչ TRNG-ների մասին

TRNGները սարքերի տեսակներ են, որոնք օգտագործելով բնական երևույթները գեներացնում են իրական պատահական թվեր: Հետաքրքրություն PRNGների այդ թվերն հնարավոր չէ գուշակել, սակայն պետք է շեշտել որ որոշ CPRNGներ կարողացել են մաքսիմալ դժվարացնել այդ գործը, այդպիսի ալգորիթմի մասին կխոսենք 4րդ գլխում: Այդպիսի սարքերը օգտագործվում են կոդավորման շատ բարձր պահանջ ունեցող ոլորտներում: TRNGները օգտագործում են տարբեր ֆիզիկական երևույթներ, որոնցից են ռադիոակտիվ քայքայումը կամ մթնոլորտային աղմուկը: Այս պրոցեսները կարող են չափվել և օգտագործվել որպես էնտրոպիայի աղբյուր, որը մեծ հաշվով թվերի պատահականության ցուցանիշ է: Օրինակ՝ ինչքան մեծ է էնտրոպիան այնքան թվերը պատահական են և հակառակը: Համակագը էնտոպիան ստանալուց հետո անցնում է թվերի գեներացմանը: Իհարկե կասկածներ չկան որ TRNGները գեներացնում են իրական պատահական թվեր և շատ ավելի անվտանգ են բոլոր ոլորտներում օգտագործելու համար, բայց միևնույն է նրանք շատ թանկ են և դժվար գործածելի: Այդ իսկ պատճառով պետք է մտածել արդյոք այդքան պարտադիր է օգտագործել այդ թանկ մեխանիզմը եթե կան CPRNGներ որոնք իհարկե չեն տրամադրում այդպիսի անվտանգություն, բայց միքանի վուլային պաշտպանության շնորհիվ կարող են դիմակայել բոլոր այսօրվա հարձակումներին:

Երրորդ գլուխ

RNG-ների օգտագործման ոլորտներ

Մենք կխոսենք այն ոլորտից որտեղ պատահական թվերի գեներատորները շատ են օգտագործվում օրինակ՝ խաղերց: Ժամանակակից խաղերը դարձել են շատ մոտ մեր կյանքին թե՛ գրաֆիկական և թե՛ ֆիզիկական առումներով: Խաղերում կատարվածը հնարավորինս մոտը է իրական կյանքին սկսած գենի կրակոցից և վերջացրած տարբեր տեսակ ծառերի տարածվածությամբ: Այդ արդյունքին հասնելու համար լայնորեն օգտագործվում են պատահական թվերի գեներատորները: Որպես օրինակ այդպիսի գեներատոր կարող է համարվել xorshiftը որը ունի բավականին լավ հատկություններ կոնկրտ խաղերի համար:

Xorshift-ը շատ արագ և արդյունավետ է, իր պարզության շնորհիվ: Այստեղ օգտագործվում է միայն երկու տեսակ օպերացիա, bitwise leftshift և bitwise rightshift, դրանցով այն աշխատում է առանձին բիթերի հետ(ավելի մանրամասն նկարագրությունը առաջիկայում): Առաջիկայում մենք կնկարագրենք նաև mersenn twisterի ալգորիթմը և ցույց կտանք որ xorshiftը ավելի պարզ է, որովհետև անում է bitwise օպերացիաների ավելի քիչ ցիկլներ: Բացի դրանից այստեղ չեն օգտագործվում այլ տեսակ ալգորիթմներ, որովհետև կոնկրետ այս ալգորիթմում առաջնայինն է աշխատանքի պարզությունը և արագությունը, իսկ կողմնակի ալգորիթմների օգտագործումը լավ է միայն անվտանգությունը բարձրացնելու համար:

Օրինակ եթե խաղերում օգտագործվեր նախապես ստեղծված քարտեզ, դա շատ մեծ բեռ կլիներ համակարգչի վրա, այդ իսկ պատճառով խաղ ստեղծողները աշխարհը բաժանում են տարբեր մասերի, և ամեն մի քարտեզի մասը ստեղծվում է այն պահին երբ խաղացողը մտնում է նրա մեջ: Այդ պարզ բայց արդյունավետ մեխանիկայի շնորհիվ համակարգիչը պարտավորված չի լինում բացել ողջ քարտեզը, որը շատ կծանրաբեռներ նրան, այլ ստեղծում է քարտեզի մասերը ըստ անհրաժեշտության:

```cpp
#include <iostream>

class Xorshift32 {
public:
    Xorshift32(std::uint32_t seed) : state_(seed) {}

    std::uint32_t operator()() {
        state_ ^= state_ << 13;
        state_ ^= state_ >> 17;
        state_ ^= state_ << 5;
        return state_;
    }

private:
    std::uint32_t state_;
};

int main() {
    Xorshift32 rng(12345);  // create a new RNG with seed 12345

    // generate and print 10 pseudorandom numbers
    for (int i = 0; i < 10; ++i) {
        std::cout << rng() << std::endl;
    }

    return 0;
}
```

Այս օրինակում կարող եմ տեսնել որ 8րդ տողում օգտագործված է bitwise leftshift օպերացիան 13 արժեքով, այնուհետև bitwise rightshift 17 արժեքով և վերջում կրկին bitwise leftshift 4 արժեքով: Որպես սկզբնական սերմ օգտագործվել է 1234 թիվը(տես 19րդ տողը) և վերջում ստացել 10 թվերի հաջորդականություն՝

```
3336926330
1697253807
2816511904
1955480042
718842323
3283620450
4285686168
3680911160
2359762552
2188185676
```

Ամփոփելով կարելի է ասել որ xorshiftը շատ պարզ և արագ գեներատոր է որը լայնորեն օգտագործվում է այն ոլորտներում որտեղ կարևոր է մեայն գեներացմանը արագությունը:

Անցնենք մյուս ոլորտին որտեղ PRNG-ները նույնպես շատ ակտիվ են օգտագործվում՝ cryptography(հայ. ծածկագրություն):

Այս ոլորտը ունի կտրուկ տարբեր պահանջներ խաղերից, ոլորտի պետական կարևորության պատճառով: Այս առումով տարբեր մեծ պետություններ սահմանել են մի բանի կանոններ որոնք պետք է հաշվի առնվեն PRNG-ներ ընտրելուց: Դիտարկենք Գերմանիայի կողմից ստեղծված չափանիշները:

K1- Թվերի հաջորդականությունը պետք է ունենա կրկնվելու փոքր հավանականություն

K2- Թվերի հաջորդականությունը չպետք է տարբերվի, համածայն հատուկ ստատիստիկ գործընթացների, ամբողջություն իրական պատահական թվերից:

K3- Ցանկացած ներխուժողի համար պետք է անհնար լինի ելնելով այս գեներատորի ինչ-որ ենթահաջորդականությունից գուշակել որևէ հաջորդ կամ նախորդ արժեք:

K4- Ցանկացած ներխուժողի համար պետք է անհնարին լինի գեներատորի ինչ-որ ներքին վիճակից ելնելով գուշակել այլ արժեքներ կամ այլ ներքին վիճակ(անգլ. Inner state):

Այս կետերը, հատկապես վերջին երկուսը պարտադիր են համարվում Տեղեկատվական անվտանգության դաշնային գրասենյակի(գեր. Bundesamt für Sicherheit in der Informationstechnik) համար: Ինչը նշանակում է որ նույնիսկ իմանալով ինֆորմացիա անցյալ ստեղծված թվերի մասին, միևնույն է անհնար է գուշակել այն թվերը որոնք կգեներացվեն ապագայում:

Իհարկե դժվար կլինի ասել մեկ կոնկրետ CPRNG(Գաղտնագրորեն ապահով կեղծ պատահական թվերի գեներատոր)-ի օրինակ, որովհետև այնտեղ նույնպես կան տարբեր ընտրության չափորոշիչներ: Սակայն կա մի շատ լավ օրինակ՝ Yarrow algorithm-ը, որը օգտագործվում է հայտնի «mac os x» օպերացիոն համակարգում:

Yarrow algorithm-ը CPRNG-ի տեսակ է երբ գեներացված թիվը անցնում է միաճանի փուլ:

էնտորպիայի պահեստ. Ամենասկզբում ստեղծվում են էնտրոպիայի պահեստներ(անգ. entropy pool), որոնք իրենց հերթին բաղկացած են երկու մասից, (fast pool) այս մասը պատասխանատու է էնտրոպիայի արագ փոփոխության համար, իսկ (slow pool)-ի մեջ էնտրոպիան փոփոխվում է ավելի ուշ և մաշտաբային:

Վերասերմացման մեխանիզմ. Այս փուլում, երբ ալգորիթմը հասկանում է որ ժամանակն է փոխել հիմնական սերմ-ը որով գեներացվում են թվերը, վերցվում է նոր էնտրոպիայի արժեք պահեստներից(այլ կերպ ասաց hash memory-ից): Այդ ամենը պետք է արվի այնպես որպեսզի հնարավոր չլինի գուշակել նոր ստեղծված վերասերմացման բանալին:

Կառավարում վերասերմացմամբ. Այս փուլը կարևոր է նրանով, որ վերասերմացման մեծ հաճախականությունը բարձրացնում է վտանգը կրկնվող հարձակումների առաջ, բայց միևնույն ժամանակ վերասերմացման երկար շրջանը հարձակվողին տալիս է ավելի շատ ինֆորմացիա ընթացիկ բանալու մասին:

Գեներացման համակարգ.Այս ամենից հետո գալիս է գեներացման համակարգը, կոնկրետ այս ալգորիթմում օգտագործվում է yarrow 160 համակարգը, որը բաղկացած է երկու մասից:

A)SHA1-ը որը իրենից ներկայացնում է Հեշ գեներատորների շատ հին, բայց լայնորեն օգտագործվող ալգորիթմ:

B)Triple DES-ը երկրորդ մասն է, որը պատասխանատու է SHA1 կողմից գեներացված թվերի k պատիկ կոդավորկման համար: Գոյություն ունի երեք 3DES ենթաալգորիթմ, որոնցից ամենահայտնին է՝ DES-EDE3ը

$C = E_{k_3}(E_{k_2}^{-1}(E_{k_1}(P)))$ այսպիսի աշխատանքի ալգորիթմով, որտեղ k_1, k_2, k_3 բանալիներ են ամեն DES ֆայլի համար: Կամ այլ կերպ ասած ենթաալգորիթմը բաղկացած է կոդավորում-վերծանում-կոդավորում ֆայլերից, այսպանով հանդերձ 3DES ալգորիթմը տրամադրում է 3*56 բիթ երկարությամբ բանալի, որը անկախ k_1, k_2, k_3 օգտագործելիս տալիս է մաքսիմալ պաշտպանություն բոլոր տեսակ հարձակումների դեմ:

Իսկ որպես էնտրոպիայի աղբյուր yarrow algorithmը օգտագործում է զանազան աղբյուրներ, որոնցից են `

- Համակարգչի ֆիզիկական(hardware) մասի հետ կապված իրադարձությունները
- Օգտատիրոջ գործողությունները, ինչպիսիք են ստեղնաշարով գրելը կամ մկնիկի շարժումները
- Ցանցային փոփոխությունները,օրինակ՝ ցանցային ինտերֆեյսի վրա ստացված փաթեթները
- Համակարգչային փոփոխությունները, օրինակ ՝ պրոցեսների սկզբի և ավարտի ժամանակները:

Այս ալգորիթմը օգտագործում է այս և այլ բազմազան աղբյուրները որպեսզի ստեղծի կեղծ պատահական թվերի հոսք որոնք շատ դժվար է գուշակել:

Չորրորդ գլուխ

Ավելին PRNGների մասին

Առաջին գլխում մենք խոսեցինք «Linear Congruential prng» ալգորիթմի մասին, բայց կան նաև բազմաթիվ ուրիշ գեներատորներ, որոնք ունեն թե լավ և թե վատ կողմեր: Համակարգչային գիտություններում շատ է օգտագործվում այսպես կոչված Blum-Blum-Shub գեներատորը: Այն ամենասկզբում ընտրում է երկու պարզ թիվ p և q, որոնք բազմապատկելով ստանում է M-ը, այնուհետև ներքին պրոցեսների շնորհիվ համակարգը ստանում է սկզբնական սերմը և սկսում է բուն ալգորիթմը:

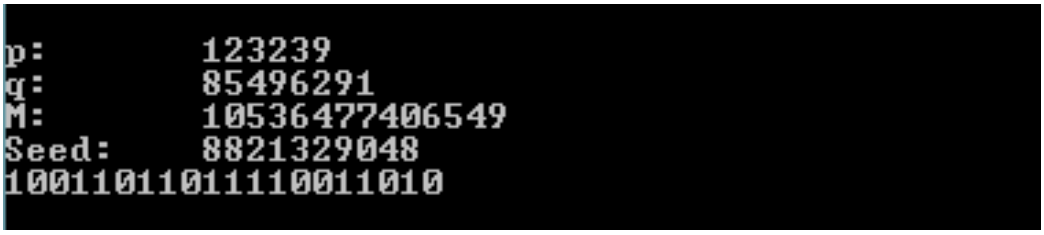
$x1=x^2 \bmod M$

$x2=x_1^2 \bmod M$

.
.
.
.

$xn=x_{(n-1)}^2 \bmod M$

Այնուհետև այս թվերը ձևափոխվում են բինար կոդի, երբ կենտ թվերը դառնում են 1, իսկ զույգերը ՝ 0: Այս ալգորիթմը համարվում է դետերմինիստական, այն է ալգորիթմ որը տալիս է եզակի և կանխորոշված արդյունք տվյալ մուտքերի համար: Սակայն միևնույն է նա օգտագործվում է գաղտնագրության մեջ և համարվում է CPRNG, մի պարզ պատճառով ՝ ունենալով m-ը որը p-ի և q-ի բազմապատիկն է միևնույն է թույլ չի տալիս գտնել վեր ճշված p-ի և q-ի արժեքները: Օրինակ ՝ եթե մենք ունենանք *1522605027922533360535618378132637429718068114961380688657908494580122963258952897654000350692006139* այս երկարություն ունեցող m-ը շատ դժվար կլինի բաժանել երկու պարզ բաղադրիչների: Նշված շնորհիվ BBS ալգորիթմը լայնորեն օգտագործվում է գաղտնագրության մեջ իր պաշտպանվածության շնորհիվ, բայց անգործածելի է սիմուլացիաները համար, դանդաղ աշխատելու պատճառով: Ալգորիթմի աշխատանքը ցույց կտանք python լեզվի օգնությամբ:



Այստեղ երևում են p-ն և q-ն, նաև ստացված M-ը: Այս պարագայում սերմի աղբյուր է հանդիսանում python լեզվի random.randint ֆունկցիան, իսկ 5րդ տողով գրվում է վերջնական արդյունքը:

Ուշացումով Ֆիբոնաչիի գեներատոր.

Այս ալգորիթմների ենթախումբը ստեղծվել է որպես այլընտրանք հնացած «Linear Congruential» գեներատորի համար: Այս գեներատորի արագ աշխատանքը ապահովված է նրանով, որ այնտեղ չկա բազմապատկման օպերատոր, բացի դրանից այնտեղի գեներացված թիվը կախված է մեկից ավել նախորդող արժեքներից:

$$X_{n+1} = (X_n + X_{n-1}) \mod (2^m),$$

Բանաձևը ունի այսպիսի տեսք, որը նայև կոչվում է Ֆիբոնաչիի թվեր: Գիտնականները հասկանալով որ կոնկրետ այս ալգորիթմը էֆֆեկտիվ չէ թվեր գեներացնելու համար առաջարկել է միիչ այլ մեթոդ, որը ունի հետևյալ տեսք

$$X_n = (X_{n-24} + X_{n-55}) \mod (2^m),$$

Կոնկրետ այստեղ $n \geq 55$, իսկ m-ն զույգ թիվ է: Այս դեպքում 55 և 24 թվերը ընտրած են դիտմամբ, այդ թվազույգի շնորհիվ X_n mod2 կրկնության պարբերությունը դառնում է $(2^{55}-1)$ երկարության, այս պատրբերությունը բավական է օգտագործման համար, բայց իհարկե կան այլ թվազույգեր որոնք տալիս են լավ արդյունք $(24, 55), (38, 89), (37, 100), (30, 127), (83, 258), (107, 378), (273, 607), (1029, 2281), (576, 3217), (4178, 9689), \ldots$

Ուղղակի ինչքան մեծանում են թվազույգի անդամները և գեներացնում են ավելի մեծ պարբերությամբ թվեր, այնքան երկարում է ալգորիթմի աշխատանքի ժամանակը

Mersenne twister

Եվ ահա հասանք ամենասիրված և տարածված գեներատորին: Այստեղ օգտագործվում է միիչ այլ համակարգ որտեղ օգտագործվում է 624 արժեք ունեցող վեկտոր, որպես սերմ,

այնուհետև սկսվում են միմյանց ցիկլ որոնցում bitwise և այլ հանրահաշվական օպերացիաների շնորհիվ ստանում ենք այսպիսի թվեր որոնք ունեն երեք հիմնական առավելություններ.

1) Գեներացված թվերի հաջորդականությունը շատ երկար է, ունենալով այսպիսի երկարություն 2^19937 - 1:
2) Նաև mersenne twisterը երկար պարբերության շնորհիվ գեներացնում է շատ որակյալ պատահական թվեր:
3) Եվ վերջինը, այս տեսակ գեներատորը շատ արագ է:

Այսթանի շնորհիվ mersenne twisterը ստացել է շատ արագ, հարմարավետ և վստահելի գեներատորի անուն, բայց ցավոք ունի մեկ առանձնահատկություն՝ գեներացված թվերը շատ հեշտ գուշակվող և վերծանվող են, ինչը թե՛ առավելություն է և թե՛ թերություն կախված օգտագործման ոլորտից: Այս տեսակ գեներատորները հիմնականում օգտագործվում են խաղերում և տարբեր տեսակ սիմուլացիաներում, օրինակ ՝ Մոնտե Կարլոյի մեթոդում, որտեղ օգտագործելով պատահական թվերը, կարելի է ստանալ ստույգ ինֆորմացիա տարբեր ալգորիթմների աշխատանքի մասին: Օրինակ կարելի է հաշվարկել միմյանց հազար հնարավոր դեպք խաղատան ինչ-որ խաղում, ստուգելով այնտեղի հաղթանակի շանսերը:

Այս գեներատորի հիմքում ընկած է մի ալգորիթմ որը կոչվում է linear feedback shift register, այն կատարում է թվերի բինար արժեքի տեղաշարժվում է մեկ բիթով դեպի աջ: Թվի սկզբնական արժեքը հաշվարկում ենք bitwise XOR ալգորիթմով:

Օրինակի համար վերցնում ենք չորս արժեք

B1	B2	B3	B4
1	0	0	1

Այնուհետև կատարվում է տեղաշարժը մեկով դեպի աջ, որից հետո ստանում ենք հետևյալ պատկերը

B1	B2	B3	B4
	1	0	0

B1-ը բաց է մնում, նոր արժեքը հաշվարկելու համար մենք օգտագործում ենք B2-ի և B3-ի արժեքները անցած վիճակից և կատարում bitwise XOR ալգորիթմը, ստանալով 1:

B1	B2	B3	B4
1	1	0	0

mersenne twisterում օգտագործվող անդրադարձ ֆունկցիան ունի հետևյալ տեսքը:

$$x_{k+n} := x_{k+m} \oplus \left((x_k{}^u \mid x_{k+1}{}^l) A \right) \qquad k = 0, 1, \ldots$$

Որտեղ
$\mathbf{X}_k{}^u$ = Ավագ w-r բիթ
$\mathbf{x}_k$ և $\mathbf{x}_{k+1}{}^l$ = կրթսեր w-r բիթ
r = Բառի անջատման կետը կամ lower bitmaskի թվերի քանակը: Lower bitmaskը դա այն թվերն են որոնք մենք ուզում ենք հեռացնել ալգորիթմից:
n = Կրկնության շարժը(անգլ. Degree of recurrence)
m = ամբողջ արժեք $1 \leq m < n$
A = w×w չափի մատրիցա {0,1} բազմության տարրերով
| = bitwise OR
⊕ = bitwise XOR

Ինչպես կարող ենք տեսնել այս ալգորիթմ լայնորեն օգտագործվում են bitwise operationները, որը ցածր մակարդակի գործողություններ կատարող ալգորիթմների ընտանիք է, նրանք փոփոխություններ են կատարում թվերի բինար արժեքների անհատական բիթերի հետ:

Ալգորիթմն օգտագործում է իր աշխատանքի համար հարմար A մատրիցա: Բազմապատկելուց հետո ստանում ենք այսպիսի պատկեր

$$\boldsymbol{x}A = \begin{cases} \boldsymbol{x} \gg 1 & x_0 = 0 \\ (\boldsymbol{x} \gg 1) \oplus \boldsymbol{a} & x_0 = 1 \end{cases}$$

որտեղ x_0-ն x շարքի ամենափոքր արժեքն է, իսկ a-ն ընտրված հաստատուն: Ավելացնել rightshiftի մասին:

1) Bitwise Left Shift: Այս ալգորիթմում թվի բինար արժեքը բազմապատկվում է 2^pով որտեղ p-ն դա shiftի արժեքն է: Right shiftի դեպքում արժեքը բազմապատկվելու տեղը պարզապես բաժանվում է 2^p-ի վրա:

1010 (decimal 10)
<< 3 (shift 3 positions to the left)
1010000 (decimal 80)

2) Bitwise XOR:

110011
010101
------- XOR
100110

3) Bitwise AND: Ամենավերջին ալգորիթմը հիմնականում օգտագործվում է գեներացված թվի երկարությունը կառավարելու համար: Այս դեպքում երբ բինար թվի արժեքը լինում է 1, վերջնական արդյունքում գրվում է 1, բոլոր այլ դեպքերում գրվում է 0:

1011001
1100100
------- (bitwise AND)
1000000

Non-linear feedback. Այս ալգորիթմը օգնում է ևվազեցնել գեներացված թվերի հաջորդականության մեջ հայտնաբերվող օրինաչափությունների թիվը: Ահա մի օրինակ թե ոնց է աշխատում այս ալգորիթմը՝

Ենթադրենք ունենք մեկ հաջորդականություն՝ $x_1, x_2, x_3, ..., x_n$: Այս հաջորդականության վրա կիրառելով ոչ գծային որևէ f ֆունկցիա, օրինակ ՝

$$f(x) = x^3 + 5$$

ալգորիթմը ստեղծում է նոր $y_1, y_2, y_3, ..., y_n$ հաջորդականություն: Այնուհետև նոր հաջորդականությունը զուգակցվում է սկզբնական հաջորդականության հետ՝ չկապված օգտագործելով XOR օպերացիան, այն է հաշվելով $xi \oplus yi$ արտահայտության արժեքը ամեն մի i-ի համար և ստանում է ելքային հաջորդականության հաջորդ արժեքը:

Նշված f ֆունկցիան կիրառելով x = 1010 մուտքային հաջորդականության վրա կստեղծի y = 1111101101 ելքային հաջորդականությունը: Այնուհետև երկու հաջորդականությունները կմիավորվեն XOR գործողության միջոցով՝ ելքային հաջորդականության հաջորդ արժեքը ստանալու համար:

Tempering. Ալգորիթմը սկսվում է 32-բիթանոց ամբողջ թվերի հաջորդականության ստեղծմամբ՝ օգտագործելով ստանդարտ Mersenne Twister ալգորիթմը: Mersenne Twister-ի ելքը այնուհետև բազմապատկվում է հատուկ հատկություններ ունեցող wxw մատրիցով, որտեղ w-ն դա բառի չափն է(անգլ. Word size): Մատրիցայով բազմապատկման ֆայլը բարելավում է ստեղծված պատահական թվերի վիճակագրական հատկությունները՝ ևվազեցնելով անցանկալի հարաբերակցությունները, որոնք կարող են գոյություն ունենալ հաջորդականության մեջ: Այլ կերպ ասած այս ֆայլը փոխում է թվի բինար արժեքի 0ների և 1երի հարաբերությունը ինչով մեծացնում է այդ թվերի ցրվածությունը:

$$y \equiv x \oplus ((x \gg u)\ \&\ d)$$
$$y \equiv y \oplus ((y \ll s)\ \&\ b)$$
$$y \equiv y \oplus ((y \ll t)\ \&\ c)$$
$$z \equiv y \oplus (y \gg l)$$

Մատրիցայի արժեքները ստեղծվում են հետևյալ տրամաբանությամբ: Որտեղ x-ը շարքի հաջորդ արժեքն է, y-ը ժամանակավոր միջանկյալ արժեք է, իսկ z-ն ալգորիթմից ստացված արժեքն է: $\ll$ և $\gg$ bitwise leftshift և bitwise rightshift օպերատորներն են, իսկ &-ը bitwise ANDն է:

Մատրիցային բազմապատկման հայլից հետո ելքը հետագայում փոխակերպվում է,bitwise leftshift և XOR ալգորիթմների համադրությամբ: Այս գործընթացը կիրառվում է հաջորդականությամբ ստեղծված յուրաքանչյուր թվի նկատմամբ:

Հինգերորդ գլուխ

QRNG(Quantum random number generator)

Մեծ հաշվով կարելի է Քվանտային թվերի գեներատորները նմանացնել իրական թվերի գեներատորներին, որովհետև երկուսնել օգտագործում են արտաքին ֆիզիկական սարքեր, ինչ-որ արժեք ստանալու համար, բայց նրանց մեջ կան մի‌ևանի որոշիչ տարբերություններ:

Նախ և առաջ TRNGները գեներացնում են վերջնական թիվ, որը կարելի է օգտագործել, իսկ քվանտային գեներատորները օգտագործվում են որպես թվային էնտրոպիայի աղբյուր, որից հետո օգտագործելով ավանդական prngները ստանում են‍ք վեջնարդյունքը: Քվանտային

գործոնը այս ամենում այն է, որ օգտագործելով քվանտային սուպերպոզիցիան հնարավոր է դառնում ստանալ պատահական թվեր շատ ավելի արագ և արդյունավետ քան TRNG-ները:

Մի փոքր խոսենք քվանտային սուպերպոզիցիայի մասին, որը կայանում է նրանում, որ երբ սովորական համակարգիչներում մենք ունենք սովորական բիթ, իսկ քվանտային համակարգիչներում դա այսպես կոչված qubit(quantum bit)ն է, որը ի տարբերություն դասական բիթերի, որոնք կարող են գոյություն ունենալ միայն 0 կամ 1 վիճակում, քյուբիթները կարող են գոյություն ունենալ միաժամանակ երկու վիճակներում սուպերպոզիցիայով: Սա նշանակում է, որ նրանք կարող են միաժամանակ մի քանի արժեքներ ներկայացնել, ինչը թույլ է տալիս քվանտային համակարգիչներին կատարել որոշակի առաջադրանքներ շատ ավելի արագ:

Բացի քվանտային սուպերպոզիցիայից կան այլ պրոցեսներ որոնք նույնպես կարող են օգտագործվել այդ տեսակ գեներատորներում, որոնցից են թունելային երևույթը,քվանտային խճճվածությունն և քվանտային ֆլուկտուացիան:

Այսպիսի շնորհիվ QRNGները ամենաշահավետ առաջարկն են պատահական թվերի գեներատորների մեջ, ունենալով PRNG-ների արագությունը և հարմարությունը միևնույն ժամանակ ունենալով TRNG-ների անվտանգությունը, ուղակի այս պահի դրությամբ այդ տեսակ գեներատորները դեռ ուսումնասիրվում և բարելավվում են, ինչի պատճառով այդ տեսակ գեներատորները դեռ չեն օգտագործվում:

Վերջաբան

Այս աշխատանքը նպատակ ուներ ընդհանուր պատկերացում տալ պատահական թվերի գեներատորների տարբեր տեսակների մասին: Քննարկումը սկսեցինք տարբեր ոլորտներում պատահական թվերի կարևորության մասին, այնուհետև ավելի մանրամասնորեն ուսումնասիրեցինք տարբեր հայտնի PRNG-ները, ինչպիսիք են Mersenne Twister-ը, Linear Congruential Generator-ը, BBS-ը և այլն: Մենք նաև հակիրճ անդրադարձ կատարեցինք TRNG-ներին և QPRNG-ներին, որոնք գնալով դառնում են ավելի տարածված իսկապես պատահական թվեր ստեղծելու ունակության շնորհիվ:

Մենք քննարկել ենք այն բոլոր նկատառումները, որոնք ըստ ինս պետք է ի նկատի ունենալ որոշակի առաջադրանքի համար գեներատոր ընտրելիս և ընդգծել ենք տարբեր տեսակի գեներատորների առավելություններն ու թերությունները: Իհարկե շատ դժվար կլիներ խոսել

բոլոր գեներատորների մասին նրանց ահռելի բանակի պատճառով, ուստի ընտրել եմ ամենաարդիական և էֆֆեկտիվ գեներատորները:

Callas, J. (2017, May 31). *Triple DES: How strong is the Data Encryption Standard?: TechTarget*. Security. Retrieved January 5, 2023, from https://www.techtarget.com/searchsecurity/tip/Expert-advice-Encryption-101-Triple-DES-explained#:~:text=Triple%20DES%20encryption%20process,a%20last%20time%20with%20K3.

Chapter 3 pseudo-random numbers generators - university of Arizona. (n.d.). Retrieved January 4, 2023, from https://www.math.arizona.edu/~tgk/mc/book_chap3.pdf

Hadlee, S. content from Z. (2023, January 1). *How random number generators are used in today's technology*. Varsity Online. Retrieved January 5, 2023, from https://www.varsity.co.uk/sponsored/how-random-number-generators-are-used-in-todays-technology

How do casinos control RTP (payouts rate) while using RNG (random numbers generator)? how do they still stay profitable and provide rando... Quora. (n.d.). Retrieved January 5, 2023, from https://www.quora.com/How-do-casinos-control-RTP-payouts-rate-while-using-RNG-random-numbers-generator-How-do-they-still-stay-profitable-and-provide-randomness-at-the-same-time

Pandya, S. (n.d.). *Difference between TRNG or PRNG? | Researchgate*. ResearchGate. Retrieved January 4, 2023, from https://www.researchgate.net/post/Difference_between_TRNG_or_PRNG

Pseudo random number generator (PRNG). GeeksforGeeks. (2022, December 30). Retrieved January 5, 2023, from https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/

Pseudo-random number generation. cppreference.com. (n.d.). Retrieved January 5, 2023, from https://en.cppreference.com/w/cpp/numeric/random

Pseudo-random number generators. Cryptography - Pseudo-Random Number Generators. (n.d.). Retrieved January 5, 2023, from https://crypto.stanford.edu/pbc/notes/crypto/prng.html#:~:text=In%20cryptography%2C%20PRNG's%20are%20used,0%20%5D%20%3D%201%20%E2%88%92%20p%20.

Tick. (2012, February 9). *Псевдослучайно vs. По-настоящему Случайно*. Хабр. Retrieved January 5, 2023, from https://habr.com/ru/post/137864/

True random number generator article: Security ip. Synopsys. (n.d.). Retrieved January 5, 2023, from https://www.synopsys.com/designware-ip/technical-bulletin/true-random-number-generator-security-2019q3.html

What are the most popular random number generator algorithms used in video games (Mersenne Twister, well, ...)? Quora. (n.d.). Retrieved January 5, 2023, from https://www.quora.com/What-are-the-most-popular-random-number-generator-algorithms-used-in-video-games-Mersenne-Twister-WELL

Wong, D. (n.d.). *How does the Mersenne's twister work? posted February 2016*. How does the Mersenne's Twister work? Retrieved January 5, 2023, from https://www.cryptologie.net/article/331/how-does-the-mersennes-twister-work/

Xoroshiro generators and the PRNG shootout. xoshiro/xoroshiro generators and the PRNG shootout. (n.d.). Retrieved January 5, 2023, from https://prng.di.unimi.it/

Blum, M., & Blum, M. (1986). How to generate cryptographically strong sequences of pseudorandom bits. SIAM Journal on Computing, 15(2), 364-383.

Niemi, J., & Kärkkäinen, T. (2010). Generation of true random numbers. In Handbook of Random Number Generation and Monte Carlo Methods (pp. 31-53). John Wiley & Sons.

NIST. (2012). Recommendation for random number generation using deterministic random bit generators. Technical report, NIST SP 800-90A.

(guest), C. E. (n.d.). *Generating cryptographic keys: Will your random number generators (prngs) do the job?* Cryptomathic. Retrieved February 12, 2023, from https://www.cryptomathic.com/news-events/blog/generating-cryptographic-keys-with-random-number-generators-prng

Blum, L., Blum, M., & Shub, M. (1982, January 1). *[PDF] comparison of two pseudo-random number generators: Semantic scholar*. [PDF] Comparison of Two Pseudo-Random Number Generators | Semantic Scholar. Retrieved February 12, 2023, from https://www.semanticscholar.org/paper/Comparison-of-Two-Pseudo-Random-Number-Generators-Blum-Blum/b83cb17e6668f17f27c1dbe793007a7c34bbc572

Brent, R. P. (n.d.). *Fast and Reliable Random Number Generators for Scientific Computing*. Retrieved February 12, 2023, from https://maths-people.anu.edu.au/~brent/pd/rpb217.pdf

Buchanan, W. J. (n.d.). Blum Blum Shub. Retrieved February 12, 2023, from https://asecuritysite.com/encryption/blum

Cobb, M. (2023, January 9). *What is triple DES and why is it being disallowed?: TechTarget*. Security. Retrieved February 12, 2023, from https://www.techtarget.com/searchsecurity/tip/Expert-advice-Encryption-101-Triple-DES-explained#:~:text=Triple%20DES%20encryption%20process,a%20last%20time%20with%20K3

Corob-Msft. (n.d.). *Left shift and right shift operators ('<<' and '>>')*. Left shift and right shift operators ('>') | Microsoft Learn. Retrieved February 12, 2023, from https://learn.microsoft.com/en-us/cpp/cpp/left-shift-and-right-shift-operators-input-and-output?view=msvc-170

Jacak, M. M., Jóźwiak, P., Niemczuk, J., & Jacak, J. E. (2021, August 9). *Quantum generators of random numbers*. Nature News. Retrieved February 12, 2023, from https://www.nature.com/articles/s41598-021-95388-7

Jean-Michel Frouin on 25 February 2021. (2021, May 17). *True random number generation choosing the right TRNG*. Accelize. Retrieved February 12, 2023, from https://www.accelize.com/true-random-number-generation/

Ma, X., Yuan, X., Cao, Z., Qi, B., & Zhang, Z. (2016, June 28). *Quantum random number generation*. Nature News. Retrieved February 12, 2023, from https://www.nature.com/articles/npjqi201621

Mersenne Twister Home Page. Mersenne Twister: A random number generator (since 1997/10). (n.d.). Retrieved February 12, 2023, from http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/emt.html

Random number generators. Federal Office for Information Security. (2022, September 9). Retrieved February 12, 2023, from https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kryptografie/Zufallszahlengenerator/zufallszahlengenerator_node.html

Shipaaa. (2020, December 23). *Blum-Blum-Shub Generator и его применение*. Хабр. Retrieved February 12, 2023, from https://habr.com/ru/post/534698/

Stanford Seminar - PCG: A family of better random number generators. YouTube. (2015, February 19). Retrieved February 12, 2023, from https://youtu.be/45Oet5qjlms

Stubbs, R. (n.d.). *Classification of cryptographic keys*. Cryptomathic. Retrieved February 12, 2023, from https://www.cryptomathic.com/news-events/blog/classification-of-cryptographic-keys-functions-and-properties

Two sh: A 128-bit block cipher - schneier. (n.d.). Retrieved February 12, 2023, from https://www.schneier.com/wp-content/uploads/2016/02/paper-twofish-paper.pdf

Wong, D. (n.d.). *How does the Mersenne's twister work? posted February 2016*. How does the Mersenne's Twister work? Retrieved February 12, 2023, from https://www.cryptologie.net/article/331/how-does-the-mersennes-twister-work/

Квадратичный вычет. Сайт Вологодской областной универсальной научной библиотеки - ВОУНБ. (n.d.). Retrieved February 12, 2023, from https://www.booksite.ru/fulltext/1/001/008/060/238.htm