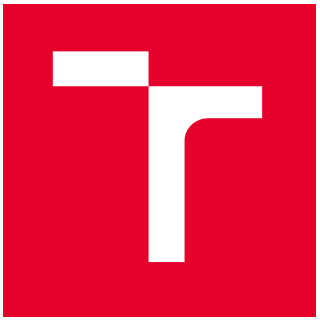


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

HODNOCENÍ GENERÁTORŮ NÁHODNÝCH ČÍSEL

RANDOM NUMBER GENERATORS EVALUATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Martin Svoboda**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Václav Zeman, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Martin Svoboda ***ID:*** 231283

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Hodnocení generátorů náhodných čísel

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je, pomocí statistického testování, porovnat kvalitu dat produkovaných různými typy náhodných a pseudonáhodných generátorů. V práci budou stručně popsány způsoby generování náhodných dat v běžné praxi a užívané způsoby testování kvality náhodnosti. V praktické části práce bude vytvořen program, který testování generátorů umožní.

DOPORUČENÁ LITERATURA:

[1] Turan M. S. aj.: Recommendation for the entropy sources used for random bit generation. NIST SP 800-90B. Gaithersburg: National Institute of Standards and Technology, 2018.

[2] Andrew Rukhin et al. Statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST special publication. [cit. 2017-12-17]. 2010.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Tématem práce jsou náhodná čísla a problematika jejich testování. V práci jsou uvedeny způsoby generování náhodných čísel a řada statistických testů náhodnosti. Součástí práce je popis vytvořené aplikace sloužící k testování náhodných čísel, která obsahuje sadu 23 statistických testů náhodnosti a 5 integrovaných generátorů. V poslední kapitole jsou pomocí aplikace otestovány různé druhy generátorů náhodných čísel a klíče vzniklé kvantovou distribucí klíčů.

KLÍČOVÁ SLOVA

Generátor, NIST, PRNG, Python, RNG, TRNG, entropie, náhodné číslo, sada testů, statistický test

ABSTRACT

The topic of the thesis is random numbers and the problem of their testing. The thesis presents methods of generating random numbers and many statistical tests of randomness. Part of the work is a description of an application created for testing random numbers, which contains a set of 23 statistical tests of randomness and 5 integrated generators. In the last chapter, various types of generators and keys created by quantum key distribution are tested using the application.

KEYWORDS

Generator, NIST, PRNG, Python, RNG, TRNG, entropy, random number, test suite, statistical test

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

SVOBODA, Martin. *Hodnocení generátorů náhodných čísel*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 79 s. Bakalářská práce. Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Martin Svoboda
VUT ID autora:	231283
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Hodnocení generátorů náhodných čísel

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc. Ing. Václavu Zemanovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále také rodině za podporu při studiu.

Obsah

Úvod

Tématem práce je hodnocení generátorů náhodných čísel. Náhodná čísla mají široké uplatnění a obzvláště pak v oblasti kryptografie jsou na jejich náhodnost kladeny velké nároky. Případná nenáhodnost těchto čísel vede k možnosti prolomení šifrované komunikace, která na náhodných číslech stojí, a dalším bezpečnostním rizikům. Proto je nutné generátory náhodných čísel před použitím řádně otestovat.

Cílem práce je stručně popsat generátory náhodných čísel, popsat statistické testy sloužící k ověření jejich kvality a následná implementace testů do vlastní aplikace pro hodnocení generátorů. Aplikace bude následně použita k testování dat produkovaných různými druhy generátorů.

V první kapitole práce jsou vysvětleny základní pojmy související s tématem náhodných čísel a jejich generování. Je zde uvedeno také využití těchto čísel v praxi.

Druhá kapitola se zabývá generátory pravých, pseudonáhodných a kryptograficky bezpečných pseudonáhodných čísel. Jsou zde popsány generátory s historickým významem i v praxi používané generátory.

Třetí kapitola popisuje testování statistických hypotéz a základní druhy rozdělení. Slouží jako teoretický základ ke kapitole následující.

Čtvrtá kapitola popisuje různé druhy statistických testů náhodných čísel – testy dobré shody, Poker test, Grafický test, Autokorelační test aj. U každého testu je popsán i postup implementace ve vlastní aplikaci. Také jsou zde zmíněny existující sady testů.

Pátá kapitola se věnuje sadě statistických testů pro generátory náhodných a pseudonáhodných čísel vydanou Národním institutem standardů a technologie (NIST), která slouží jako standard pro testování náhodných čísel. V kapitole je popsáno všech patnáct testů obsažených v této publikaci [1].

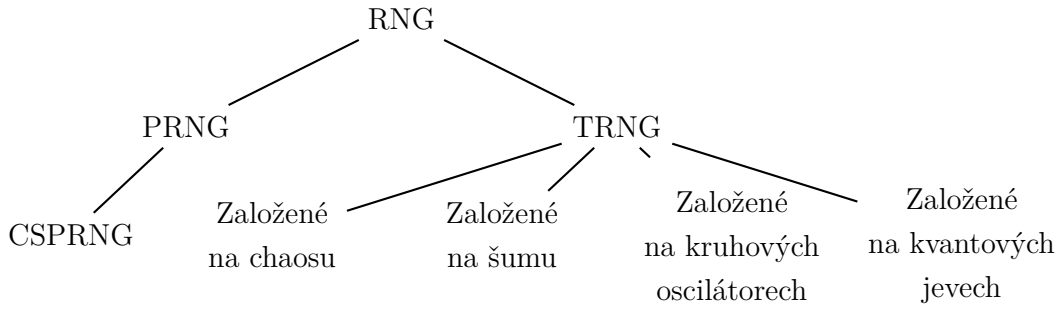
V šesté kapitole se nachází popis vytvořené aplikace sloužící k testování generátorů náhodných čísel. Kapitola se odkazuje na popis generátorů a statistických testů z kapitol předchozích.

Sedmá kapitola se věnuje hodnocení generátorů náhodných čísel s použitím vytvořené aplikace.

1 Náhodná čísla

1.1 Základní pojmy

- **Náhodné číslo** je definováno jako hodnota výběru z množiny čísel, kde výběr každého čísla má stejnou pravděpodobnost – jeho hodnota je nepředvídatelná. Jde tedy o realizaci náhodné veličiny s rovnoměrným rozdělením. Náhodný bit je případem náhodného čísla s omezenou množinou výběru pouze na hodnoty „0“ a „1“, které mají pravděpodobnost výběru přesně 1/2. V práci je používáno označení „náhodné číslo“, které zastřešuje i pojem „náhodný bit“.
- **Náhodná sekvence** je posloupnost náhodných čísel, kde hodnota dalšího v pořadí nemůže být odvozena od těch předchozích, je tedy nezávislá a postrádá vzor.
- **Generátor náhodných čísel** (RNG, generátor náhodných bitů – RBG) je proces generující náhodné sekvence čísel – posloupnosti bitů s hodnotou „0“ a „1“ jakožto nezávislých jevů se stejnou pravděpodobností. Rozlišujeme dva přístupy ke konstrukci generátorů náhodných čísel. Jejich rozdělení je zobrazeno na obrázku 1.1 (převzat ze zdroje [2]):



Obr. 1.1: Rozdělení generátorů náhodných čísel.

- **Generátor pravých náhodných čísel** (TRNG, označován také jako generátor skutečných náhodných čísel, generátor opravdu náhodných čísel apod.) je typem konstrukce generátorů, který generuje na základě náhodnosti získané z fyzikálních jevů. Takovéto generátory jsou nedeterministické (mají vždy přístup ke zdroji entropie a produkují bitové posloupnosti s maximální entropií), jsou neperiodické a zpravidla méně efektivní.
- **Generátor pseudonáhodných čísel** (PRNG) generuje čísla pomocí algoritmů, náhodnost je tedy pouze zdánlivá. Tyto generátory jsou deterministické (po vložení stejného vstupu produkují stejný výstup), výpočetně efektivnější než TRNG a jsou periodické [3].

- **Kryptograficky bezpečný generátor pseudonáhodných čísel** (CSPRNG, kryptografický generátor náhodných bitů) je druhem PRNG vhodným k využití v kryptografii. Takovýto generátor musí splňovat kromě základních požadavků na náhodnost také test dalšího bitu (*next-bit test*). Ten se zakládá na tom, že útočník znající prvních k bitů sekvence, není schopný v polynomiálním čase s přesností větší než nepatrně odlišující se od 1/2 předpovědět následující $(k+1)$ bit [4]. Dalším požadavkem je odolnost generátoru vůči kryptoanalýze – útočník by neměl být schopný předpovědět následující bity ani se znalostí části vnitřního stavu generátoru [5].
- **Entropie** (Shannonova entropie) je veličina udávající střední hodnotu informace na jeden vygenerovaný symbol dat. Čím větší je entropie, tím menší je znalost výsledku experimentu pro pozorovatele. Entropie H diskrétní náhodné proměnné X s pravděpodobnostní funkcí $\mathrm{P}(X)$ má tento vzorec:

$$H(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i), \tag{1.1}$$

  kde $b = 2$ pro výpočet entropie v bitech. Dalším způsobem vyjádření entropie je **min-entropie**, která vyjadřuje míru náhodnosti vzhledem k nejčastější vyskytující se hodnotě. Pro množinu hodnot $M = \{x_1, x_2, \ldots, x_k\}$ s pravděpodobností $P(X = x_i) = p_i$ je definována následovně:

$$H = \min_{1 \le i \le k} (-\log_2 p_i) = -\log_2 \max_{1 \le i \le k} p_i. \tag{1.2}$$

  Pravděpodobnost pozorování konkrétního výskytu hodnoty X s *min-entropií* H je nejvýše 2^{-H} [6].
- **Random seed** (náhodné semínko) je náhodné číslo nebo pole, které slouží jako inicializační hodnota PRNG.

1.2 Využití náhodných čísel v praxi

Čísla vybraná náhodou nacházejí využití v mnoha odvětvích. Například:

- *Simulace.* Pří počítačových simulacích je potřeba náhodných čísel pro realistické chování modelu. Jde například o simulace kvantových jevů, počasí, vývoj populace aj.
- *Simulace Monte Carlo.* Jde o třídu algoritmů, která využívá pseudonáhodných čísel pro simulaci systémů. Využití těchto simulací je možné například pro výpočty určitých a násobných integrálů, řešení systémů lineárních rovnic, hledání kořenů rovnic a výpočtů hodnot funkce [7].
- *Počítačové programování.* Náhodná čísla jsou používána například při testování a optimalizaci aplikací.

- *Umění.* Prvek náhodnosti ovlivní algoritmické generování umění tak, že výsledkem je unikátní a nepredikovatelný výstup imitující kreativitu.
- *Rozhodování.* Čísla vnášejí prvek náhodnosti do rozhodování v oblastech jako řízení rizik, vědeckých experimentů s vlivem náhodnosti aj.
- *Kryptografie.* Náhodná čísla jsou využívána například jako klíče relace, efemérní klíče algoritmů DSA, ECDSA, DH, jako parametry podpisů nebo u protokolů s nulovou znalostí. Tyto čísla by měly zůstat nepředvídatelné i za předpokladu, že útočník zná množství jiných vygenerovaných čísel nebo dokonce část počátečního *seedu* [8].

2 Generátory náhodných čísel

V této kapitole jsou představeny typické a v praxi používané generátory náhodných čísel. Kvalitní generátor by měl splňovat co nejvíce vlastností z následujícího výčtu [9]:

- *Náhodný vzor.* Měl by projít statistickými testy náhodnosti.
- *Dlouhá perioda.* Měl by generovat co nejdelší sekvenci před tím, než se začne opakovat.
- *Efektivita.* Měl by být rychlý a zabírat co nejméně paměti.
- *Opakovatelnost.* Měl by produkovat stejnou sekvenci čísel při zopakovaní počátečních podmínek.
- *Přenositelnost.* Měl by být spustitelný na různých počítačích a generovat na nich stejné sekvence.

2.1 Generátory pravých náhodných čísel

Tento typ zařízení generuje čísla přímo z fyzických procesů, které mohou být součástí počítače nebo připojeny externě.

Jako hardwarové zdroje náhodnosti lze využít například čas mezi emisí částic při radioaktivním rozpadu, tepelný šum z polovodičové diody či rezistoru, nebo kvantových jevů světla.

Generátory se vstupem založeným na softwaru mohou náhodnost získat ze systémových hodin, času mezi pohyby myší a stisknutím kláves, z vyrovnávací paměti, uživatelského vstupu nebo proměnných operačního systému (statistiky načítání, síťového připojení) [10].

2.1.1 Intel RNG

Na procesorech vyrobených společností Intel s architekturou x86-64 a IA-32 jsou implementovány funkce RDRAND pro generování kvalitních klíčů pro kryptografické protokoly a funkce RDSEED pro generování vstupních semínek softwarových generátorů pseudonáhodných čísel.

Generátor získává náhodnost z tepelného šumu křemíku, který transformuje na proud bitů [11].

2.1.2 Quantis TRNG

Quantis TRNG je generátor využívající kvantové vlastnosti světla. Fotony jsou posílány postupně na polopropustné zrcadlo a následně je detekováno jejich chování,

což může být buď průchod, nebo odražení. Tyto dva exkluzivní stavy jsou asociovány s bitovými hodnotami – „0“ a „1“.

Tento generátor je dostupný k zakoupení na USB disku a PCI. Náhodnost této metody potvrdilo i množství testů, mimo jiné testovací sada NIST SP 800-22 (viz kapitola 5) [12].

2.2 Generátory pseudonáhodných čísel

PRNG jsou deterministické algoritmy, které ze vstupní sekvence k (tzv. seedu) generují pseudonáhodnou sekvenci l, která se zdá být náhodná a zároveň platí, že délka $l \gg k$ [10].

2.2.1 Lineární kongruentní generátor

Lineární kongruentní generátor (LCG) patří mezi typické zástupce PRNG. Produkuje sekvenci čísel $x_1, x_1, \ldots, x_m$ podle vztahu:

$$X_{n+1} = (aX_n + c) \bmod m, \quad n \geq 1, \tag{2.1}$$

kde násobitel a a operátor inkrementace c jsou celá čísla, m je modul a X_0 počáteční *seed*. Vygenerovaná sekvence se periodicky opakuje, je odhadnutelná, a proto nevhodná pro využití v kryptografii [13].

2.2.2 Posuvný registr s lineární zpětnou vazbou

Generátory založené na posuvném registru s lineární zpětnou vazbou (*Linear Feedback Shift Register* – LFSR) jsou skupinou konstrukčně jednoduchých generátorů využívajících bitové operace. Náhodné číslo se získává opakovaným aplikováním *exkluzivní disjunkce* (XOR) mezi sekvencí a její kopií posunutou o jeden bit. Tento přístup ke konstrukci generátoru je vhodný k implementaci na zařízeních s omezeným výpočetním výkonem, protože bitový posun vpravo a operace XOR jsou jediné používané operace.

Prvky generátoru jsou registr, charakteristický polynom a vstupní stav registru. Velikost registru v bitech n určuje maximální periodu generátoru jako $2^n - 1$ (jde o všechny možné kombinace kromě nulového stavu). Vstupní sekvence je náhodný počáteční stav bez stavu obsahující jen nuly, protože za tohoto stavu by registr zůstal konstantní. LFSR je definován svým charakteristickým mnohočlenem, který musí být pro získání maximální periody primitivním mnohočlenem [10].

Generátor je nevhodný pro použití v kryptografii, protože útočník může z vygenerované části sekvence dopočítat atributy generátoru a zjistit následující vygenerované hodnoty. Proto bývá obvykle požíván jako součást komplexnějších algoritmů.

Sám obstojí tam, kde nejsou kladeny nároky na bezpečnost a na zařízeních nedisponujících výpočetním výkonem.

2.2.3 Xorshift

Jde o třídu výpočetních generátorů patřící do skupiny LFSR. Autorem generátoru je George Marsaglia, který je také znám pro vytvoření sady testů Diehard (viz kapitola 4.2.2).

Xorshift byl jedním z nejrychlejších generátorů. Na běžném počítači v roce vydání (2003) dokázal generovat až 220 miliónů náhodných čísel za sekundu. Periodou je 2^k , kde $k = 64, 96, 192 \ldots$ [14] Generátor úspěšně prošel jen některými soubory testů a dále se nedoporučuje pro svou krátkou periodu a nespolehlivost [15].

2.2.4 Merssene Twister

Merssene Twister je generátor s periodu $2^{19937-1}$ a je základním generátorem pro programovací jazyk Python, C, výpočetní systém Maple a mnoho dalších [16]. Přes jeho velké využití se nedá považovat za kryptograficky bezpečný.

Algoritmus generování je založen na této lineární rekurentní rovnici:

$$x_{k+n} := x_{k+m} \oplus \left(x_k^u \mid x_{k+1}^l\right) A, \quad (k = 0, 1, \ldots). \tag{2.2}$$

Konstanta n určuje míru rekurzivity, konstanta m v rozsahu $1 \leq m < n$ zajišťuje posun, x_{k+1}^l je spodních r bitů ve slově x_{k+1}, x_k^u horních $w - r$ bitů slova x_k, A je konstantní matice o rozměrech $w \cdot w$, $\oplus$ bitová operace XOR a $\mid$ spojovací operace zřetězení. Detailní popis algoritmu, který je nad rámec této práce, je uveden v autorské publikaci [17].

2.2.5 Squares

Na čítači založený pseudonáhodný generátor *„Squares: A Fast Counter-Based RNG“* publikován roku 2022 implementuje „metodu prostředku čtverce“ Johna von Neumanna. Tato metoda z roku 1946 má následující postup: vybere se *seed* o $2n$ číslicích, je umocněn na druhou a případně doplněn nulami zleva na $4n$ číslic. Z takto vygenerovaného čísla je vzato prostředních $2n$ číslic, se kterou se pokračuje v další iteraci [13].

Implementace Squares generuje 32-bitový výstup po 4 iteracích a 64-bitový výstup po 5 iteracích. Výhodou je jeho rychlost (autoři uvádí miliardu vygenerovaných čísel za $1,35$ sekundy s procesorem Intel Core i7-9700 3.0 GHz) a bezpečnost (generátor obstál sadu testů náhodnosti BigCrush obsaženou v baterii testů TestU01 (4.2.3) [18].

2.3 Kryptograficky bezpečné generátory náhodných čísel

Jde o PRNG generující čísla s vlastnostmi vhodnými k použití v kryptografii. Kombinují výhody obou přístupů generování – kvalitní zdroj entropie typický pro TRNG a výpočetní rychlost a softwarovou implementaci jako PRNG. Větší nepredikovatelnost výstupu typicky souvisí s nevýhodou větší časové a paměťové náročnosti generování.

2.3.1 Blum Blum Shub

Jde o jednoduchý generátor publikován roku 1986. Je pojmenován podle autorů, kterými jsou Lenore Blum, Manuel Blum a Michael Shub. Generátor *Blum Blum Shub* (BBS) stojí na rekurzivní formuli:

$$x_i = (x_{i-1}^2) \bmod M, \tag{2.3}$$

kde M je součin dvou prvočísel p a q pro které platí, že jsou kongruentní 3 modulo 4 a jsou stejné délky. Inicializační hodnota x_0 splňuje $0 < x_0 < M$ a $\gcd(x_0, M) = 1$. Výhodou generátoru je možnost přímo vypočítat x_i pomocí Eulerovy věty jako:

$$x_i = \left(x_0^{2^i \bmod \lambda(M)}\right) \bmod M, \tag{2.4}$$

kde λ je Carmichaelova funkce, v tomto případě jako $\lambda M = \lambda(p \cdot q) = \mathrm{lcm}(p-1, q-1)$. bezpečnost generátoru je založena na problému nalezení kvadratických zbytků [19].

2.3.2 Micali-Schnorr

Tento kryptograficky bezpečný generátor náhodných čísel je založen na RSA šifře – stojí tedy na problému faktorizace velkých čísel. Postup generování je následující [10]:

1. Příprava: Generování tajných prvočísel p a q, výpočet $n = pq$ a $\phi = (p-1)(q-1)$, $N = \lfloor \log n \rfloor + 1$ s bitovou délkou n. Zvolení celého čísla e, z rozmezí $1 < e < \phi$ tak, že $\gcd(e, \phi) = 1$ a $80e \leq N$. Nechť $k = \lfloor N(1-2/e) \rfloor$ a $r = N-k$.
2. Zvolení náhodné sekvence x_0 (vstupního seedu) o bitové délce r.
3. Generování pseudonáhodné sekvence o délce $k \cdot l$. V rozsahu $i = (1, \ldots, l)$ je proveden výpočet:
   (a) $y_i \leftarrow x_{i-1}^e \mod n$.
   (b) $x_i \leftarrow \mathrm{msb}(r)$, kde msb označuje nejvýznamnější bit.
   (c) $z_i \leftarrow \mathrm{lsb}(k)$, kde lsb označuje nejméně významný bit.
4. Výstupem je posloupnost $z_1, z_2, \ldots, z_l$.

2.3.3 ISAAC

Generátor ISAAC je považován za kryptograficky bezpečný, přestože není jeho bezpečnost prokázána jiným způsobem, než obstáním statistických testů. Algoritmus je podobný šifře RC4. Vnitřním stavem je pole 256 čtyřoktetových celých čísel, která jsou čtena po jednom a po vyprázdnění pole jsou přepočítána záměnou i-tého prvku za ($i \oplus 128$ prvek) [20].

3 Testování statistických hypotéz

Pro potřeby následujících kapitol, věnujících se statistickému testování generátorů náhodných čísel, je zde popsána metodika statistického testování se zaměřením právě na testování náhodnosti. Kapitola vychází ze zdroje [21].

Statistická hypotéza je jakékoliv tvrzení o rozdělení náhodné veličiny. Základem správně provedeného testu je dobře formulovaná dvojice hypotéz – nulová hypotéza H_0 a alternativní hypotéza H_1, která je jejím logickým opakem.

Pro náhodný výběr $X = (X_1, \ldots, X_N)$ z určitého rozdělení R_θ s parametrickou funkcí $h(\theta)$ a reálnou konstantou k mějme nulovou hypotézu ve tvaru:

$$H_0\text{: } h(\theta) = k. \tag{3.1}$$

V tomto případě můžeme rozlišit mezi dvěma druhy alternativních hypotéz:

1. Oboustranná alternativní hypotéza ve tvaru H_1: $h(\theta) \neq k$.
2. Jednostranná alternativní hypotéza H_1: $h(\theta) > k$ (pravostranná) a $H_1 : h(\theta) < k$ (levostranná).

Výsledkem testu je to, zdali hypotézu H_0 přijímáme, nebo odmítáme. K učinění tohoto rozhodnutí slouží *testová statistika (testovací kritérium)*, kterou označíme jako T. Její obor je rozdělen na dvě disjunktní množiny – obor zamítnutí (kritický obor) a obor přijetí. Tyto množiny jsou ohraničeny *kritickými hodnotami* určenými z teoretického rozložení statistiky.

Při rozhodnutí o výsledku statistického testu se můžeme dopustit dvou typů chyb:

Tab. 3.1: Možné výsledky testu hypotéz.

Skutečnost	Rozhodnutí	
	zamítáme H_0	nezamítáme H_0
H_0 platí	chyba 1. druhu	správné rozhodnutí
H_0 neplatí	správné rozhodnutí	chyba 2. druhu

Chyba 1. druhu nastává při zamítnutí ve skutečnosti pravdivé hypotézy a je označována jako hladina významnosti α. Hladina významnosti pro kryptografii je typicky $0,01$ [1].

Chyba 2. druhu nastává při nezamítnutí nepravdivé hypotézy a je označována jako β. Doplňkem k β je síla testu, která udává pravděpodobnost zamítnutí nepravdivé hypotézy.

Při testování požadujeme aby hodnota obou druhů chyb byla co nejmenší. Za konstantního rozsahu n výběrového souboru platí, že snížením chyby jednoho druhu se zvětší chyba druhu druhého.

Obvykle jako součást výstupů matematických programů a ve většině dále uvedených testů se objevuje pojem *p-hodnota*. Jde o nejmenší hladinu významnosti při které bychom hypotézu H_0 zamítli. Jestliže je *p-hodnota* větší než stanovená hladina významnosti, hypotéza H_0 se přijímá. V opačném případě se H_0 zamítá.

Pro náhodný výběr $X = (X_1, \ldots, X_N)$ z určitého rozdělení R_θ s testovou statistikou $X = (T_1, \ldots, T_N)$ a její hodnotou t_0 je *p-hodnota* určena podle typu alternativní hypotézy H_1 následujícími způsoby:

1. Pro oboustrannou alternativní hypotézu je

$$p\text{-}hodnota = 2\min\{\Pr(T \geq t|H_0), \Pr(T \leq t|H_0)\}, \tag{3.2}$$

   kde Pr označuje pravděpodobnost daného jevu.
2. Pro jednostrannou alternativní hypotézu rozlišujeme druh pravostranný:

$$p\text{-}hodnota = \Pr(T \geq t|H_0), \tag{3.3}$$

   a druh levostranný:

$$p\text{-}hodnota = \Pr(T \leq t|H_0)\}, \tag{3.4}$$

Výsledek testu tedy závisí na zvolení hladiny významnosti α. Po výpočtu *p-hodnoty* můžeme tyto hodnoty porovnat a rozhodnout o zamítnutí, či potvrzení nulové hypotézy. Při testu statistické hypotézy založeném na *p-hodnotě* je postup tedy následující:

1. Formulace nulové a alternativní hypotézy.
2. Volba hladiny významnosti α.
3. Výpočet hodnoty testové statistiky.
4. Výpočet *p-hodnoty*.
5. Učinění závěru testu: pokud je *p-hodnota* $\leq \alpha$, tak H_0 zamítáme. V opačném případě H_0 nezamítáme.

3.1 Vybrané druhy rozdělení

Při testování jsou využívány rozdělení spojitých náhodných veličin. Jsou zde popsány normální, χ^2 a Poissonovo rozdělení, které jsou využívány u následně popsaných testů náhodnosti.

3.1.1 Normální rozdělení

Normální (Gaussovo) rozdělení $N(\mu, \sigma^2)$ náhodné veličiny X, kde $\mu \in \mathbb{R}$ a $\sigma > 0$ má hustotu pravděpodobnosti ve tvaru:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbb{R}. \tag{3.5}$$

Normální rozdělení s $X \in N(0, 1)$ je označeno za normované normální rozdělení. Toto rozdělení má tedy hustotu pravděpodobnosti ve tvaru:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \quad x \in \mathbb{R}. \tag{3.6}$$

Distribuční funkce normálního rozdělení je:

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \mathrm{d}t, \quad x \in \mathbb{R}. \tag{3.7}$$

3.1.2 χ^2 rozdělení

Rozdělení χ^2 („chí kvadrát“, Pearsonovo rozdělení) je rozdělení náhodné veličiny X, která je součtem n vzájemně nezávislých druhých mocnin veličin U s normovaným rozdělením $N(0, 1)$. Rozdělení $\chi^2(n)$ o n stupních volnosti má hustotu pravděpodobnosti:

$$f(x) = \begin{cases} 0 & x \leq 0 \\ \frac{x^{\frac{n}{2}-1}}{2^{\frac{n}{2}}\Gamma(\frac{n}{2})} \mathrm{e}^{-\frac{x}{2}} & x > 0, \end{cases} \tag{3.8}$$

kde $\Gamma(n)$ je gama funkce (viz vzorec 5.3). Střední hodnota rozdělení $\chi^2(n)$ je $E(X) = n$ a rozptyl $\sigma^2(X) = 2n$. Distribuční funkci můžeme vyjádřit pomocí neúplné gama funkce γ (viz vzorec 5.4) jako:

$$F(x) = \frac{\gamma(\frac{n}{2}, \frac{x}{2})}{\Gamma(\frac{n}{2})}. \tag{3.9}$$

3.1.3 Poissonovo rozdělení

Poissonovo rozdělení popisuje náhodnou veličinu vyjadřující počet výskytů nezávislých jevů v určitém intervalu. V praxi se s rozložením setkáme například při pozorování částic nebo počtu událostí s malou pravděpodobností.

Náhodná veličina X má Poissonovo rozdělení s parametrem $\lambda > 0$:

$$P(X = x) = \frac{\lambda^x}{x!}\mathrm{e}^{-\lambda}, \quad x = 0, 1, \ldots, \tag{3.10}$$

kde e je Eulerovo číslo, x počet výskytů události a λ střední hodnota výskytu jevů za časovou jednotku. Střední hodnota náhodné veličiny je $E(X) = \lambda$ a rozptyl $D(X) = \lambda$. Distribuční funkce Poissonova rozdělení je tvaru:

$$F(x) = \sum_{x_i \leq x} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}. \tag{3.11}$$

4 Testy generátorů náhodných čísel

Tato kapitola se věnuje testování kvality vygenerovaných sekvencí náhodných čísel. Testovat náhodnost čísel můžeme pomocí jednotlivých statistických testů nebo pomocí již připravených baterií testů. Rozlišujeme dva typy testů:

- **Teoretické testy**, které o kvalitě generátoru rozhodnou na základě principů fungování bez potřeby generovat náhodná data.
- **Empirické testy**, které kvalitu generátoru posuzují skrze aplikaci testů na vygenerovanou sekvenci. Na tyto typy testů je tato práce zaměřena.

Většina testů porovnává určitou vlastnost vygenerované sekvence s teoretickou sekvencí s náhodnými vlastnostmi.

4.1 Statistické testy

4.1.1 Test χ^2 dobré shody

Tento statistický test je používán ke srovnání skutečných (empirických) hodnot a očekávaných (teoretických) četností odpovídajících příslušnému očekávanému rozdělení pravděpodobnosti. Test se aplikuje po rozdělení čísel do k kategorií provedením n nezávislých pokusů. Testová statistika T je vypočítána podle vztahu:

$$T = \sum_{j=i}^{k} \frac{(n_j - np_j)^2}{np_j}, \tag{4.1}$$

kde n_j označuje počet hodnot intervalu j a p_j očekávanou pravděpodobnost výskytu čísel v tomto intervalu. Stupně volnosti testu v jsou určeny jako $k - 1$. Pro vyhodnocení testu je po výpočtu testového kritéria T a stupňů volnosti v vyhledána v tabulkách kritická hodnota. Jestliže T překročí kritickou hodnotu je nulová hypotéza zamítnuta [13].

Implementace χ^2 testu

Jde o základní statistický test vhodný k aplikaci na diskrétní soubory hodnot a je využíván celou řadou dalších testů. Pro použití v jazyce Python je k dispozici funkce `chisquare`, kterou je možné importovat jako:

```
from scipy.stats import chisquare
```

Funkce je součástí rozšířené open-source knihovny pro statistické výpočty `scipy` [22].

4.1.2 Kolmogorovův–Smirnovův test

Kolmogorovův–Smirnovův test (K–S test) ověřuje, zda se naměřené rozdělení náhodné veličiny liší od teoretického rozdělení. Nulová hypotéza H_0 říká, že výběr

o rozsahu n pochází ze základního souboru s očekávaným rozdělením. Testovací kritérium D je vypočítáno:

$$D = \max_{1 \leq i \leq n} \left\{ \max \left\{ |F(x_i) - \frac{i}{n}|, |F(x_i) - \frac{i-1}{n}| \right\} \right\}, \tag{4.2}$$

kde x_i jsou pozorované hodnoty uspořádané podle velikosti ($x_1 \leq x_2 \leq \cdots \leq x_n$). Hypotéza H_0 se zamítá, je-li $D \geq D_\alpha$. Kritické hodnoty D_α pro jednotlivé hladiny významnosti jsou uvedeny v tabulkách a pro $n \geq 30$ je lze aproximovat jako [23]:

$$D_n(\alpha) \approx \sqrt{\frac{1}{2n} \ln \frac{2}{\alpha}}. \tag{4.3}$$

Implementace Kolmogorovova–Smirnovova testu

Jedná se o další základní statistický test, který je vhodný k testování veličin se spojitým rozdělením pravděpodobnosti a malým počtem vzorků. K dispozici je funkce `kstest`, která je rovněž součástí knihovny `scipy` [22]:
`from scipy.stats import kstest`.

4.1.3 Poker test

Test se zabývá četnostmi výskytu kombinací číslic typických pro hru poker, podle které je pojmenován. Číslům jsou v tomto testu přiřazena označení a, b, c, d a e. Pravděpodobnosti výskytu kombinací po rozdělení na pětice jsou uvedeny v tabulce 4.1:

Tab. 4.1: Pravděpodobnosti kombinací při poker testu.

Kombinace	Název	Pravděpodobnost
abcde	žádná shoda	0,3024
aabcd	dvojice	0,5040
aabbc	dvě dvojice	0,1080
aaabc	trojice	0,0720
aaabb	dvojice, trojice	0,0090
aaaab	čtveřice	0,0045
aaaaa	pětice	0,0001

Po rozdělení sekvence do skupin po pěti je součet četností výskytu kombinací v každé pětici porovnán pomocí χ^2 testu ($k = 5$) s teoretickým očekávaným počtem.

Další možností je rozdělit testovanou sekvenci do čtveřic namísto pětic z důvodu toho, že většina generátorů produkuje čísla o velikosti 64 nebo 32 bitů. Autoři článku

[24] pak ukazují, že rozdělení do menších skupin (trojic, čtveřic) se výrazně pozitivně projeví na době výpočtu testu.

V publikaci [25] je popsán algoritmus testu a jeho zjednodušená verze, která dosahuje srovnatelných výsledků v časech výpočtů, ale značně zjednodušuje programátorům implementaci.

Implementace poker testu

Modifikovaná verze rozděluje pětice čísel do pěti kategorií podle počtu rozdílných hodnot, jak je uvedeno v tabulce 4.2:

Tab. 4.2: Kombinace při modifikovaném poker testu.

Kombinace	Možnosti v klasickém testu	Pravděpodobnost
5 rozdílných	všechny rozdílné	0,3024
4 rozdílné	jedna dvojice	0,504
3 rozdílné	dvě dvojice, nebo trojice	0,180
2 rozdílné	trojice a dvojice nebo čtveřice	0,135
1 rozdílná	pět stejných	0,0001

Tyto kombinace zjednodušují postup v tom, že není nutné rozlišovat o která čísla se jedná, ale stačí je mezi sebou porovnat.

Podle této logiky je v aplikaci implementována bitová verze – sekvence bitů je rozdělena do trojic, které jsou rozděleny do skupin podle počtu výskytu hodnoty „1“. Tyto počty pozorovaných hodnot výskytů „1“ ve skupinách jsou porovnány s očekávanými hodnotami (pravděpodobnosti očekávaných výskytů jsou uvedeny v tabulce 4.3) v opravdu náhodné sekvenci pomocí testu χ^2. Test se od klasického Poker testu tedy liší počtem čísel ve skupině a také bitovou úpravou – v řetězci bitů nemůžou být všechny hodnoty rozdílné, jako v klasické verzi.

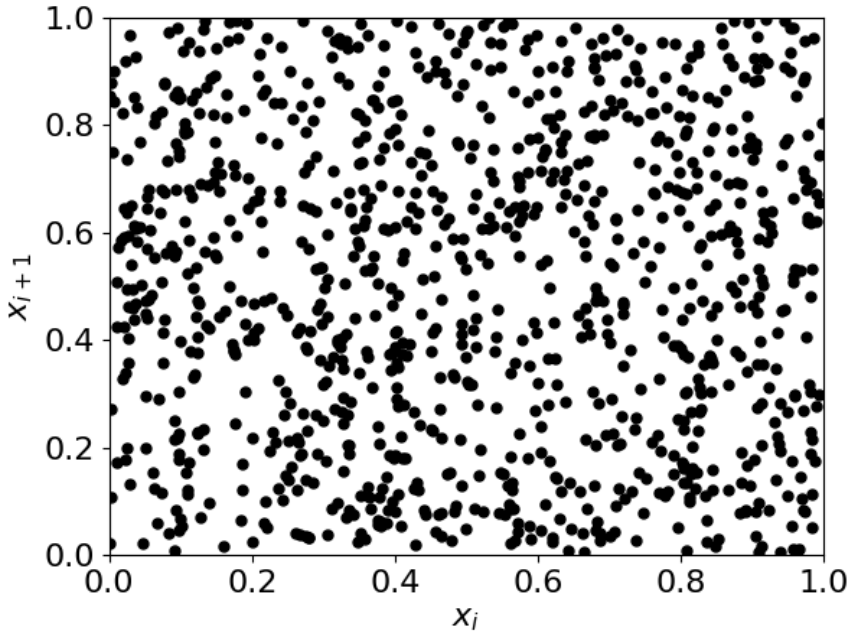
Tab. 4.3: Kombinace bitové verze poker testu v trojicích.

Kombinace	Pravděpodobnost
žádný bit „1“	0,125
jeden bit „1“	0,375
dva bity „1“	0,375
všechnu bity „1“	0,125

4.1.4 Grafický test

Podstatou grafických testů náhodnosti je vizualizovat rozložení náhodných čísel. Na obrazovce jsou zobrazeny body, které mají jako souřadnice dvě po sobě jdoucí čísla z vygenerované sekvence. Uživatel tedy sám vyhodnotí to, zdali se body objevují s pravidelností indikující nenáhodnost, nebo rovnoměrně jako očekáváme v náhodné sekvenci.

Graf 4.1 zobrazuje příklad kvalitního generátoru[1]. Graf 4.2 ukazuje příklad výsledku testu u generátoru, který generuje nenáhodná čísla[2]

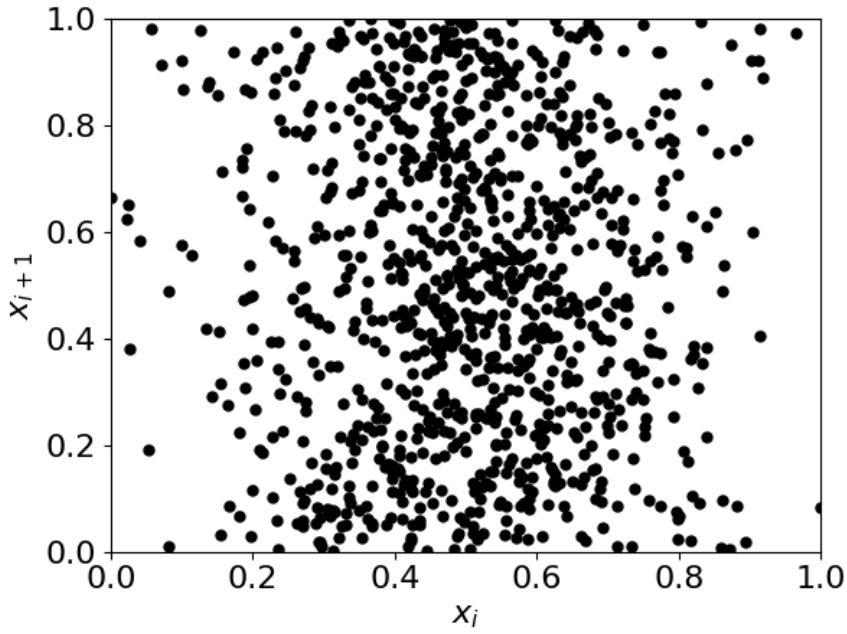


Obr. 4.1: Grafický test kvalitního generátoru.

Efektivita testu závisí i na rozlišení monitoru, protože monitor s lepšími vlastnostmi je schopný vykreslit více bodů a v případě překrytí bodů měnit odstín barvy. Tento test je i přes svou jednoduchost užitečný pro schopnost demonstrovat uživateli vzory nenáhodnosti a je schopný odhalit nenáhodnost i v sekvencích, které by jednoduchý K-S nebo χ^2 test označil za náhodné [26].

[1] Jedná se o generátor jazyka Python (modul `random`).

[2] Graf byl zkonstruován jako ukázka. Hodnoty na ose x jsou čísla vygenerovaná modulem `random` s normálním rozdělením (s parametry $\mu = 0,5$ a $\sigma = 0,1$) a hodnoty na ose y jsou náhodné hodnoty s rovnoměrným rozložením.



Obr. 4.2: Grafický test nekvalitního generátoru.

Implementace grafického testu

Při implementaci byla využita knihovna `matplotlib.pyplot`, která dokáže vhodně vykreslovat grafy a vizualizovat data. Postup testu je následující:

1. Převedení vložené sekvence na hodnoty z intervalu $(0, 1)$.
2. Střídavě rozdělit hodnoty na souřadnice bodů x a y.
3. Vykreslení souřadnicového systému a bodů.
4. Vyhodnocení testu.

Test se vymyká ostatním statistický testům, protože nevyužívá metody statistického testování, ale rozhodnutí o náhodnosti ponechává na uživateli.

Vyčíslení výsledku grafického testu může být uskutečněno pomocí Monte Carlo simulace π s následujícím postupem: body sekvence jsou zobrazeny v jednotkovém čtverci, jehož obsah $S = 1$. Vykreslení kružnice se středem v bodě $(0, 0)$ a poloměrem $r = 1$ rozdělí celkový počet bodů n na body ležící uvnitř kružnice n_{in} a ležící vně n_{out}. Obsah této kružnice $S_{\text{k}} = \pi/4$. Pro náhodně vygenerovanou sekvenci o dostatečně velkém n pak platí:

$$4 \cdot \frac{n_{\text{in}}}{n_{\text{in}} + n_{\text{out}}} \approx \pi. \tag{4.4}$$

Takto vypočítanou hodnotu π porovnáme se skutečnou hodnotou, ke které by se měla co nejvíce blížit.

4.1.5 Test autokorelace

Test slouží k odhalení závislosti vygenerovaných čísel v rozestupech m, tedy čísel x_i, $x_{i+m}, x_{i+2m}, \ldots, x_{i+(M+1)m}$, kde M je nejvyšší číslo pro které platí $i+(M+1)m \leq n$ a n je počet vygenerovaných čísel. Testované hypotézy testu:

$$H_0 : \rho_{im} = 0, \quad \text{čísla jsou nezávislá,} \tag{4.5}$$

$$H_1 : \rho_{im} \neq 0, \quad \text{čísla jsou závislá.} \tag{4.6}$$

Pro velké hodnoty M a za předpokladu nezávislosti testovaných prvků x je rozdělení pravděpodobnosti ρ_{im} přibližně normální. Testová statistika s normovaným normálním rozdělením $N(0,1)$ je:

$$z_0 = \frac{\hat{\rho_{im}}}{\sigma_{\hat{\rho_{im}}}}. \tag{4.7}$$

Vzorec pro výpočet $\hat{\rho_{im}}$ je:

$$\hat{\rho_{im}} = \frac{1}{M+1}\left[\sum_{k=0}^{M} x_{i+km}x_{i+(k+1)m}\right] - 0,25 \tag{4.8}$$

a směrodatné odchylky:

$$\sigma_{\hat{\rho_{im}}} = \frac{\sqrt{13M+7}}{12(M+1)}. \tag{4.9}$$

Jestliže platí:

$$-z_{\alpha/2} \leq z_0 \leq z_{\alpha/2}, \tag{4.10}$$

tak nezamítáme hypotézu H_0. Nevýhodou tohoto testu je vysoká šance chyby 1. druhu, tedy nalezení autokorelace a zamítnutí H_0 i v náhodné sekvenci [27].

Implementace autokorelačního testu

Test autokorelace je implementován v podobě testující posuny o 1–10 bitů. Pro každý posun je vypočítán průměr, rozptyl a autokorelační koeficient. Dále jsou pro jednotlivé posuny vypočítány *p-hodnoty*, které jsou porovnány s kritickou hodnotou $0,01$.

4.1.6 Test mezer

„*Gap test*“ vyšetřuje délky mezer mezi výskyty daného prvku sekvence x_i v určitém intervalu. Jsou určena čísla $\alpha, \beta \in \mathbb{R}$ pro které platí $0 \leq \alpha < \beta \leq 1$ a test hledá souvislé sekvence $x_i, x_{i+1}, \ldots, x_{i+r}$ takové, kde x_{i+r} leží mezi α a β, ale další vygenerovaná čísla x ne. Tyto podřetězce $r+1$ čísel reprezentují mezeru o velikosti r. Postup testu mezer je následující [13]:

1. Zjistíme počty mezer o délkách $0, 1, \ldots, t-1, t$ a $> t$.
2. Počet kategorií délek mezer je $k = t + 1$.
3. Výpočet pravděpodobností $p_r = p(1-p)^r$, kde $p = \beta - \alpha$.

Na takto vypočítané pravděpodobnosti p_r se dále aplikuje χ^2 test (4.1.1) nebo K-S test (4.1.2). *Test běhů nad průměrem* je zvláštním případem testu, kdy $(\alpha, \beta) = (0, \frac{1}{2})$ a *test běhů pod průměrem* je variantou s $(\alpha, \beta) = (\frac{1}{2}, 1)$.

Implementace testu mezer

Test je vykonán obdobně i na sekvenci bitů. Rozdílem je, že se nevybírají prvky α a β jako krajní hodnoty mezer, ale pozorují se mezery nulových bitů mezi hodnotami jedna. Postup je tedy následující:

1. Inicializace slovníku, do kterého budou přidávány jako klíče délky nalezených mezer a jako hodnoty počet takto nalezených mezer.
2. Průchod bitovou sekvencí:
   (a) Při hodnotě „0“ je proměnná uchovávající aktuální délku mezery inkrementována o jedna.
   (b) Je-li hodnota „1“ a zároveň délka aktuální mezera nenulová, tak je délka mezery přidána do slovníku, případně hodnota výskytů inkrementována o jedna, jestliže se ve slovníku mezera takovéto délky nachází.
3. Vytvoření listu očekávaných hodnot pro jednotlivé délky mezer.
4. Porovnání pozorovaných počtů mezer jednotlivých délek s počtem očekávaným pomocí testu χ^2.

4.1.7 Test bodů zvratu

Tento jednoduchý test zkoumá frekvence výskytu bodů zvratu. Jako bode zvratu je označen prvek sekvence x_i, jehož sousední prvky x_{i-1} a x_{i+1} jsou současně buď menší, nebo větší. V náhodné sekvenci, ze které jsou tyto tři sousední hodnoty vybrány, by měly být rovnoměrně zastoupené všechny možnosti kombinací těchto prvků ($3! = 6$ možností), ve kterých se body zvratu vyskytují s pravděpodobností 2/3.

Celkový počet bodů zvratu v posloupnosti je označen Y. Za předpokladu platnosti H_0 má testová statistika Y asymptoticky normální rozložení se střední hodnotou $E(Y) = \frac{2(n-2)}{3}$ a rozptylem $D(Y) = \frac{16n-29}{90}$. Standardizovaná statistika U má tvar:

$$U = \frac{Y}{\sqrt{\frac{16n-29}{90}}} \approx N(0,1). \tag{4.11}$$

Kritickým oborem W je:

$$W = (-\infty, -u_{1-\frac{\alpha}{2}}\rangle \cup \langle u_{1-\frac{\alpha}{2}}, \infty). \tag{4.12}$$

Jestliže $U \in W$, tak na hladině významnosti α hypotézu H_0 zamítáme [28].

Implementace testu zvratů

Program obsahuje test v následující podobě:

1. Převod bitové sekvence na sekvenci celých čísel.
2. Pro každé tři po sobě jdoucí prvky je zjištěno, zdali se jedná o bod zvratu. V případě že ano, tak je počet bodů zvratu inkrementován o jedna.
3. Výpočet střední hodnoty, rozptylu a testové statistiky.
4. Výpočet p-hodnoty a vyhodnocení testu.

4.1.8 Lempel–Ziv kompresní test

Kompresní Lempel–Ziv (LZ) test se zabývá počtem rozdílných vzorů v testované sekvenci. Test byl obsažen v původní sadě NIST z roku 2000 jako šestnáctý test, ale z aktuální verze byl vyřazen. Smyslem testu je zjistit, jak moc může být sekvence komprimována. V případě možnosti velké komprese je sekvence označena za nenáhodnou. Naopak náhodná sekvence se vyznačuje velkým množstvím rozdílných vzorů.

Ve vyřazení testu z baterie testů patrně hrálo roli to, že je navržen jen na sekvence o délce 10^6 bitů. Pro jinak dlouhé sekvence neuvádí postup výpočtu σ a μ, které jsou nutné pro výpočet testové statistiky. V těchto hodnotách byla dokonce nalezena chyba [29]. Potenciální chyby vedoucí k chybným výsledkům jsou nežádoucí z důvodů možného ohrožení šifrované komunikace. Baterie testů NIST byla totiž použita i k výběru finálního kandidáta pro šifrovací algoritmus AES, na který byl požadavek náhodnosti.

Implementace LZ testu

Test byl naprogramován podle popisu v publikaci NIST [30] s následujícím postupem:

1. Kontrola počátečních podmínek. Sekvence musí mít délku 10^6 bitů, delší sekvence je oříznuta.
2. Rozdělení testované sekvence na po sobě jdoucí, rozdílné a nepřekrývající se podsekvence, ze kterých je vytvořen list unikátních kombinací. Délka tohoto listu je označena jako W_{obs}.
3. Výpočet $p\text{-}hodnoty = \frac{1}{2} erfc\left(\frac{\mu - W_{obs}}{\sqrt{2\sigma^2}}\right)$, kdy $\mu = 69586,25$ a $\sigma = \sqrt{70,448718}$.
4. Vyhodnocení testu: je-li $p\text{-}hodnota \geq 0,01$, tak je sekvence považována za náhodnou. V opačném případě je posouzena jako nenáhodná.

4.1.9 Test Hammingovy váhy

Test porovnává hodnoty Hammingovy váhu napříč podsekvencemi a porovnává jejich počet výskytů s očekávanými hodnotami v náhodné sekvenci. Test je aplikován na řetězce o délce 32 bitů, ve kterých je možná Hammingova váha nula až 32. Počty výskytů u jednotlivých kategorií jsou rozděleny do kategorií:

- $K_1 = 1$ pro podsekvence s počtem jedniček menším než 8.
- $K_2 = 2$ pro podsekvence s počtem jedniček rovným 8.
- $K_3 = 3$ pro podsekvence s počtem jedniček rovným 9.
- ...
- $K_{18} = 18$ pro podsekvence s počtem jedniček rovným 24.
- $K_{19} = 19$ pro podsekvence s počtem jedniček větším než 24.

Dále jsou vypočítány pravděpodobnosti výskytu jednotlivých kategorií a očekávané počty výskytů v celkové sekvenci. Tyto očekávané hodnoty jsou porovnány s pozorovanými a vyhodnoceny pomocí χ^2 test ($k = 18$) [31].

Implementace testu Hammingovy váhy

V programu je implementována verze testující podsekvence o délce 8 bitů. U těchto kratších řetězců není nutné sdružovat méně pravděpodobné kategorie do jedné skupiny, protože na rozdíl od delších sekvencí nastanou s větší pravděpodobností. Průběh testu tedy je:

1. Rozdělení sekvence bitů do osmic.
2. Výpočet Hammingovy váhy pro každou podsekvenci.
3. Součet počtu výskytů v každé kategorii.
4. Výpočet očekávaných pravděpodobností výskytu a teoretických četností.
5. Porovnání empirických a teoretických hodnot testem χ^2 a vyhodnocení na základě vypočítané *p-hodnoty*.

4.1.10 Test dvojic bitů

Smyslem testu je obdobně jako u předchozích testů porovnat pozorované hodnoty výskytu určitého jevu v dané sekvenci s očekávanými hodnotami v náhodné sekvenci. V tomto případě se jedná o skupiny dvojic bitů (jejich kombinace mohou být 00, 01, 10, 11). V náhodné sekvenci je předpokládáno rovnoměrné rozložení – každá z těchto dvojic má stejnou pravděpodobnost výskytu [31].

Implementace testu dvojic bitů

Implementace testu je následující:

1. Rozdělení testované sekvence do sousedních dvojic.

2. Součet počtu výskytů každé kombinace.
3. Výpočet teoretických hodnot výskytů.
4. Srovnání testem χ^2, vyhodnocení na základě vypočítané *p-hodnoty*.

4.2 Baterie testů

Pro komplexnější otestování kvality generátorů náhodných čísel existují baterie testů. Jde o soubory více testů sloužící k důkladnému otestování generátoru a k posouzení jeho případné vhodnosti použití v kryptografii. V této podkapitole jsou zmíněny nejznámější z nich.

4.2.1 ENT

Tento soubor testů publikován roku 1992 Johnem Walkerem obsahuje pět testů [32]:

- výpočet entropie,
- statistický χ^2 test,
- aritmetický průměr,
- Monte Carlo hodnota π,
- sériový korelační koeficient.

4.2.2 Diehard Testy

Baterie testů vyvinutá Georgem Marsagliem vydaná v roce 1995 původně obsahovala 15 testů, které byly v novější verzi doplněny o další dva. Mezi nedostatky sady patří absence doporučení pro interpretaci výsledků a testy, které vyžadují různé délky vstupu a nadbytečnou část netestují. Robert G. Brown vydal rozšířenou baterii testů „DIEHARDER“, která kromě původních upravených testů obsahuje tři testy NISTu (frekvenční test, test sekvencí bitů a test sérií) a dalších šest vlastních testů [33].

4.2.3 TestU01

TestU01 je softwarová knihovna implementována v jazyce ANSI C určená pro testování generátorů náhodných čísel. Byla vydána v roce 2007. Autory jsou Pierre L'Ecuyer a Richard Simard z Univerzité de Montréal.

Knihovna implementuje řadu testů překrývajících se s testy NIST, klasické statistické testy i originální testy. Oproti testům Diehard není nutné, aby testovací vstup měl formu 32-bitových celých čísel, ale může mít i méně. Další výhodou je, že vstup může být ve formě binárního souboru nebo čísel na intervalu $(0, 1)$ [34].

Před testováním je možné si vybrat sadu předdefinovaných testů, na které poté závisí počet aplikovaných testů a čas samotného testování. Sada „SmallCrush“ obsahuje 10 testů a může trvat přibližně 8 sekund, „Crush“ 96 testů trvajících přibližně 30 minut a největší sada „BigCrush“ aplikuje 106 testů a trvá více než hodinu [35].

4.2.4 Testovací sada NIST

Celým názvem „Sada statistických testů pro generátory náhodných a pseudonáhodných čísel pro kryptografické aplikace“ je soubor testů vydán Národním institutem standardů a technologie (NIST) jako speciální publikace 800-22 v dubnu roku 2010. Obsahuje popis 15 testů (podrobně vysvětlených v kapitole 5) vhodných k detekci mnoha druhů nenáhodnosti, které se můžou objevit jak u generátorů typu TRNG, tak i PRNG.

U každého testu popisuje jeho význam, referenční distribuční funkci, rozhodovací hladinu testu, interpretaci výsledků, minimální velikost vstupu a příklad [1].

Přestože jsou NIST testy považovány za standard, tak mají mnoho nedostatků, jako například velkou časovou náročnost. Optimalizací testů se zabývá projekt fakulty informatiky Masarykovu univerzity dostupný ze zdroje [36]. Jeho autoři dosáhli průměrného třicetinásobného zrychlení testů s využitím vyhledávacích tabulek, extrakce čísel z pole bajtů (namísto bitů) a operací „slovo–slovo“ namísto „bit–bit“ [37].

O testech bylo také vydáno několik kritických publikací. Například autoři článku [38] je označují jako „zjevně zastaralé, možná škodlivé“. Problém testů vidí v poskytnutí falešného pocitu bezpečí při úspěšném vykonání testování. Testy také neřeší možné útoky postranními kanály a samotnou konstrukci generátoru a jeho zdroj entropie. Série publikací NIST SP 800-90 se těmto tématům věnuje, ale do nedostatečné hloubky a neposkytuje systematickou sadu testů jako publikace 800-22.

5 Statistický soubor testů NIST

V následujících podkapitolách jsou detailně popsány jednotlivé testy s důrazem na popis implementace pro navržení vlastní aplikace na testování náhodných čísel (viz kapitola 6). Všechny testy jsou rozhodovány na hladině významnosti 1%. Veškerý popis testů vychází ze speciální publikace NIST SP 800-22 revize 1a [1].

5.1 Frekvenční test

V angličtině „*The Frequency (Monobit) Test*“ porovnává poměr jedniček a nul v celé sekvenci. V případě, že je poměr přibližně stejný jako v opravdu náhodné sekvenci, tedy 1/2, tak sekvence testem projde a můžeme pokračovat s dalšími testy.

Funkce je volána jako „Frequency(n)“, kde n je délka řetězce bitů a dodatečným vstupem je ϵ[1], což je vygenerovaná sekvence bitů; $\epsilon = \epsilon_1, \epsilon_2, \ldots, \epsilon_n$.

Testovací kritérium S_{obs} je absolutní hodnota sumy X_i, kde $X_i = 2\epsilon_i - 1 = \pm 1$ v sekvenci vydělené odmocninou délky sekvence. Referenční distribuční funkce testovacího kritéria je semi-normální (pro velká n).

Popis testu:

1. Konverze na ± 1: bity vstupní sekvence ϵ jsou převedeny na hodnoty -1 a $+1$ a sečteny jako $S_n = X_1 + X_2 + \cdots + X_n$, kde $X_i = 2\epsilon_i - 1$.

2. Výpočet statistiky: $s_{obs} = \frac{|S_n|}{\sqrt{n}}$.

3. Výpočet *p-hodnoty* $= erfc\left(\frac{s_{obs}}{\sqrt{2}}\right)$, kde *erfc* je doplňková chybová funkce (viz rovnice 5.2).

Testovací kritérium na hladině významnosti 1%: jestliže je *p-hodnota* $< 0,01$, tak považujeme sekvenci za nenáhodnou, v opačném případě je prohlášena za náhodnou. Doporučená minimální délka vstupu je 100 bitů (tedy $n \geq 100$).

5.2 Blokový frekvenční test

„*Frequency Test within a Block*“ se obdobně jako frekvenční test zaměřuje na poměr jedniček a nul v sekvenci, ale až po rozdělení na M-bitové bloky, ve kterých se poměr má blížit k $M/2$.

Funkce je volána jako „BlockFrequency(M, n)“, kde M je délka každého bloku, n je délka řetězce bitů, dodatečný vstup je ϵ, což je vygenerovaná sekvence bitů; $\epsilon = \epsilon_1, \epsilon_2, \ldots, \epsilon_n$.

[1] Vstup ϵ je dodán kódem, jde o globální strukturu v době volání funkce

Testovací kritérium $\chi^2(obs)$ udává jak moc poměr jedniček v M-bitovém bloku odpovídá očekávané hodnotě 1/2. Referenční distribuční funkce pro test je χ^2 rozdělení.

Popis testu:

1. Rozdělení sekvence do $N = \frac{n}{M}$ bloků, nevyužité bity se zahodí.

2. Výpočet poměru jedniček v bloku π_i podle vztahu $\pi_i = \frac{\sum_{j=1}^{M} \varepsilon_{(i-1)M+j}}{M}$, pro $1 \leq i \leq N$.

3. Výpočet χ^2 statistiky: $\chi^2(obs) = 4M \sum_{i=1}^{N} \left(\pi_i - \frac{1}{2}\right)^2$.

4. Výpočet *p-hodnoty* $= igmac(N/2, chi^2(obs)/2)$, kde *igmac* je nekompletní gamma funkce (viz vzorec 5.4).

Na hladině významnosti 1% je testovací kritérium popsáno: jestliže je *p-hodnota* $< 0,01$ považujeme sekvenci za nenáhodnou, jestliže naopak, tak sekvenci považujeme za náhodnou. Doporučená délka vstupu je 100 bitů (tedy $n \geq 100$). Velikost bloku M by měla být podle doporučení zvolena jako $M \geq 20$, $M > 0,01n$ a $N < 100$.

5.3 Test sekvencí bitů

„*Runs Test*“ se zaměřuje na počty nepřerušených sekvencí identických bitů (tzv. běhů). Běh délky k se skládá z k identických bitů po sobě uzavřených bitem opačné hodnoty před i za sekvencí. Smyslem je zjistit, zdali střídání bitů různé hodnoty je moc rychlé, nebo pomalé a jestli se blíží očekávané oscilaci v náhodné sekvenci.

Volání funkce je „Runs(n)“, kde n je délka bitové sekvence a dodatečný vstup ϵ je vygenerovaná sekvence bitů.

Testovací statistikou je $V_n(obs)$; celkový počet běhů (součet běhů jedniček i nul) napříč n bity. Referenční distribuční funkcí je χ^2 rozdělení.

Popis testu:

1. Výpočet poměru jedniček ve vstupní sekvenci $\pi = \frac{\sum_j \epsilon_j}{n}$.

2. Rozhodnutí, jestli je splněn test frekvenční, který je prerekvizitou tohoto testu, tedy když platí $|\pi - 1/2| \geq \tau$, kdy $\tau = \frac{2}{\sqrt{n}}$, tak se v testu dál nepokračuje. Jestliže je hodnota π uvnitř daných mezí, tak test proběhne.

3. Výpočet statistiky $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$, kde $r(k) = 0$ jestliže platí podmínka $\epsilon_k = \epsilon_{k+1}$, v opačném případě $r(k) = 1$.

4. Výpočet *p-hodnoty* $= erfc \frac{|V_n(obs) - 2n\pi(1-\pi|}{2\sqrt{2n}\pi(1-\pi)}$, kde *erfc* je doplňková chybová funkce uvedena v rovnici 5.2.

Jestliže je *p-hodnota* $< 0,01$, tak je sekvence považována za nenáhodnou. Doporučená minimální délka testovací sekvence je 100 bitů.

5.4 Test nejdelší sekvence jedniček v bloku

„*Test for the Logest Run of Ones in a Block*“ podobně jako předchozí test zkoumá počty a délky „běhů“ jedniček, tentokrát ale v M-bitových blocích.

Funkce je volána jako „LongestRunsOfOnes(n)“, kde n je délka bitové sekvence, dodatečné vstupy jsou sekvence bitů ϵ, počet bloků N, který je dopočítán po zvolení velikosti bloků M. Velikost bloku M je zvolena dle podmínek v tabulce A.1, která je uvedena v příloze práce.

Testovací statistikou je $\chi^2(obs)$, která vyjadřuje jak moc běhy v jednotlivých blocích odpovídají očekávaným hodnotám. Referenční distribuční funkcí je χ^2 rozdělení.

Popis testu:

1. Rozdělení sekvence do M-bitových bloků.

2. Přiřazení frekvencí v_i nejdelších běhů jedniček v každém bloku do kategorií, kde každá buňka obsahuje počet běhů jedniček dané délky. Hodnoty jsou uvedeny v tabulce frekvencí A.2 podle velikosti bloku.

3. Výpočet $\chi^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, kde hodnota π_i je uvedena v sekci A.1. Hodnoty K a N jsou určeny podle hodnoty M z tabulky A.3.

4. Výpočet *p-hodnoty* $= igmac(\frac{K}{2}, \frac{\chi^2(obs)}{2})$, kde *igmac* je doplňková chybová funkce (viz 5.4).

V případě, že je vypočítaná *p-hodnota* $< 0,01$, tak je sekvence považována za nenáhodnou.

5.5 Test binárních matic

„*Binary Matrix Ranks Test*“ testuje lineární závislost mezi řetězci fixní délky po rozdělení sekvence na disjunktní submatice.

Funkce je volána jako „Rank(n)“, kde n je délka sekvence. Dodatečné vstupy jsou sekvence ϵ, počet řádků a sloupců matice M a Q, které jsou v této verzi stanoveny na 32.

Testovacím kritériem je $\chi^2(obs)$ a referenční distribuční funkcí je χ^2 rozložení.

Popis testu:

1. Rozdělení sekvence do N bloků s počtem řádků M obsahujících Q bitů. Počet bloků bude $N = \left\lfloor \frac{n}{MQ} \right\rfloor$. Nevyužité bity jsou zahozeny.

2. Výpočet hodnosti každé matice R_l, kde $l = 1, \ldots, N$.

3. Přiřazení F_M počtu matic s $R_l = M$ (maximální hodnost), F_{M-1} = počet matic s $R_l = M - 1$ (maximální hodnost -1). Počet zbývajících matic = $N - F_M - F_{M-1}$.

4. Výpočet $\chi^2(obs) = \frac{(F_M - 0{,}2888N)^2}{0{,}2888N} + \frac{(F_{M-1} - 0{,}5776N)^2}{0{,}5776N} + \frac{(N - F_M - F_{M-1} - 0{,}1336N)^2}{0{,}1336N}$.

5. Výpočet $p\text{-}hodnoty = e^{-\chi^2(obs)/2}$.

Jestliže je $p\text{-}hodnota < 0,01$, je sekvence považována za nenáhodnou. Minimální doporučený počet matic je 38. Pro velikosti matic $M = Q = 32$ musí tedy testovaná sekvence obsahovat minimálně 38912 bitů.

5.6 Test diskrétní Fourierovy transformace

„*Discrete Fourier Transform (Spectral) Test*“ se zaměřuje na vrcholy v diskrétní Fourierově transformaci vygenerované sekvence. Periodické vlastnosti transformované sekvence indikují nenáhodnost počátečního vstupu.

Funkce je volána jako „DiscreteFourierTransform(n)“, kde n je délka vstupu a dodatečný vstup je samotná sekvence ϵ.

Testovací kritérium d je normalizovaný rozdíl mezi očekávanými a pozorovanými hodnotami za hranicí 95 %. Referenční distribucí je normální rozložení.

Popis testu:

1. Bity vstupní sekvence ϵ jsou převedeny na hodnoty -1 a $+1$ a tvoří sekvenci $X = x_1, x_2, \ldots, x_n$; kde $X_i = 2\epsilon_i - 1$.

2. Aplikace diskrétní Fourierovy transformace (DFT): $S = \mathrm{DFT}(X)$.

3. Výpočet $M = modulus(S') \equiv |S'|$, kde S' je prvních $n/2$ prvků v S a funkce modulo vytváří sekvenci výšek vrcholů.

4. Výpočet $T = \sqrt{\left(\log_{10} \frac{1}{0{,}05}\right) n}$, kde T je hranice výšek vrcholů, kterou by v náhodné sekvenci hodnot nemělo 95 % hodnot překročit.

5. Výpočet $N_o = 0,95n/2$, kde N_o je očekávaný počet vrcholů.

6. Přiřazení N_1 jako počet naměřených vrcholů.

7. Výpočet $d = \frac{(N_1 - N_0)}{\sqrt{n(0,95)(0,05)/4}}$.

8. Výpočet *p-hodnoty* $= erfc\left(\frac{|d|}{\sqrt{2}}\right)$.

Je-li *p-hodnota* $< 0,01$, je sekvence považována za nenáhodnou. Minimální doporučená délka vstupu je 1000 bitů.

5.7 Test nepřekrývajících se vzorů

„*Non-Overlapping Template Matching Test*“ se zaměřuje na počet opakování určitého vzoru napříč celou sekvencí. Tento test a test překrývajících se vzorů 5.8 hledají m-bitový vzor v m-bitovém okně. V případě že vzor není nalezen, okno se posune o jednu pozici. V případě že je nalezen, se okno posune o bit za daný nalezený úsek a pokračuje dál s hledáním.

Funkce je volána jako „NonOverlappingTemplateMatching(m, n)“, kde m je délka hledaného vzoru a n je délka testované sekvence. Dodatečný vstup je vygerenovaná sekvence ϵ, hledaný vzor B o délce m, počet nezávislých bloků N – v kódu nastaveno fixně na 8.

Testovací kritérium $\chi^2(obs)$ udává, jak moc výskyt hledaného vzoru odpovídá očekávání v náhodné sekvenci. Referenční distribuční funkcí je χ^2 rozložení.

Popis testu:

1. Rozdělení sekvence do N nezávislých bloků délky M.

2. Počet výskytu vzoru B je označen $W_j (j = 1, \ldots, N)$.V m-bitovém okně se hledá shoda se vzorem. V případě neshody se okno posune o jeden bit dále, v případě shody se okno posune o m bitů dále.

3. Výpočet teoretického průměru $\mu = (M - m + 1)/2^m$ a rozptylu $\sigma = M\left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right)$.

4. Výpočet $\chi^2(obs) = \sum_{j=1}^{N} \frac{(W_j - \mu)^2}{\sigma^2}$.

5. Výpočet *p-hodnoty* pro každý vzor: *p-hodnota* $= igmac\left(\frac{N}{2}, \frac{\chi^2(obs)}{2}\right)$.

Jestliže je *p-hodnota* $< 0,01$, je sekvence považována za nenáhodnou. Je doporučeno používat vzor velikosti 9 nebo 10. Počet bloků N je v kódu definován jako 8, ale je možné ho měnit aby platilo $N \leq 100$.

5.8 Test překrývajících se vzorů

„*Overlapping Template Matching Test*“ stejně jako Test nepřekrývajících se vzorů porovnává výskyt určitého vzoru ve vygenerované sekvenci. Hlavní rozdíl je, že v případě nalezení shody se m-bitové okno neposune o m bitů, ale pouze o jeden.

Funcke je volána jako „OverlappingTemplateMatching(m, n)“, kde m je délka vzoru, n počet bitů sekvence. Dodatečnými vstupy je sekvence ϵ, vzor k porovnávání B, počet stupňů volnosti K (fixní hodnota 5), délka částí sekvence M (fixní hodnota 1032) a počet nezávislých bloků N (stanoveno na 968).

Testovacím kritériem je $\chi^2(obs)$, které vyjadřuje jak moc pozorovaná shoda vzoru a sekvence odpovídá náhodnému výskytu tohoto vzoru. Referenční distribuční funkcí je χ^2 rozložení.

Popis testu:

1. Rozdělení sekvence do N nezávislých bloků délky M.

2. Výpočet počtu shody se vzorem B v každém z bloků. Při shodě m-bitového okna a m-bitového vzoru je B inkrementováno o jedna a okno se posune o jeden bit. V případě neshody se okno pouze posune bez navýšení B. Po kontrole každého bloku je navýšeno v_i, kde $i = 0, \ldots, 5$. V případě že v bloku není shoda je k v_0 přičtena jednička, v případě jedné shody k v_1, ... a v případě ≥ 5 shod je přičtena k v_5.

3. Výpočet $\lambda = (M-m+1)/2^m$ a $\eta = \lambda/2$. Tyto hodnoty jsou využity při výpočtu teoretických pravděpodobností π_i (uvedeny v sekci A.2) pro odpovídající třídy v_i.

4. Výpočet $\chi^2(obs) = \sum_{i=0}^{5} \frac{(v_i - N\pi_i)^2}{N\pi_i}$.

5. Výpočet $p\text{-}hodnota = igmac\left(\frac{N}{2}, \frac{\chi^2(obs)}{2}\right)$.

Je-li $p\text{-}hodnota < 0,01$, tak je sekvence považována za nenáhodnou. Délka vzoru m je doporučena minimálně jako 9 nebo 10, a hodnoty K, M, N jsou voleny tak, aby byla testovaná sekvence dlouhá nejméně 10^6 bitů.

5.9 Maurerův univerzální statistický test

„*Maurer's Universal Statistical Test*“ se zaměřuje na počet bitů nacházejících se mezi stejnými vzory. Pokud je možné sekvenci zkomprimovat (mezi opakováním stejného vzoru se nachází málo rozdílných bitů), je považována za nenáhodnou.

Funkce je volána jako „Universal(L, Q, n)“, kde L je délka bloku, Q počet bloků, n délka sekvence a dodatečný vstup ϵ je vygenerovaná sekvence bitů.

Testovacím kritériem je f_n: součet $\log_2$ vzdáleností mezi souhlasnými L-bitovými vzory. Referenční distribuční funkcí je semi-normální rozdělení.

Popis testu:

1. Rozdělení n-bitové sekvence na inicializační část Q L-bitových nepřekrývajících se bloků a K L-bitových testovaných bloků. Nadbývající bity na konci

sekvence jsou zahozeny.

2. Vytvoření tabulky pro každou kombinaci L-bitových hodnot v inicializační části a přiřazení čísla bloku posledního výskytu této hodnoty $i = T_j$.

3. Průchod testovací částí a zápis počtu bloků od posledního výskytu dané kombinace do tabulky $(i - T_j)$. Zápis v tabulce se nahradí číslem bloku aktuálního výskytu $(T_j = i)$ a hodnota mezi těmito bloky je přičtena do sumy log_2 všech takovýchto rozdílů v K blocích.

4. Výpočet testovacího kritéria: $f_n = \frac{1}{K}\sum_{i=Q+1}^{Q+K} \log_2(i - T_j)$, kde T_j je hodnota z tabulky reprezentující decimální hodnotu i-tého L-bitového bloku.

5. Výpočet *p-hodnota* $= erfc\left(|\frac{f_n - L_{\text{expected}}}{\sqrt{2}\sigma}|\right)$, kde L_{expected} je teoretická očekávaná hodnota z tabulky (uvedena v sekci A.3) pro danou L-bitovou délku. Teoretická směrodatná odchylka $\sigma = c\sqrt{\frac{rozptyl(L)}{K}}$, kde $c = 0,7 - \frac{0,8}{L} + (4 + \frac{32}{L})\frac{K^{-3/L}}{15}$.

Je-li *p-hodnota* $< 0,01$, tak je sekvence považována za nenáhodnou. Test vyžaduje dlouhé sekvence bitů, doporučeny jsou velikosti dle tabulky v příloze A.3.

5.10 Test linární složitosti

„*Linear Complexity test*“ se zaměřuje na délku posuvného registru s lineární zpětnou vazbou (LFSR – Linear Feedback Shift Register). Delší LFSR indikuje náhodnou sekvenci, kratší nenáhodnou.

Volaní funkce je „LinearComplexity(M, n)“, kde M je délka bloku, n délka vstupní sekvence, ϵ samotná sekvence a K je počet stupňů volnosti (fixně 6).

Testovacím kritérium $\chi^2(obs)$ udává, jak určité délky LFSR odpovídají výskytu v náhodné sekvenci. Referenční distribuční funkce je χ^2 rozdělení.

Popis testu:

1. Rozdělení sekvence do N nezávislých bloků o M bitech, kde $n = M \cdot N$.

2. Určení lineární komplexity L_i každého z N bloků pomocí Berlekam-Massey algoritmu 5.16. L_i je délka nejkratší sekvence posuvného registru která generuje všechny bity i-tého bloku. V sekvenci L_i některé součtu bitů modulo 2 produkují následující bit $L_i + 1$.

3. Výpočet teoretického průměru: $\mu = \frac{M}{2} + \frac{\left(9+(-1)^{M+1}\right)}{36} - \frac{M/3+2/9}{2^M}$.

4. Výpočet hodnoty T_i pro každý podřetězec: $T_i = (-1)^M \cdot (L_i - \mu) + 2/9$.

5. Navýšit hodnoty v_k podle vypočítané T_i dle tabulky 5.1:

Tab. 5.1: Navýšení hodnot v_k u testu lineární složitosti.

$T_i \leq -2,5$	navýšení v_o o jedna
$-2,5 < T_i \leq -1,5$	navýšení v_1 o jedna
$-1,5 < T_i \leq -0,5$	navýšení v_2 o jedna
$-0,5 < T_i \leq 0,5$	navýšení v_3 o jedna
$0,5 < T_i \leq 1,5$	navýšení v_4 o jedna
$1,5 < T_i \leq 2,5$	navýšení v_5 o jedna
$T_i > 2,5$	navýšení v_6 o jedna

6. Výpočet $\chi^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, hodnoty π_i uvedeny v podkapitole A.4.

7. Výpočet $p\text{-}hodnota = igmac\left(\frac{K}{2}, \frac{\chi^3(obs)}{2}\right)$.

Je-li $p\text{-}hodnota < 0,01$; je sekvence považována za nenáhodnou. Minimální doporučené hodnoty jsou: $n \geq 10^6$, hodnota M v rozmezí $500 \leq M \leq 5000$ a $N \geq 200$.

5.11 Test sérií

„*Serial Test*“ hledá frekvence výskytu všech možných překrývajících se m-bitových vzorů. Cílem testu je porovnat, jestli je počet výskytů 2^m m-bitových vzorů odpovídá předpokládanému výskytu v náhodné sekvenci. Při zvolení $m = 1$ je test degradován na frekvenční test (5.1).

Funkce je volána jako „Serial(m, n)“, kde m je délka bloku, n délka vstupní sekvence a ϵ vygenerovaná sekvence bitů.

Testovacím kritériem je $\nabla\psi_m^2(obs)$ a $\nabla^2\psi_m^2(obs)$ vyjadřující jak moc sledované frekvence výskytu vzoru odpovídají předpokládanému výskytu v náhodné sekvenci. Referenční distribuční funkcí je χ^2 rozložení.

Popis testu:

1. Rozšíření sekvence na ϵ': prvních $m - 1$ bitů je přidáno na konec.

2. Určení počtu výskytu m-bitových bloků jako $v_{i_1}, \ldots, v_{i_m}$, (m-1)-bitových bloků $v_{i_1}, \ldots, v_{i_{m-1}}$ a (m-2)-bitových bloků jako $v_{i_1}, \ldots, v_{i_{m-2}}$.

3. Výpočet:

$$\psi_m^2 = \frac{2^m}{n} \sum_{i=i_1}^{i_m} \left(v_i - \frac{n}{2^m} \right) = \frac{2^m}{n} \sum_{i=i_1}^{i_m} v_i^2 - n,$$

$$\psi_{m-1}^2 = \frac{2^{m-1}}{n} \sum_{i=i_1}^{i_{m-1}} \left(v_i - \frac{n}{2^{m-1}} \right) = \frac{2^{m-1}}{n} \sum_{i=i_1}^{i_{m-1}} v_i^2 - n,$$

$$\psi_{m-2}^2 = \frac{2^{m-2}}{n} \sum_{i=i_1}^{i_{m-2}} \left(v_i - \frac{n}{2^{m-2}} \right) = \frac{2^{m-2}}{n} \sum_{i=i_1}^{i_{m-2}} v_i^2 - n.$$

4. Výpočet testovacích kritérií:

$$\nabla \psi_m^2 = \psi_m^2 - \psi_{m-1}^2 a$$

$$\nabla^2 \psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2.$$

5. Výpočet *p-hodnot*:

$$p - hodnota1 = igmac\left(2^{m-2}, \nabla \psi_m^2\right),$$

$$p - hodnota2 = igmac\left(2^{m-3}, \nabla^2 \psi_m^2\right),$$

Je-li $p\text{-}hodnota < 0,01$; je sekvence považována za nenáhodnou. Výběr m a n musí být v souladu s podmínkou $m < \lfloor \log_2 n \rfloor - 2$.

5.12 Test přibližné entropie

„*Approximate Enotropy Test*“ se obdobně jako „Test sérií“ (5.11) zaměřuje na počty překrývajících se m-bitových vzorů. Smyslem testu je porovnání frekvencí výskytu dvou sousedních bloků (m a $m+1$) s očekávanými frekvencemi v náhodné sekvenci.

Funkce je volána jako „ApproximateEntropy(m, n)“, kde vstupem je délka bloku m, délka celé sekvence n a vygenerovaná sekvence ϵ.

Testovací kritérium $\chi^2(obs)$ udává, jak moc pozorované hodnoty výskytu odpovídají teoretickým v náhodné sekvenci. Referenční distribuční funkcí je χ^2 rozložení.

Popis testu:

1. Rozšíření původní sekvence tak, že prvních $m-1$ bitů je přidáno na konec. Tento postup je opakován pro všechny hodnoty m.

2. Označení počtu každé kombinace m-bitového vzoru jako C_i^m, kde i je každá kombinace bitů o délce m. Součet počtu výskytů dané kombinace jako $\#i$.

3. Výpočet $C_i^m = \frac{\#i}{n}$ pro všechna i.

4. Výpočet $\varphi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i \log \pi_i$, kde $\pi_i = C_j^m$ a $j = \log_2 i$.

5. Opakování kroků 1-4 s $m+1$ namísto m.

6. Výpočet $\chi^2 = 2n\left[\log 2 - ApEn(m)\right]$, kde $ApEn(m) = \varphi^{(m)} - \varphi^{(m+1)}$.

7. Výpočet $p\text{-}hodnota = igmac\left(2^{m-1}, \frac{\chi^2}{2}\right)$.

Jestliže je $p\text{-}hodnota < 0,01$, je sekvence považována za nenáhodnou. Výběr m a n je doporučen podmínkou $m < \lfloor \log_2 n \rfloor - 5$.

5.13 Test kumulativních součtů

„*Cumulative Sums Test*“ se zaměřuje na maximální odchylku od nuly náhodné procházky definované kumulativním součtem normalizované sekvence (-1 a 1). Kumulativní suma náhodné sekvence by měla být blízká nule, pro určité typy nenáhodných sekvencí bude ale vzdálená od nuly.

Funkce je volána jako „CumulativeSums($mode, n$)“, kde n je délka vstupu, ϵ vygenerovaná sekvence a $mode$ parametr udávající směr průchodu sekvencí ($mode = 0$ pro průchod od začátku, $mode = 1$ pro průchod pozpátku).

Testovací kritérium z je největší odchylka kumulativního součtu od nuly. Referenční distribuční funkcí je normální rozložení.

Popis testu:

1. Vytvoření normalizované sekvence: jedničky a nuly jsou zaměněny na hodnoty „-1“ a „$+1$“ pomocí předpisu $X_i = 2\epsilon_i - 1$.

2. Výpočet dílčích součtů S_i narůstajících podsekvencí začínajících od X_i v případě $mode = 0$, nebo od X_n v případě $mode = 1$. Součet $S_k = S_{k-1} + X_k$ pro mód 0 a $S_k = S_{k-1} + X_{n-k+1}$ pro mód 1.

3. Výpočet testové statistiky $z = \max_{1 \le k \le n} |S_k|$, kde $\max_{1 \le k \le n} |S_k|$ je maximální hodnotou z dílčích součtů S_k.

4. Výpočet $p\text{-}hodnoty$:

$$p\text{-}hodnota = 1 - \sum_{k=(\frac{-n}{z}+1)/4}^{(\frac{n}{z}-1)/4} \left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right] +$$

$$\sum_{k=(\frac{-n}{z}-3)/4}^{(\frac{n}{z}-1)/4} \left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right],$$

kde Φ je funkce standardního normálního kumulativního pravděpodobnostního rozložení viz vzorec 5.1.

Je-li *p-hodnota* $< 0,01$, je sekvence považována za nenáhodnou. Minimální doporučená délka vstupní sekvence je 100 bitů ($n \geq 100$).

5.14 Test náhodných návštěv

„*Random Excursions Test*“ se zaměřuje na počet cyklů s K návštěvami během kumulativního součtu náhodné procházky. Kumulativní součet je odvozen z dílčích součtů normalizované sekvence. Cyklus náhodné procházky je posloupnost náhodných kroků, které se vrátí zpět do počátečního stavu. Smyslem testu je porovnat jestli počet návštěv určitého stavu odpovídá teoretickému počtu návštěv v náhodné sekvenci. Test se skládá z osmi testů pro stavy: $-4, -3, -2 - 1, 1, 2, 3, 4$, které jsou označeny x.

Funkce je volána jako „RandomExcursions(n)“, kde n je délka vstupu a ϵ vygenerovaná sekvence bitů.

Kritériem testu je $\chi^2(obs)$ vyjadřující, jak počet pozorovaných návštěv v cyklu odpovídá očekávané náhodné hodnotě. Referenční distribuční funkcí je χ^2 rozdělení.

Popis testu:

1. Vytvoření normalizované sekvence: jedničky a nuly jsou zaměněny na hodnoty „-1“ a „$+1$“ předpisem $X_i = 2\epsilon_i - 1$.

2. Výpočet dílčích sum S_i začínajících od X_1 a tvořících množinu $S = S_i$.

3. Vytvoření nové sekvence S' přidáním nuly na začátek a konec množiny S.

4. Počet průchodů nuly v S' je označeno J. J označuje také počet cyklů v S', kde cyklus je podsekvence S' začínající nulou, pokračující nenulovými hodnotami a končící opět nulou. Jestliže $J < 500$, je test přerušen a sekvence vyhodnocena jako nenáhodná.

5. Výpočet počtu výskytů pro každou nenulovou hodnotu stavu x v mezích $-4 \leq x \leq -1$, nebo $1 \leq x \leq 4$ zvlášť pro každý cyklus.

6. Pro všech osm stavů x je vypočítána $v_k(x) =$ celkový počet cyklů, ve kterých se stav x vyskytuje právě k-krát, kdy $k = 0, 1, \ldots, 5$. Do $v_5(x)$, tedy když $k = 5$, jsou uloženy všechny frekvence ≥ 5. Platí tedy, že $\sum_{k=0}^{5} v_k(x) = J$.

7. Výpočet testové statistiky pro každý z osmi stavů x:

$$\chi^2(obs) = \sum_{k=0}^{5} \frac{(v_k(x) - J\pi_k(x))^2}{J\pi_k(x)},$$

kde $\pi_k(x)$ je pravděpodobnost výskytu stavu x právě k-krát v náhodném rozdělení. Konkrétní hodnoty jsou uvedeny v tabulce v podkapitole A.5-

8. Výpočet *p-hodnoty* pro každý stav x:

$$p\text{-}hodnota = igmac\left(\frac{5}{2}, \frac{\chi^2(obs)}{2}\right).$$

Je-li $p\text{-}hodnota < 0,01$, je sekvence považována za nenáhodnou. V případě neshody výsledků v různých z osmi stavů k je doporučeno testovat další generované sekvence, aby bylo zjištěno, jestli je takové chování pro daný generátor typické. Minimální doporučená délka vstupní sekvence je 10^6 bitů.

5.15 Test náhodných variant návštěv

„Random Excursions Variant Test“ se zaměřuje na počet návštěv určitého stavu při kumulativním součtu náhodné procházky. Cílem je porovnat pozorované hodnoty s očekávanými hodnotami v náhodné sekvenci. Test je podobný testu náhodných návštěv (5.14) s tím rozdílem, že test probíhá v osmnácti stavech: $-9, -8, \ldots, +8, +9$ s výjimkou nuly.

Volání funkce je „RandomExcursionVariant(n)“, kde n je délka vstupu a ϵ vygenerovaná sekvence bitů.

Testovým kritériem je ξ: počet návštěv daného stavu v průběhu náhodné procházky. Referenční distribuční funkce je semi-normální.

Popis testu:

1. Vytvoření normalizované sekvence: jedničky a nuly jsou zaměněny na hodnoty „-1“ a „$+1$“ předpisem $X_i = 2\epsilon_i - 1$.

2. Výpočet dílčích sum S_i začínajících od x_1 a tvořících množinu $S = S_i$.

3. Vytvoření sekvence S' přidáním nuly na začátek a konec množiny S.

4. Výpočet $\xi(x)$ = celkový počet výskytu stavu x napříč všemi J cykly pro všech osmnáct nenulových stavů x.

5. Výpočet *p-hodnoty* pro každé kritérium $\xi(x)$:

$$p\text{-}hodnota = erfc\left(\frac{|\xi(x) - J|}{\sqrt{2J(4|x| - 2)}}\right).$$

Je-li $p\text{-}hodnota < 0,01$, je sekvence považována za nenáhodnou. Aby mohla být sekvence považována za náhodnou, musí projít všech osmnáct částí testu. Minimální doporučená délka vstupu n je $n \geq 10^6$.

5.16 Doplňující informace pro popis testů

5.16.1 Matematické funkce

Některé testy využívají další matematické funkce. Jedná se například o gamma funkci, nekompletní gamma funkci, doplňkovou chybovou funkci a standardní normální funkci. Dále testy využívají Fourierovu transformaci a Berlekamp-Massey algoritmus.

Funkce normovaného normálního rozdělení:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{u^2}{2}} \mathrm{d}u. \tag{5.1}$$

Doplňková chybová funkce:

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_{z}^{\infty} e^{-u^2} \mathrm{d}u. \tag{5.2}$$

Gamma Funkce:

$$\Gamma(z) = \int_{0}^{\infty} t^{z-1} e^{-t} \mathrm{d}t. \tag{5.3}$$

Nekompletní gamma funkce:

$$P(a,x) \equiv \frac{\gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_{0}^{x} e^{-t} t^{a-1} \mathrm{d}t, \tag{5.4}$$

kde $P(a,0) = 0$ a $P(a,\infty) = 1$ a

$$Q(a,x) \equiv 1 - P(a,x) \equiv \frac{\Gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_{x}^{\infty} e^{-t} t^{a-1} \mathrm{d}t, \tag{5.5}$$

kde $Q(a,0) = 1$ a $Q(a,\infty) = 0$.

Tyto funkce jsou v originálním kódu NISTu implementovány matematickou knihovnou `Cephes`. Pro jazyk Python existuje rozhraní k této knohovně `ncephes`.

Fourierova transformace je v originálním kódu realizována knihovnou `fft.c`. Pro implementaci v Pythonu je možné použít funkci `fft` obsaženou v knihovně `numpy.fft`.

Berlekamp-Massey algoritmus, který vypočítá nejkratší LFSR, který generuje konečnou sekvenci prvků v daném poli, je popsán ve zdroji [10].

6 Tvorba aplikace pro testování generátorů náhodných čísel

Praktickým výstupem bakalářské práce je program, který umožňuje testování generátorů náhodných čísel.

6.1 Charakteristika aplikace

Cíl aplikace

Aplikace slouží k testování generátorů náhodných čísel. Pomocí statistických testů rozhoduje o náhodnosti vygenerovaných sekvencí, které uživatel vloží k otestování.

Programovací jazyk

Jako jazyk vhodný k vytvoření aplikace s tímto účelem byl zvolen Python. Jedná se o jazyk používaný v aplikacích zaměřených na statistiku a datovou analýzu. K dispozici je také velké množství knihoven, má širokou základnu uživatelů, studijních materiálů a je vhodný k tvorbě desktopových i webových aplikací. Alternativou by mohl být jazyk R, který vyniká ve vizuální reprezentaci dat, ale nevýhodou je méně dostupných knihoven a komplikovanější tvorba uživatelského prostředí aplikace.

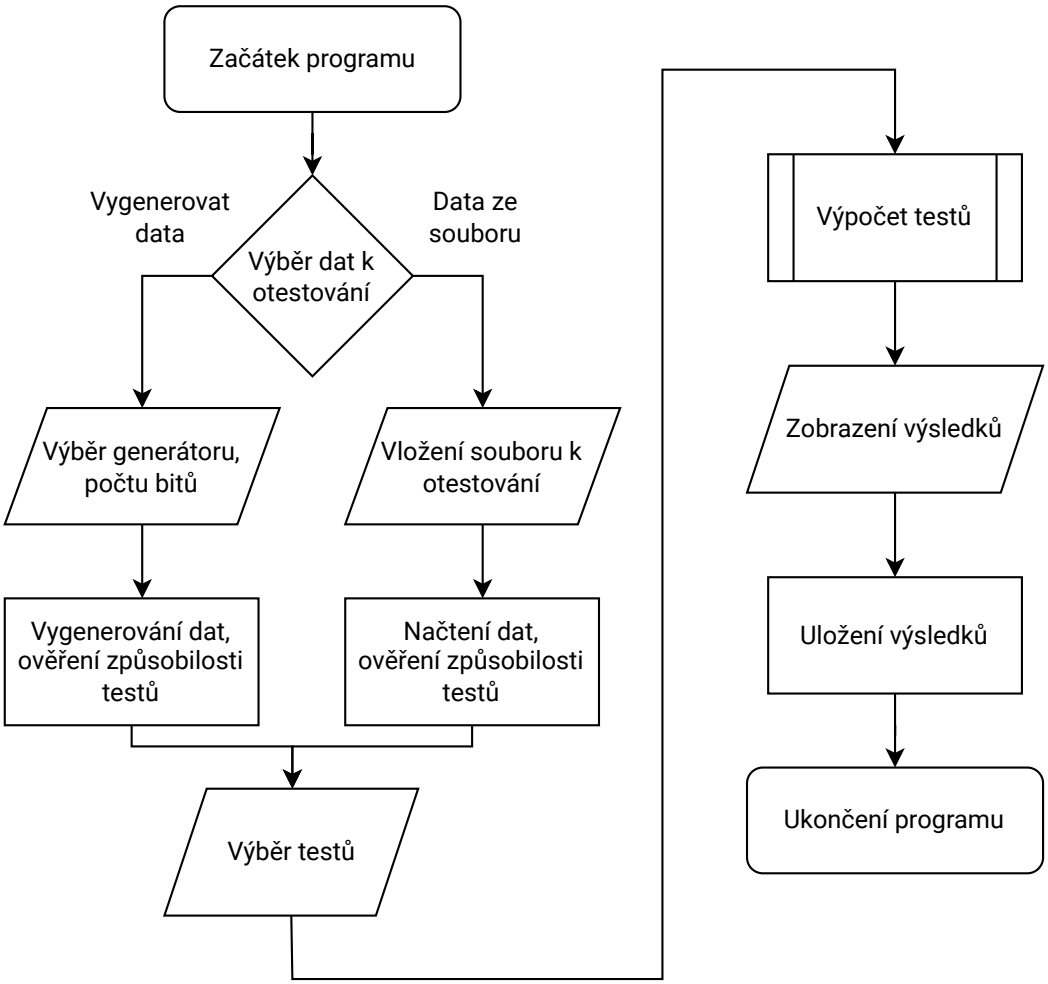
Pro vytvoření grafického prostředí byla použita knihovna PyQt5, která poskytuje nástroje pro tvorbu multiplatformních aplikací pro programovací jazyk Python. Knihovna je dostupná pod komerční licencí a GPL licencí, která dovoluje volné užití, modifikace a sdílení za podmínky zveřejnění zdrojového kódu [39].

Platforma

Aplikace může svůj účel plnit jako desktopová aplikace. Její součástí je i grafické uživatelské prostředí, díky kterému program získává i informačně-vzdělávací význam.

Funkcionalita

Funkcionalita programu je znázorněna na diagramu 6.1.



Obr. 6.1: Vývojový diagram aplikace.

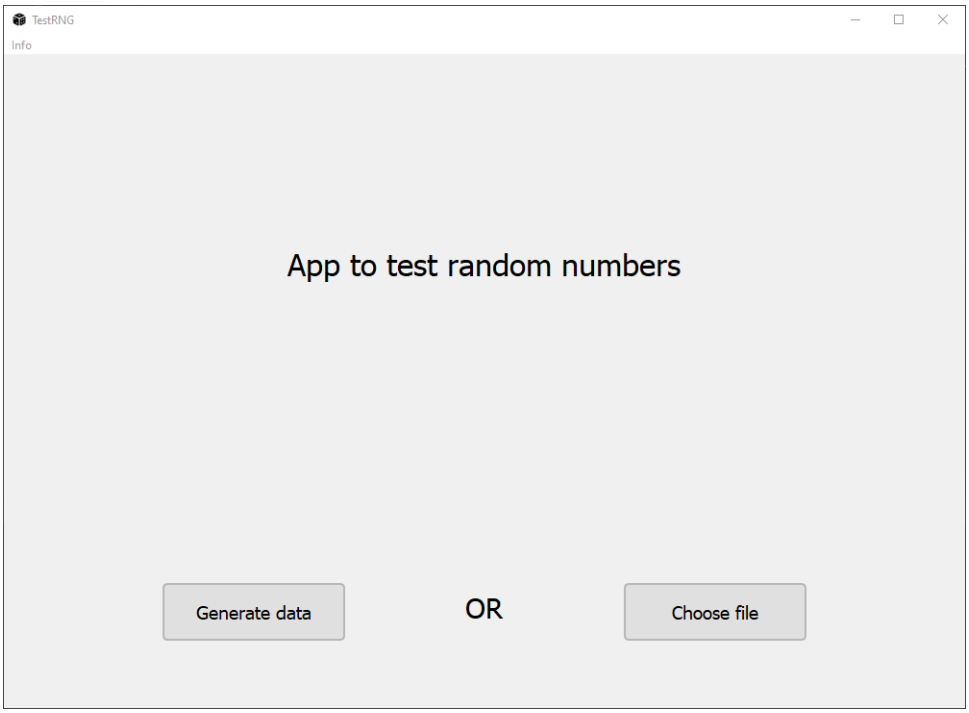
6.2 Popis ovládání aplikace

Po spuštění aplikace je zobrazeno hlavní okno (na obrázku 6.2), ve kterém si uživatel může vybrat ze dvou možností:

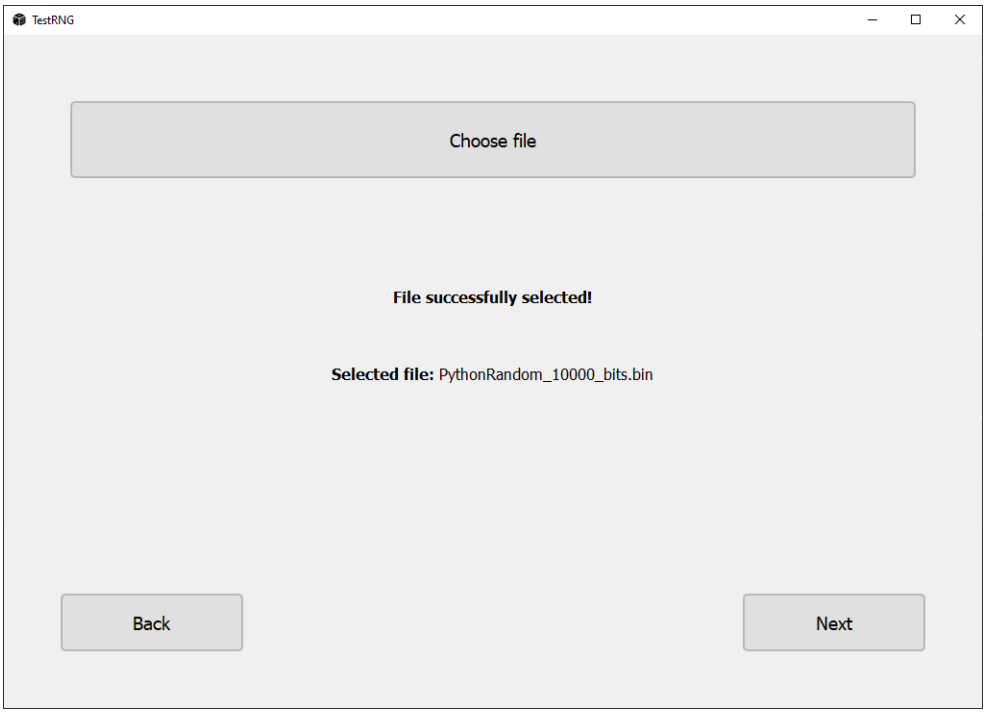
- Vložit vlastní sekvenci.
- Vygenerovat sekvenci bitů pro otestování.

V okně pro vložení souboru (viz obrázek 6.3) uživatel může vložit vlastní soubor s daty k otestování. Jsou akceptovány soubory typu `.bin` a `.txt`. Obsahem souboru mohou být bitové hodnoty, celá čísla, nebo čísla z intervalu $(0, 1)$. Jako Oddělovač hodnot je akceptována mezera, nový řádek a čárka.

V případě zvolení možnosti generování sekvence aplikací je zobrazeno okno (viz obr. 6.4) ve kterém je možné zvolit počet generovaných bitů a generátor (druhy generátorů jsou detailněji popsány v kapitole 6.4). Po najetí myší na jednotlivý generátor

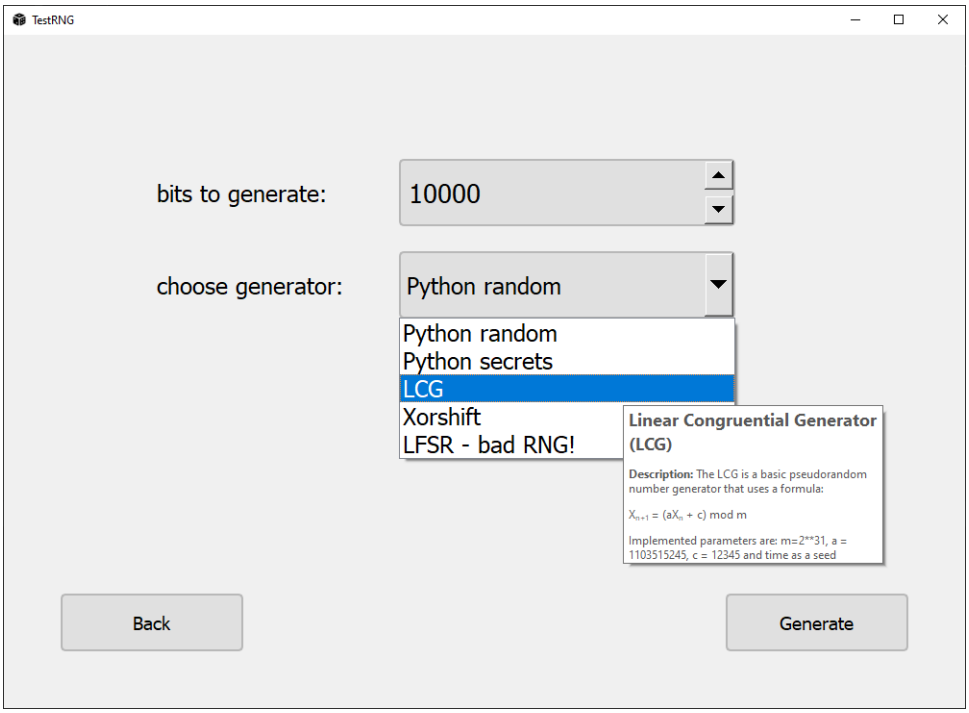


Obr. 6.2: Úvodní okno aplikace.



Obr. 6.3: Výběr souboru s daty k otestování.

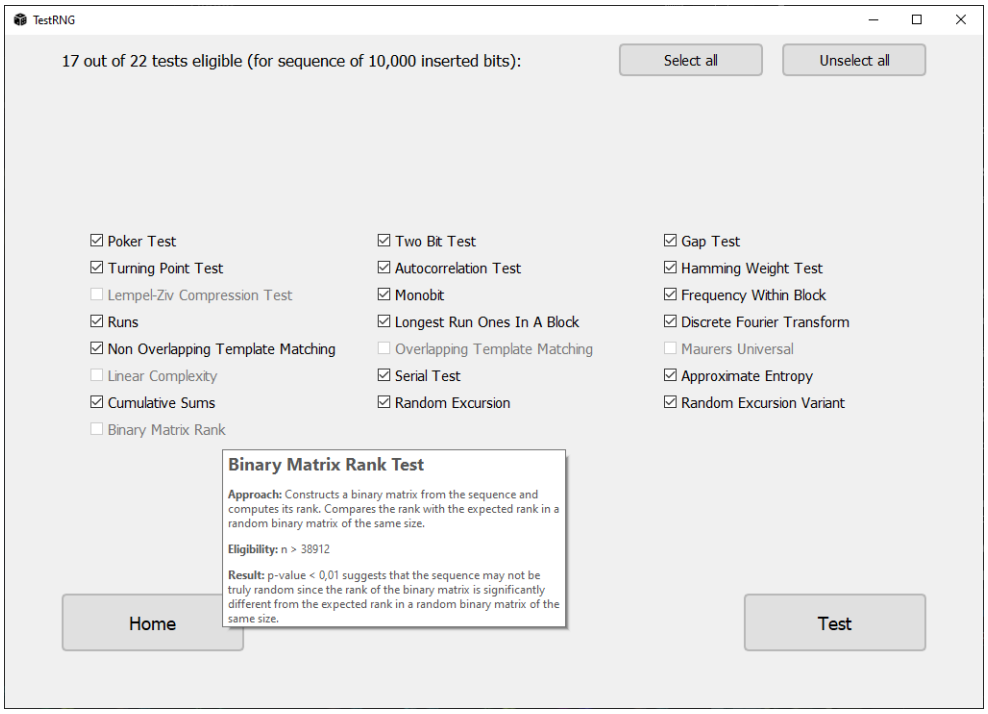
je zobrazen popis jeho principu. Postupem do dalšího okna je sekvence vygenerována, uložena ve složce `generated_data` a zároveň držena v paměti programu pro následné otestování.



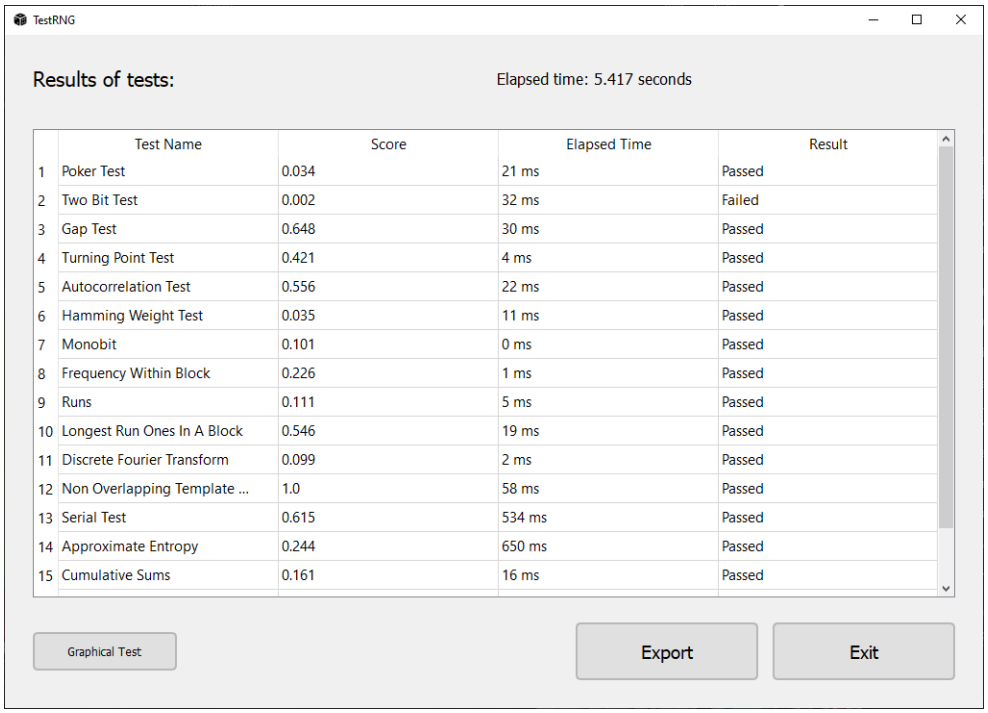
Obr. 6.4: Generování dat k otestování.

Po zvolení testovaných dat je zobrazeno okno s výběrem testů (viz obr. 6.5). Z celkové baterie testů (testy jsou uvedeny v sekci 6.3) je možné vybrat testy, které budou aplikovány. Testy u kterých nebyl splněn požadavek na minimální délku sekvence zvolit nelze. Po najetí myší na test je zobrazen jeho popis, podmínka na délku sekvence a informace k vyhodnocení výsledku.

Po výběru testů je sekvence otestována a výsledky jsou zobrazeny uživateli (okno aplikace s výsledky je zobrazeno na obrázku 6.6). V tabulce výsledků je uveden název testu, vypočítaná *p-hodnota* a čas výpočtu. Dalšími údaji je také celkový výsledek každého testu – prošel/neprošel a to na základě porovnání *p-hodnoty* s hladinou významnosti. Po najetí myší na každý test se zobrazí informace o testu a co nám výsledná *p-hodnota* říká o vlastnostech testované sekvence. Grafický test je zobrazen po stisknutí příslušného tlačítka. Výsledky testování mohou být uloženy do souboru formátu `.csv`, `.xlsx` nebo `.txt` pro další zpracování.



Obr. 6.5: Výběr z dostupných statistických testů.



	Test Name	Score	Elapsed Time	Result
1	Poker Test	0.034	21 ms	Passed
2	Two Bit Test	0.002	32 ms	Failed
3	Gap Test	0.648	30 ms	Passed
4	Turning Point Test	0.421	4 ms	Passed
5	Autocorrelation Test	0.556	22 ms	Passed
6	Hamming Weight Test	0.035	11 ms	Passed
7	Monobit	0.101	0 ms	Passed
8	Frequency Within Block	0.226	1 ms	Passed
9	Runs	0.111	5 ms	Passed
10	Longest Run Ones In A Block	0.546	19 ms	Passed
11	Discrete Fourier Transform	0.099	2 ms	Passed
12	Non Overlapping Template ...	1.0	58 ms	Passed
13	Serial Test	0.615	534 ms	Passed
14	Approximate Entropy	0.244	650 ms	Passed
15	Cumulative Sums	0.161	16 ms	Passed

Obr. 6.6: Okno s výsledky testů a možností jejich uložení.

6.3 Implementované testy

Součástí aplikace je 23 testů náhodnosti. Jedná se o 15 testů obsažených v baterii NIST a 8 dalších statistických testů. Celkem 22 z nich tvoří baterii, která sdílí podobnost ve vyhodnocování výsledků pomocí *p-hodnoty* a jejím porovnáním s hladinou významnosti $0,01$. Testy po otestování sekvence vrací *p-hodnotu*, výsledek testu (úspěch/neúspěch) a čas výpočtu. Grafický test se od této baterie odlišuje tím, že není automaticky vyhodnocován, ale pouze zobrazen uživateli k posouzení.

Aplikace obsahuje následující testy:

1. **Poker test** (4.1.3)
2. **Test autokorelace** (4.1.5)
3. **Test mezer** (4.1.6)
4. **Test bodů zvratu** (4.1.7)
5. **Lempel–Ziv kompresní test** (4.1.8)
6. **Test Hammingovy váhy** (4.1.9
7. **Test dvojic bitů** (4.1.10)
8. **Grafický test** (4.1.4)

9-23. **Testy baterie NIST** (detailně v kapitole 5). Testy této sady jsou v dnešní době standardem pro testování generátorů, proto tvoří i základ baterie testů aplikace. Pro jejich implementaci v jazyce Python je k dispozici knihovna `nistrng`. Tato knihovna je pod licencí *BSD 3-Clause License*, která patří mezi nejvolnější licence a povoluje využití i v komerčních programech. Knihovna obsahuje všech 15 testů a je v případě zjištění nedostatků aktualizována [40].

Testy jsou pro snížení doby výpočtu počítány paralelně s pomocí knihovny `multiprocessing`, která umožňuje výpočty na více jádrech procesoru. Program tedy zjistí počet logických procesorů a využije je pro výpočet více testů zároveň. Na osobním počítači s 4 jádry a 8 logickými procesory se implementace paralelních výpočtů projevila zkrácením doby výpočtu testů pro delší sekvence. Pro kratší sekvence se doba výpočtů nezkrátila, protože čas tvorby procesů je delší než doba ušetřená souběžným výpočtem. Časy před a po implementaci paralelizace jsou uvedeny v tabulce 6.1. Průměrné časy (každý záznam je aritmetickým průměrem pěti opakování) jsou v této tabulce uvedeny v závislosti na délce testované sekvence[1].

[1] Pro každý počet bitů jsou aplikovány všechny vhodné testy, takže jejich počet se liší.

Tab. 6.1: Porovnání doby trvání výpočtu testů před a po implementaci paralelních výpočtů.

Počet testovaných bitů	Čas před paralelizací (s)	Čas po paralelizaci (s)
10 000	1, 16	4, 26
100 000	11, 18	8, 95
250 000	26, 57	16, 76
500 000	52, 58	30, 16
1 000 000	390, 70	179, 83

6.4 Implementované generátory

6.4.1 Základní generátor jazyka Python

Základní generátor jazyka Python je obsažen v modulu `random`. Umožňuje zvolit rozložení generované sekvence (uniformní, normální, gamma a další). Dále obsahuje funkci generování celých čísel z intervalu, náhodné permutace listu a náhodného vzorkování. Tento modul využívá Mersenne Twister PRNG (viz 2.2.4) s periodou $2^{19937}-1$, který je implementován v jazyce C. Jde o dlouhodobě testovaný a rozšířený generátor, který je ale nevhodný pro využití v kryptografii [41]. V rámci aplikace generuje uživatelem zadaný počet bitů.

6.4.2 Modul Secrets pro jazyk Python

Pro generování kryptograficky bezpečných náhodných čísel a jejich využití v kryptografii (pro autentizaci, hesla, tokeny, ...) je součástí standardní knihovny Pythonu modul `secrets`. Byl přidán roku 2015 jako reakce na hojné využívání modulu `random` pro kryptografické účely, ke kterým není vhodný.

Samotná data jsou dodávána bezpečným generátorem operačního systému, to znamená že se jedná o obdobu využití funkce `os.urandom()`. Tato funkce generuje náhodné bajty pomocí systémového generátoru. Rozdílem je, že modul `secrets` této funkci přidává funkcionalitu na vyšších úrovních a negeneruje jen nezpracovaná data [42].

V **unixových systémech** funkce `os.urandom()` využívá systémového volání funkce `getrandom()`, která získá entropii ze souboru `/dev/urandom`, kde je shromažďována z více zdrojů, například z událostí v hardwaru, aktivity myši a klávesnice, aktivity disku aj.

Operační systém **Windows** využívá funkci `BCryptGenRandom`, která je zprostředkována skrze *Cryptography API: Next Generation (CNG)* nahrazující starší funkci `CryptGenRandom` patřící pod `Microsoft CryptoAPI`. Pro generování čísel rovněž využívá více zdrojů entropie [43].

6.4.3 Lineární kongruentní generátor

Dalším generátorem obsaženým v aplikaci je LCG (popsán v kapitole 2.2.1). Je implementován s parametry $m = 2^{31}$, $a = 1103515245$ a $c = 12345$, což jsou klasicky používané parametry, například generátorem jazyka ANSI C [44]. Jako *seed* je použit systémový čas.

6.4.4 Xorshift

Vstupním *seedem* generátoru je systémový čas a na něm jsou prováděny operace bitového posunu (viz kapitola 2.2.3). Ze vzniklého řetězce je extrahován nejméně významný bit, který je přidán do vygenerované sekvence a řetězec je dále použit jako *seed* následující iterace. Implementovaná funkce generátoru navrací sekvenci o požadované bitové délce.

6.4.5 LFSR

Jako ukázka nekvalitního generátoru je implementován generátor na základě posuvného registru s lineární zpětnou vazbou (vysvětlen v kapitole 2.2.2). Byla implementována verze s délkou registru $n = 4$ bity a polynomem $x^4 + x^3 + 1$, který generuje sekvenci bitů s periodou $2^n - 1 = 15$ bitů. Na výsledcích testování je pak možné ukázat, které testy dokáží odhalit generátor s takto malou periodou a které testy tuto záměrně nekvalitní implementaci neodhalí.

7 Hodnocení generátorů

Postup vyhodnocení výsledků testování dle publikace NISTu ([1]) je následující: m je označení počtu testovaných sekvencí daným testem, $\hat{p} = 1 - \alpha$ a poměr úspěšnosti r je podílem počtu úspěšných testů a všech provedených testů. Poměr r musí spadat do intervalu:

$$r \in \left(\hat{p} - 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}, \hat{p} + 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}} \right). \tag{7.1}$$

Jestliže poměr úspěšnosti nepatří do tohoto intervalu, tak je sekvence označena za nenáhodnou. Tento způsob vyhodnocení je ale přesný jen pro opakování každého testu alespoň tisíckrát ($m > 1000$).

Tento počet opakování je vzhledem k době výpočtů nepraktický pro uskutečnění v domácích podmínkách, proto byl zvolen postup demonstrace funkce testů opakováním testování desetkrát ($m = 10$) a následné diskuze nad výsledky. Dalším důvodem zvolení tohoto postupu je implementace dalších testů, které nemusí být vhodné pro tento způsob statistického vyhodnocení.

7.1 Testování dat z kvantové distribuce klíčů

Protokoly kvantové distribuce klíčů (kvantového ustanovení klíčů, QKD – Quantum Key Distribution) jsou skupinou kryptografických protokolů, které využívají principy kvantové mechaniky pro distribuci klíčů mezi dvěma stranami – „Alicí“ a „Bobem“. Tento druh výměny klíčů je bezpečný, protože jakýkoliv pokus o odposlech nebo změnu přenášeného klíče pozmění stav kvantové částice a je detekován.

Jde o velmi rychle se rozvíjející oblast kryptografie, takže existuje mnoho variant implementace QKD. Na Fakultě elektrotechniky a komunikačních technologií Vysokého učení technického v Brně je vytvořen testovací polygon pro QKD. Jeho topologie se skládá ze dvou QKD serverů (Alice a Boba), které jsou propojeny kvantovým a servisním kanálem. Servery jsou propojeny s šifrátory trasou pro šifrovaná data. Alice slouží jako vysílač a Bob jako přijímač. Do kvantového kanálu je také možné zapojit odposlouchávajícího útočníka „Evu“. Spotřebitel je zastoupen skriptem, který si vyžádá kvantový klíč a jeho identifikátor z obou serverů (master a slave) ve formátu JSON.

Na takto vygenerované klíče jsou také kladeny nároky na náhodnost (stejně jako na všechny kryptografické klíče), a proto byly otestovány vytvořenou aplikací na testování náhodných čísel. Polygon QKD může klíče generovat neustále, proto byly z vygenerovaných dat vybrány vzorky napříč v čase, aby odhalily případnou změnu náhodnosti. Testování bylo uskutečněno na deseti sekvencích klíčů o délce 100 000 bitů.

Kompletní výsledky testů jsou uvedeny v tabulce 7.1. Jsou zde ve sloupcích uvedeny výsledky jednotlivých iterací testování. U každého testu je uvedena výsledná *p-hodnota*, výsledek testu („P“ jako prošel a „N“ neprošel) a v posledním sloupci celkový poměr úspěšnosti u daného testu.

Data z QKD bez problémů obstojí ve většině aplikovaných testů. Jedna z deseti sekvencí neprošla testem mezer – její počty délek mezer (hodnot „0“ mezi dvěma „1“) neodpovídají očekávaným počtům v opravdu náhodné sekvenci. Jeden test skončil neúspěšně u testu bodů zvratu, což indikuje nenáhodný počet výskytu bodů zvratu a dva z deseti u testu Hammingovy váhy, což znamená nenáhodné rozložení Hammingovy váhy v osmicích bitů. Celkový poměr úspěšnosti je 156/160 testů, tedy $97,5\,\%$. Na základě výsledků testování sekvencí (a srovnání s dalšími generátory v následující kapitole) data mohou být označena jako náhodná.

Pro implementaci v kryptografických aplikacích by bylo vhodné data otestovat nejen statistickými testy, ale například analýzou postranních kanálů, útokem na konstrukci generátoru, nebo otestovat možnost predikce následujících bitů na základě znalosti bitů předchozích.

7.2 Srovnání implementovaných pseudonáhodných generátorů

Vytvořenou aplikací byly otestovány i sekvence produkované generátory obsaženými v rámci aplikace. Pro každý generátor bylo vygenerováno 10 sekvencí o délce 100 000 bitů. Kompletní výsledky testování generátorů jsou uvedeny v přílohách – Python random v tabulce B.1, Python secrets v tabulce B.2, LCG v tabulce B.3, Xorshift v tabulce B.4 a výsledky generátoru LFSR v tabulce B.5.

Tabulka 7.2 uvádí celkové výsledky všech generátorů a porovnává jejich úspěšnost. Všechny generátory (až na nekvalitně implementovaný generátor LFSR) dopadly s obdobně uspokojivými výsledky. Testy neprošly jen v jednotkách testů a převážně se jedná o testy nepatřící do baterie NIST. To může znamenat to, že přidané testy mají přísnější kritéria než testy NISTu.

Na generátoru LFSR s periodou 15 bitů je možné ukázat, kterými testy takto nekvalitně implementovaný generátor neprojde. Například u Frekvenčního testu vždy neprojde, protože vygenerovaná sekvence s lichou periodou obsahuje vždy rozdílný počet „1“ a „0“ (v tomto případě 7 a 8), což ve dlouhé sekvenci znamená disproporci mezi počtem jedniček a nul. Generátor má u testů úspěšnost jedině 0/10, nebo 10/10. To je způsobeno tím, že vstupní *seed* v periodickém generátoru ovlivní jen to, od které hodnoty bude začínat, ale další hodnoty se budou vždy opakovat ve stejném pořadí. Generátor prošel jen testem bodů zvratu, který nezohledňuje proporci nul

a jedniček ani periodicitu, ale pouze body zvratu, které s těmito parametry přímo nesouvisí. Podobným případem je úspěch u testu nepřekrývajících se vzorů, který porovnává vzory v blocích o délce 8 bitů, na kterých se nedostatky generátoru také neprojeví.

Další skutečností je to, že základní generátor Pythonu `random` a bezpečný kryptografický generátor `secrets` dosáhly stejných výsledků, i přestože modul `secrets` je prezentován jako bezpečnější a kvalitnější. Důvodem je, že i modul *random* má vysokou periodu a produkuje náhodná čísla. Rozdíl mezi těmito moduly by se projevil při reverzním inženýrství sekvence (snaze o zjištění pokračování aktuální sekvence) nebo při snaze o zjištění *seedu*, na což by modul `random` mohl být na rozdíl od modulu `secrets` více náchylný.

Tab. 7.1: Výsledky testování klíčů z kvantového přenosu dat v deseti iteracích po 100 000 bitech.

Název testu	**Iterace testu**										**Úspěšnost**
Poker test	0,905 (P)	0,864 (P)	0,080 (P)	0,769 (P)	0,283 (P)	0,848 (P)	0,704 (P)	0,438 (P)	0,452 (P)	0,361 (P)	10/10
Test dvojic bitů	0,705 (P)	0,411 (P)	0,503 (P)	0,675 (P)	0,558 (P)	0,034 (P)	0,131 (P)	0,265 (P)	0,044 (P)	0,818 (P)	10/10
Test mezer	0,408 (P)	0,259 (P)	0,685 (P)	0,348 (P)	0,229 (P)	0,030 (P)	0,115 (P)	0,001 (N)	0,278 (P)	0,875 (P)	9/10
Test bodů zvratu	0,567 (P)	0,243 (P)	0,001 (N)	0,611 (P)	0,130 (P)	0,027 (P)	0,641 (P)	0,641 (P)	0,252 (P)	0,049 (P)	9/10
Test autokorelace	0,305 (P)	0,567 (P)	0,415 (P)	0,631 (P)	0,470 (P)	0,516 (P)	0,485 (P)	0,575 (P)	0,470 (P)	0,551 (P)	10/10
Test Hammingovy váhy	0,974 (P)	0,574 (P)	0,004 (N)	0,920 (P)	0,083 (P)	0,002 (N)	0,919 (P)	0,339 (P)	0,663 (P)	0,597 (P)	8/10
Frekvenční test	0,503 (P)	0,985 (P)	0,448 (P)	0,429 (P)	0,297 (P)	0,373 (P)	0,690 (P)	0,107 (P)	0,452 (P)	0,859 (P)	10/10
Blokový frekvenční test	0,041 (P)	0,334 (P)	0,901 (P)	0,848 (P)	0,512 (P)	0,134 (P)	0,958 (P)	0,434 (P)	0,946 (P)	0,776 (P)	10/10
Test sekvencí bitů	0,654 (P)	0,608 (P)	0,132 (P)	0,911 (P)	0,546 (P)	0,193 (P)	0,829 (P)	0,931 (P)	0,530 (P)	0,491 (P)	10/10
Test nejdelší sekvence jedniček v bloku	0,845 (P)	0,371 (P)	0,685 (P)	0,267 (P)	0,474 (P)	0,799 (P)	0,580 (P)	0,514 (P)	0,838 (P)	0,086 (P)	10/10
Test diskrétní Fourierovy transformace	0,542 (P)	0,353 (P)	0,173 (P)	1,000 (P)	0,024 (P)	0,117 (P)	0,772 (P)	0,212 (P)	0,014 (P)	0,310 (P)	10/10
Test nepřekrývajících se vzorů	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	10/10
Test sérií	0,904 (P)	0,995 (P)	0,250 (P)	0,425 (P)	0,218 (P)	0,707 (P)	0,289 (P)	0,649 (P)	0,624 (P)	0,966 (P)	10/10
Test přibližné entropie	0,946 (P)	0,983 (P)	0,141 (P)	0,460 (P)	0,212 (P)	0,637 (P)	0,965 (P)	0,561 (P)	0,244 (P)	0,938 (P)	10/10
Test kumulativních součtů	0,740 (P)	0,955 (P)	0,481 (P)	0,701 (P)	0,338 (P)	0,491 (P)	0,750 (P)	0,120 (P)	0,427 (P)	0,826 (P)	10/10
Test binárních matic	0,652 (P)	0,708 (P)	0,293 (P)	0,728 (P)	0,129 (P)	0,464 (P)	0,885 (P)	0,177 (P)	0,226 (P)	0,403 (P)	10/10

Tab. 7.2: Souhrn výsledků všech testovaných generátorů.

Název testu	Python random	Python secrets	LCG	Xorshift	LFSR	QKD
Poker test	10/10	9/10	10/10	8/10	0/10	10/10
Test dvojic bitů	10/10	10/10	10/10	10/10	0/10	10/10
Test mezer	9/10	8/10	10/10	9/10	0/10	9/10
Test bodů zvratu	9/10	10/10	9/10	9/10	10/10	9/10
Test autokorelace	9/10	9/10	9/10	9/10	0/10	10/10
Test Hammingovy váhy	10/10	10/10	10/10	10/10	0/10	8/10
Frekvenční test	10/10	10/10	10/10	10/10	0/10	10/10
Blokový frekvenční test	9/10	10/10	10/10	10/10	0/10	10/10
Test sekvencí bitů	10/10	10/10	10/10	10/10	-	10/10
Test nejdelší sekvence jedniček v bloku	10/10	10/10	10/10	10/10	0/10	10/10
Test diskrétní Fourierovy transformace	10/10	10/10	10/10	10/10	0/10	10/10
Test nepřekrývajících se vzorů	10/10	10/10	10/10	10/10	10/10	10/10
Test sérií	10/10	10/10	10/10	10/10	0/10	10/10
Test přibližné entropie	10/10	10/10	10/10	10/10	0/10	10/10
Test kumulativních součtů	10/10	10/10	10/10	10/10	0/10	10/10
Test binárních matic	10/10	10/10	10/10	10/10	0/10	10/10
Celková úspěšnost generátoru	156/160	156/160	158/160	155/160	20/150	156/160
Úspěšnost (%)	97.50%	97.50%	98.75%	96.88%	13.33%	97.50%

Závěr

Cílem bakalářské práce bylo statisticky porovnat kvalitu dat produkovaných různými generátory náhodných čísel. Dále také nastudovat a popsat problematiku generování náhodných čísel, popsat možnosti jejich statistického testování a vytvořit vlastní aplikaci sloužící k testování generátorů náhodných čísel. Všechny tyto cíle byly v práci naplněny.

Prvních pět kapitol se problematice věnuje převážně teoreticky. V první kapitole byly uvedeny pojmy a využití náhodných čísel, ve druhé způsoby generování a ve třetí teorie testování statistických hypotéz. Čtvrtá kapitola popisuje statistické testy náhodných čísel a kapitola následující detailně rozebírá sadu testů NIST. Tyto dvě kapitoly se zabývají konkrétním popisem testů a slouží jako opora pro implementaci ve vlastní aplikaci.

Kapitola šestá, spadající do praktické části práce, se věnuje popisu vytvořené aplikace pro testování náhodných čísel. Výstupem práce je právě tato desktopová aplikace v jazyce Python, která obsahuje 23 statistických testů náhodnosti a 5 vestavěných generátorů pseudonáhodných čísel. Aplikace bude použita i k výukovým účelům – ke každému obsaženému generátoru a testu je v rámci aplikace dostupný i jeho popis.

V rámci sedmé kapitoly bylo provedeno samotné testování generátorů. Kromě implementovaných generátorů byly otestovány také klíče, které byly vyprodukovány polygonem pro distribuci kvantových klíčů nacházejícím se na Fakultě elektrotechniky a komunikačních technologií VUT v Brně. Do poslední kapitoly je také zahrnuta diskuze o výsledcích testování a jejich interpretace.

Literatura

1. BASSHAM, Lawrence; RUKHIN, Andrew; SOTO, Juan; NECHVATAL, James; SMID, Miles; LEIGH, Stefan; LEVENSON, M; VANGEL, M; HECKERT, Nathanael; BANKS, D. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. Special Publication (NIST SP), National Institute of Standards a Technology, Gaithersburg, MD, 2010. Dostupné také z: `https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762`.
2. STIPČEVIĆ, Mario; KOÇ, Çetin Kaya. True Random Number Generators. In: *Open Problems in Mathematics and Computational Science*. Ed. KOÇ, Çetin Kaya. Cham: Springer International Publishing, 2014, s. 275–315. ISBN 978-3-319-10683-0. Dostupné z DOI: `10.1007/978-3-319-10683-0_12`.
3. HAAHR, Mads. *RANDOM.ORG: Introduction to Randomness and Random Numbers* [`https://www.random.org/randomness`]. 1998–2022. Accessed: 2022-10-24.
4. KATZ, Jonathan; LINDELL, Yehuda. *Introduction to modern cryptography*. Boca Raton: Chapman & Hall/CRC, 2008. ISBN 978-1-58488-551-1.
5. KELSEY, John; SCHNEIER, Bruce; WAGNER, David; HALL, Chris. Cryptanalytic Attacks on Pseudorandom Number Generators. In: VAUDENAY, Serge (ed.). *Fast Software Encryption*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, s. 168–188. ISBN 978-3-540-69710-7.
6. SONMEZ, Meltem; BARKER, Elaine; KELSEY, John; MCKAY, Kerry; BAISH, Mary; BOYLE, Mike. *Recommendation for the Entropy Sources Used for Random Bit Generation*. Special Publication (NIST SP), National Institute of Standards a Technology, Gaithersburg, MD, 2018. Dostupné z DOI: `https://doi.org/10.6028/NIST.SP.800-90b`.
7. FABIAN, František; KLUIBER, Zdeněk. *Metoda Monte Carlo a možnosti jejího uplatnění*. 1. vyd. Praha: Prospektrum, 1998. ISBN 80-7175-058-1.
8. SCHINDLER, Werner. Random Number Generators for Cryptographic Applications. In: *Cryptographic Engineering*. Ed. KOÇ, Çetin Kaya. Boston, MA: Springer US, 2009, s. 5–23. ISBN 978-0-387-71817-0. Dostupné z DOI: `10.1007/978-0-387-71817-0_2`.
9. HEATH, Michael T. *Scientific computing: An introductory survey*. 2. vyd. Society for Industrial a Applied Mathematics, 2018.

10. MENEZES, Alfred J.; OORSCHOT, Paul C.van; VANSTONE, Scott A. *Handbook of applied cryptography*. Vyd. 1. Boca Raton: CRC Press, 1997. ISBN 0-8493-8523-7.

11. MECHALAS, John P. *Intel® Digital Random Number Generator (DRNG) Software Implementation Guide*. 2014. Dostupné také z: `https://www.intel.com/content/www/us/en/developer/articles/guide/intel-digital-random-number-generator-drng-software-implementation-guide.html`.

12. *Quantis QRNG USB*. 2022. Dostupné také z: `https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator/`.

13. KNUTH, Donald Ervin. *The art of computer programming*. 3rd ed. Upper Saddle River, NJ: Addison-Wesley, c1997. ISBN 978-0-201-89683-1.

14. MARSAGLIA, George. Xorshift RNGs. *Journal of Statistical Software*. 2003, roč. 8, č. 14, 1–6. Dostupné z DOI: `10.18637/jss.v008.i14`.

15. PANNETON, François; L'ECUYER, Pierre. On the xorshift random number generators. *ACM Trans. Model. Comput. Simul.* 2005, roč. 15, s. 346–361. Dostupné z DOI: `10.1145/1113316.1113319`.

16. VIGNA, Sebastiano. *It is high time we let go of the Mersenne Twister*. arXiv, 2019. Dostupné z DOI: `10.48550/ARXIV.1910.06437`.

17. MATSUMOTO, Makoto; NISHIMURA, Takuji. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Trans. Model. Comput. Simul.* 1998, roč. 8, č. 1, 3–30. ISSN 1049-3301. Dostupné z DOI: `10.1145/272991.272995`.

18. WIDYNSKI, Bernard. *Squares: A Fast Counter-Based RNG*. arXiv, 2020. Dostupné z DOI: `10.48550/ARXIV.2004.06278`.

19. BLUM, L.; BLUM, M.; SHUB, M. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*. 1986, roč. 15, č. 2, s. 364–383. Dostupné z DOI: `10.1137/0215025`.

20. AUMASSON, Jean-Philippe. On the pseudo-random generator ISAAC. *IACR Cryptology ePrint Archive*. 2006, roč. 2006, s. 438.

21. KOUTKOVÁ, Helena. *Pravděpodobnost a matematická statistika*. Vyd. 1. Brno: Akademické nakladatelství CERM, 2007. ISBN 978-80-7204-527-3.

22. JONES, Eric; OLIPHANT, Travis; PETERSON, Pearu et al. *SciPy: Open source scientific tools for Python*. 2001–. Dostupné také z: `http://www.scipy.org/`.

23. LIKEŠ, Jiří; MACHEK, Josef. *Matematická statistika*. Vyd. 2. 2019. ISBN neuvedeno.

24. ABDEL-REHIM, Wael Mohamed Fawaz; ISMAIL, Ismail Amr; MORSY, Ehab. Testing Randomness: Poker Test with Hands of Three Numbers. *Journal of Computer Science*. 2012, roč. 8, č. 8, s. 1353–1357. Dostupné z DOI: 10.3844/jcssp.2012.1353.1357.

25. ABDEL-REHIM, Wael MF; ISMAIL, Ismail A; MORSY, Ehab. Implementing the classical poker approach for Testing Randomness. *International Journal*. 2014, roč. 4, č. 8.

26. DŘÍMAL, Jiří; TRUNEC, David; BRABLEC, Antonín. *Úvod do metody Monte Carlo*. Brno: Masarykova univerzita, 2006. Dostupné také z: https://www.physics.muni.cz/~trunec/mc.pdf.

27. RITZHAUPT, Albert. *Autocorrelation Random Number Test*. [B.r.]. Dostupné také z: https://aritzhaupt.com/autocorrelation/index.html.

28. MATEUS, Ayana; CAEIRO, Frederico. Comparing several tests of randomness based on the difference of observations. *AIP Conference Proceedings*. 2013, roč. 1558, č. 1, s. 809–812. Dostupné z DOI: 10.1063/1.4825618.

29. KIM, Song-Ju; UMENO, Ken. Revisions to the Spectral Test and the Lempel-Ziv Compression Test in the NIST Statistical Test Suite. In: 2005. Dostupné také z: https://www.researchgate.net/publication/294756057_Revisions_to_the_Spectral_Test_and_the_Lempel-Ziv_Compression_Test_in_the_NIST_Statistical_Test_Suite.

30. RUKHIN, Andrew; SOTO, Juan; NECHVATAL, James; SMID, Miles; BARKER, Elaine; LEIGH, Stefan; LEVENSON, Mark; VANGEL, Mark; BANKS, David; HECKERT, N.; AL., et. *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. 2001. Dostupné také z: https://csrc.nist.gov/publications/detail/sp/800-22/archive/2001-05-15.

31. DRAGOMIR, Ioana. Statistical Assessment of Binary Sequences Generated by Cryptographic Algorithms. *Social Economic Debates*. 2016, roč. 5, č. 2. Dostupné také z: https://ssrn.com/abstract=2890975.

32. WALKER, John. *ENT: Pseudorandom Number Sequence Test Program*. 2008. Dostupné také z: https://www.fourmilab.ch/random/.

33. BROWN, Robert G. *Dieharder: A Random Number Test Suite*. [B.r.]. Dostupné také z: https://webhome.phy.duke.edu/~rgb/General/dieharder.php.

34. L'ECUYER, Pierre; SIMARD, Richard. TestU01: A C Library for Empirical Testing of Random Number Generators. *ACM Trans. Math. Softw.* 2007, roč. 33, č. 4. ISSN 0098-3500. Dostupné z DOI: 10.1145/1268776.1268777.

35. SLEEM, Lama; COUTURIER, Raphael. TestU01 and Practrand: Tools for a randomness evaluation for famous multimedia ciphers. *Multimedia Tools and Applications*. 2020, roč. 79, č. 33-34, s. 24075 –24088. Dostupné také z: https://hal.archives-ouvertes.fr/hal-02993846.

36. ŘÍHA, Zdeněk; SÝS, Marek. *Faster randomness testing*. [B.r.]. Dostupné také z: https://randomness-tests.fi.muni.cz/.

37. SÝS, Marek; ŘÍHA, Zdeněk. Faster Randomness Testing with the NIST Statistical Test Suite. In: CHAKRABORTY, Rajat Subhra; MATYAS, Vashek; SCHAUMONT, Patrick (ed.). *Security, Privacy, and Applied Cryptography Engineering*. Cham: Springer International Publishing, 2014, s. 272–284. ISBN 978-3-319-12060-7.

38. SAARINEN, Markku-Juhani O. SP 800–22 and GM/T 0005–2012 Tests: Clearly Obsolete, Possibly Harmful. In: *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2022, s. 31–37. Dostupné z DOI: 10.1109/EuroSPW55150.2022.00011.

39. RIVERBANK COMPUTING LIMITED. *Pyqt5 reference guide*. 2023. Dostupné také z: https://www.riverbankcomputing.com/static/Docs/PyQt5/.

40. PASQUALINI, Luca. *Nistrng*. [B.r.]. Dostupné také z: https://pypi.org/project/nistrng/.

41. PYTHON SOFTWARE FOUNDATION. *Random - generate pseudo-random numbers*. 2023. Dostupné také z: https://docs.python.org/3/library/random.html#random.random.

42. D'APRANO, Steven. *PEP 506 – Adding A Secrets Module To The Standard Library*. 2015. Dostupné také z: https://peps.python.org/pep-0506/.

43. PYTHON SOFTWARE FOUNDATION. *OS - miscellaneous operating system interfaces*. 2023. Dostupné také z: https://docs.python.org/3/library/os.html#os.urandom.

44. ENTACHER, Karl. A Collection of Selected Pseudorandom Number Generators With Linear Structures. 1999. Dostupné také z: https://www.researchgate.net/publication/2683298_A_Collection_of_Selected_Pseudorandom_Number_Generators_With_Linear_Structures.

Seznam symbolů a zkratek

AES	Advanced Encryption Standard – standard pokročilého šifrování
API	Application Programming Interface – rozhraní pro programování aplikací
BBS	Blum Blum Shub
BSD	Berkeley Software Distribution – distribuce softwaru Berkeley
CNG	Cryptography API: Next Generation – nová generace rozhraní pro kryptografii
CSPRNG	Cryptographically Secure Pseudorandom Number Generator – generátor kryptograficky bezpečných náhodných čísel
DH	Diffie–Hellman
DSA	Digital Signature Algorithm – algoritmus digitálního podpisu
ECDSA	Elliptic Curve Digital Signature Algorithm – protokol digitálního podpisu s využitím eliptických křivek
ERFC	Complementary Error Function – doplňková error funkce
GCD	Greatest Common Divisor – největší společný dělitel
GPL	General Public License – obecná veřejná licence
JSON	JavaScript Object Notation – JavaScriptový objektový zápis
K-S test	Kolmogorovův–Smirnovův test
LCG	Linear Congruential Generator – lineární kongruentní generátor
LCM	Least Common Multiple – nejmenší společný násobek
LFSR	Linear Feedback Shift Register – posuvný registr s lineární zpětnou vazbou
LSB	Least Significant Bit – nejméně významný bit
LZ test	Lempel–Ziv test
MSB	Most Significant Bit – nejvýznamnější bit

NIST	National Institute of Standards and Technology – národní institut standardů a technologie
PCI	Peripheral Component Interconnect – počítačová sběrnice pro připojení periferií
PRNG	Pseudorandom Number Generator – pseudonáhodný generátor náhodných čísel
QKD	Quantum Key Distribution – kvantová distribuce klíče
RBG	Random Bit Generator – generátor náhodných bitů
RNG	Random Number Generator – generátor náhodných čísel
RSA	Rivest–Shamir–Adlemann
SP	Special Publication – speciální publikace
STS	Statistical Test Suite - soubor statistických testů
TRNG	True Random Number Generator – generátor pravých náhodných čísel
USB	Universal Serial Bus – univerzální sériová sběrnice
XML	eXtensible Markup Language – rozšířitelný značkovací jazyk
XOR	exclusive or – exkluzivní disjunkce

Seznam příloh

A Tabulky pro testy NIST

Některé testy pracují s tabulkovými hodnotami π_i, které se z důvodů zachování přehlednosti testů nacházejí v této kapitole příloh.

A.1 Test nejdelší sekvence jedniček v bloku

Velikost bloku testu je volena podle tabulky A.1. Frekvence jsou určeny dle tabulky A.2 v závislosti na velikosti bloku M. Na této velikosti také závisí hodnoty K a N uvedené v tabulce A.3.

Tab. A.1: Test nejdelší sekvence jedniček v bloku: podmínky pro velikost bloku M.

Minimální n	M
128	8
6272	128
750000	10^4

Tab. A.2: Test nejdelší sekvence jedniček v bloku: rozdělení do kategorií podle hodnoty M.

v_i	$M = 8$	$M = 128$	$M = 10^4$
v_0	≤ 1	≤ 4	≤ 10
v_1	2	5	11
v_2	3	6	12
v_3	≥ 4	7	13
v_4		8	14
v_5		≥ 9	15
v_6			≥ 16

Tabulky hodnot pravděpodobností π_i pro test nejdelší sekvence jedniček v bloku jsou rozděleny podle velikosti K a M: tabulka A.4 pro $K = 3$ a $M = 8$, tabulka A.5 pro $K = 5$ a $M = 128$ a A.6 pro $K = 6$ a $M = 1000$.

A.2 Test překrývajících se vzorů

Test počítá s hodnotami π_i z tabulky A.7.

Tab. A.3: Test nejdelší sekvence jedniček v bloku: určení hodnot K a N na základě velikosti bloku M.

M	**K**	**N**
8	3	16
128	5	49
10^4	6	75

Tab. A.4: Test nejdelší sekvence jedniček v bloku: hodnoty pravděpodobností π_i pro $K = 3$ a $M = 8$.

Třídy	Pravděpodobnosti
$v \leq 1$	$\pi_0 = 0,2148$
$v = 2$	$\pi_1 = 0,3672$
$v = 3$	$\pi_2 = 0,2305$
$v \geq 4$	$\pi_3 = 0,1875$

Tab. A.5: Test nejdelší sekvence jedniček v bloku: hodnoty pravděpodobností π_i pro $K = 5$ a $M = 128$.

Třídy	Pravděpodobnosti
$v \leq 4$	$\pi_0 = 0,1174$
$v = 5$	$\pi_1 = 0,2430$
$v = 6$	$\pi_2 = 0,2493$
$v = 7$	$\pi_3 = 0,1752$
$v = 8$	$\pi_4 = 0,1027$
$v \geq 9$	$\pi_5 = 0,1124$

A.3 Maurerův univerzální statistický test

Tabulka očekávaných hodnot A.8 a doporučených velikostí vstupu z tabulky A.9.

A.4 Test lineární složitosti

Tabulka hodnot pravděpodobností π_i pro test LFSR A.10.

Tab. A.6: Test nejdelší sekvence jedniček v bloku: hodnoty pravděpodobností π_i pro $K = 6$ a $M = 1000$.

Třídy	Pravděpodobnosti
$v \leq 10$	$\pi_0 = 0,0882$
$v = 11$	$\pi_1 = 0,2092$
$v = 12$	$\pi_2 = 0,2483$
$v = 13$	$\pi_3 = 0,1933$
$v = 14$	$\pi_4 = 0,1208$
$v = 15$	$\pi_5 = 0,0675$
$v \geq 16$	$\pi_6 = 0,0727$

Tab. A.7: Test překrývajících se vzorů: hodnoty pravděpodobností π_i.

π_i	Hodnota
π_0	$0,364091$
π_1	$0,185659$
π_2	$0,139381$
π_3	$0,100571$
π_4	$0,0704323$
π_5	$0,139865$

A.5 Test náhodných návštěv

V testu náhodných návštěv jsou hodnoty pravděpodobností π_i uvedeny v tabulce A.11.

Tab. A.8: Maurerův univerzální statistický test: očekávané hodnoty L.

L	$\mathbf{L}_{\text{expected}}$	**Rozptyl**
6	5,2177052	2,954
7	6,1962507	3,125
8	7,1836656	3,238
9	8,1764248	3,311
10	9,1723243	3,356
11	10,170032	3,384
12	11,168765	3,401
13	12,168070	3,410
14	13,167693	3,416
15	14,167488	3,419
16	15,167379	3,421

Tab. A.9: Maurerův univerzální statistický test: doporučené velikosti vstupu.

n	**L**	$\mathbf{Q = 10 \cdot 2^L}$
≥ 387840	6	640
≥ 904960	7	1280
≥ 2068480	8	2560
≥ 4654080	9	5120
≥ 10342400	10	10240
≥ 22753280	11	20480
≥ 49643520	12	40960
≥ 107560960	13	81920
≥ 231669760	14	163840
≥ 496435200	15	327680
≥ 1059061760	16	655360

Tab. A.10: Test lineární složitosti: hodnoty pravděpodobností π_i.

π_i	Hodnota
π_0	$0,364091$
π_1	$0,185659$
π_2	$0,139381$

Tab. A.11: Test náhodných návštěv: hodnoty pravděpodobností π_i pro test náhodných návštěv.

	$\pi_0(x)$	$\pi_1(x)$	$\pi_2(x)$	$\pi_3(x)$	$\pi_4(x)$	$\pi_5(x)$
$x = 1$	$0,5000$	$0,2500$	$0,1250$	$0,0625$	$0,0312$	$0,0312$
$x = 2$	$0,7500$	$0,0625$	$0,0469$	$0,352$	$0,0264$	$0,0791$
$x = 3$	$0,8333$	$0,0278$	$0,0231$	$0,0193$	$0,0161$	$0,0804$
$x = 4$	$0,8750$	$0,0156$	$0,0137$	$0,0120$	$0,0105$	$0,0733$
$x = 5$	$0,9000$	$0,0100$	$0,0090$	$0,0081$	$0,0073$	$0,0656$
$x = 6$	$0,9167$	$0,0069$	$0,0064$	$0,0058$	$0,0053$	$0,0058$
$x = 7$	$0,9286$	$0,0051$	$0,0047$	$0,0044$	$0,0041$	$0,0541$

B Výsledky testování generátorů náhodných čísel

V příloze B jsou uvedeny tabulky s kompletními výsledky testování generátorů.

Tab. B.1: Výsledky testů generátoru Python random na sekvencích 100 000 bitů v 10 iteracích.

Název testu	**Iterace testu**										**Úspěšnost**
Poker test	0,570 (P)	0,208 (P)	0,869 (P)	0,065 (P)	0,478 (P)	0,400 (P)	0,652 (P)	0,892 (P)	0,214 (P)	0,718 (P)	10/10
Test dvojic bitů	0,516 (P)	0,020 (P)	0,610 (P)	0,705 (P)	0,718 (P)	0,219 (P)	0,794 (P)	0,854 (P)	0,179 (P)	0,952 (P)	10/10
Test mezer	0,325 (P)	0,003 (N)	0,060 (P)	0,283 (P)	0,029 (P)	0,606 (P)	0,506 (P)	0,999 (P)	0,238 (P)	0,592 (P)	9/10
Test bodů zvratu	0,098 (P)	0,656 (P)	0,626 (P)	0,086 (P)	0,003 (N)	0,949 (P)	0,196 (P)	0,056 (P)	0,396 (P)	0,420 (P)	9/10
Test autokorelace	0,566 (P)	0,433 (P)	0,470 (P)	0,436 (P)	0,425 (P)	0,309 (P)	0,006 (N)	0,575 (P)	0,597 (P)	0,588 (P)	9/10
Test Hammingovy váhy	0,838 (P)	0,731 (P)	0,324 (P)	0,453 (P)	0,100 (P)	0,080 (P)	0,748 (P)	0,335 (P)	0,420 (P)	0,125 (P)	10/10
Frekvenční test	0,277 (P)	0,786 (P)	0,479 (P)	0,617 (P)	0,418 (P)	0,131 (P)	0,695 (P)	0,709 (P)	0,064 (P)	0,995 (P)	10/10
Blokový frekvenční test	0,465 (P)	0,210 (P)	0,556 (P)	0,620 (P)	0,627 (P)	0,406 (P)	0,853 (P)	0,002 (N)	0,419 (P)	0,036 (P)	9/10
Test sekvencí bitů	0,384 (P)	0,103 (P)	0,765 (P)	0,182 (P)	0,302 (P)	0,547 (P)	0,536 (P)	0,561 (P)	0,628 (P)	0,608 (P)	10/10
Test nejdelší sekvence jedniček v bloku	0,876 (P)	0,736 (P)	0,441 (P)	0,460 (P)	0,190 (P)	0,772 (P)	0,797 (P)	0,154 (P)	0,929 (P)	0,461 (P)	10/10
Test diskrétní Fourierovy transformace	0,270 (P)	0,124 (P)	0,052 (P)	0,131 (P)	0,486 (P)	0,353 (P)	0,110 (P)	0,451 (P)	0,400 (P)	0,706 (P)	10/10
Test nepřekrývajících se vzorů	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	10/10
Test sérií	0,295 (P)	0,087 (P)	0,859 (P)	0,248 (P)	0,188 (P)	0,149 (P)	0,832 (P)	0,886 (P)	0,403 (P)	0,508 (P)	10/10
Test přibližné entropie	0,216 (P)	0,060 (P)	0,537 (P)	0,699 (P)	0,096 (P)	0,051 (P)	0,852 (P)	0,751 (P)	0,438 (P)	0,952 (P)	10/10
Test kumulativních součtů	0,374 (P)	0,715 (P)	0,695 (P)	0,760 (P)	0,533 (P)	0,191 (P)	0,668 (P)	0,906 (P)	0,093 (P)	0,483 (P)	10/10
Test binárních matic	0,151 (P)	0,197 (P)	0,182 (P)	0,832 (P)	0,253 (P)	0,077 (P)	0,326 (P)	0,292 (P)	0,687 (P)	0,659 (P)	10/10

Tab. B.2: Výsledky testů generátoru Python secrets na sekvencích 100 000 bitů v 10 iteracích.

Název testu	**Iterace testu**										**Úspěšnost**
Poker test	0,715 (P)	0,451 (P)	0,988 (P)	0,761 (P)	0,720 (P)	0,126 (P)	0,008 (N)	0,243 (P)	0,655 (P)	0,396 (P)	9/10
Test dvojic bitů	0,080 (P)	0,702 (P)	0,971 (P)	0,878 (P)	0,949 (P)	0,113 (P)	0,434 (P)	0,108 (P)	0,266 (P)	0,920 (P)	10/10
Test mezer	0,000 (N)	0,993 (P)	0,003 (N)	0,864 (P)	0,969 (P)	0,403 (P)	0,122 (P)	0,541 (P)	0,127 (P)	0,192 (P)	8/10
Test bodů zvratu	0,019 (P)	0,196 (P)	0,094 (P)	0,299 (P)	0,525 (P)	0,319 (P)	0,010 (P)	0,203 (P)	0,289 (P)	0,219 (P)	10/10
Test autokorelace	0,439 (P)	0,399 (P)	0,485 (P)	0,344 (P)	0,489 (P)	0,707 (P)	0,006 (N)	0,516 (P)	0,551 (P)	0,619 (P)	9/10
Test Hammingovy váhy	0,110 (P)	0,259 (P)	0,876 (P)	0,077 (P)	0,490 (P)	0,191 (P)	0,526 (P)	0,133 (P)	0,948 (P)	0,132 (P)	10/10
Frekvenční test	0,452 (P)	0,415 (P)	0,820 (P)	0,985 (P)	0,980 (P)	0,096 (P)	0,199 (P)	0,051 (P)	0,810 (P)	0,820 (P)	10/10
Blokový frekvenční test	0,514 (P)	0,208 (P)	0,112 (P)	0,462 (P)	0,016 (P)	0,148 (P)	0,035 (P)	0,406 (P)	0,485 (P)	0,767 (P)	10/10
Test sekvencí bitů	0,070 (P)	0,555 (P)	0,955 (P)	0,282 (P)	0,437 (P)	0,646 (P)	0,691 (P)	0,819 (P)	0,266 (P)	0,544 (P)	10/10
Test nejdelší sekvence jedniček v bloku	0,476 (P)	0,247 (P)	0,240 (P)	0,137 (P)	0,085 (P)	0,734 (P)	0,605 (P)	0,244 (P)	0,806 (P)	0,182 (P)	10/10
Test diskrétní Fourierovy transformace	0,562 (P)	0,283 (P)	0,310 (P)	0,015 (P)	0,258 (P)	0,025 (P)	0,246 (P)	0,258 (P)	0,324 (P)	0,931 (P)	10/10
Test nepřekrývajících se vzorů	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	10/10
Test sérií	0,264 (P)	0,785 (P)	0,241 (P)	0,807 (P)	0,743 (P)	0,164 (P)	0,206 (P)	0,521 (P)	0,669 (P)	0,639 (P)	10/10
Test přibližné entropie	0,323 (P)	0,696 (P)	0,953 (P)	0,765 (P)	0,796 (P)	0,494 (P)	0,166 (P)	0,376 (P)	0,450 (P)	0,853 (P)	10/10
Test kumulativních součtů	0,569 (P)	0,306 (P)	0,945 (P)	0,968 (P)	0,856 (P)	0,123 (P)	0,138 (P)	0,084 (P)	0,883 (P)	0,569 (P)	10/10
Test binárních matic	0,101 (P)	0,240 (P)	0,677 (P)	0,128 (P)	0,885 (P)	0,359 (P)	0,277 (P)	0,588 (P)	0,832 (P)	0,253 (P)	10/10

Tab. B.3: Výsledky testů generátoru LCG na sekvencích 100 000 bitů v 10 iteracích.

Název testu	**Iterace testu**										**Úspěšnost**
Poker test	0,394 (P)	0,092 (P)	0,072 (P)	0,686 (P)	0,514 (P)	0,026 (P)	0,374 (P)	0,461 (P)	0,258 (P)	0,727 (P)	10/10
Test dvojic bitů	0,409 (P)	0,252 (P)	0,020 (P)	0,827 (P)	0,793 (P)	0,055 (P)	0,587 (P)	0,300 (P)	0,045 (P)	0,080 (P)	10/10
Test mezer	0,556 (P)	0,054 (P)	0,243 (P)	0,325 (P)	0,363 (P)	0,163 (P)	0,325 (P)	0,850 (P)	0,246 (P)	0,632 (P)	10/10
Test bodů zvratu	0,553 (P)	0,362 (P)	0,966 (P)	0,023 (P)	0,611 (P)	0,013 (P)	0,641 (P)	0,005 (N)	0,497 (P)	0,056 (P)	9/10
Test autokorelace	0,669 (P)	0,527 (P)	0,458 (P)	0,608 (P)	0,527 (P)	0,550 (P)	0,405 (P)	0,348 (P)	0,645 (P)	0,006 (N)	9/10
Test Hammingovy váhy	0,197 (P)	0,052 (P)	0,045 (P)	0,763 (P)	0,845 (P)	0,138 (P)	0,178 (P)	0,465 (P)	0,284 (P)	0,930 (P)	10/10
Frekvenční test	0,184 (P)	0,103 (P)	0,117 (P)	0,995 (P)	0,569 (P)	0,044 (P)	0,479 (P)	0,274 (P)	0,051 (P)	0,820 (P)	10/10
Blokový frekvenční test	0,987 (P)	0,730 (P)	0,296 (P)	0,914 (P)	0,459 (P)	0,497 (P)	0,702 (P)	0,610 (P)	0,197 (P)	0,469 (P)	10/10
Test sekvencí bitů	0,849 (P)	0,908 (P)	0,014 (P)	0,737 (P)	0,129 (P)	0,201 (P)	0,419 (P)	0,428 (P)	0,677 (P)	0,010 (P)	10/10
Test nejdelší sekvence jedniček v bloku	0,755 (P)	0,261 (P)	0,891 (P)	0,058 (P)	0,423 (P)	0,426 (P)	0,330 (P)	0,738 (P)	0,167 (P)	0,221 (P)	10/10
Test diskrétní Fourierovy transformace	0,147 (P)	0,885 (P)	0,048 (P)	0,728 (P)	0,601 (P)	0,486 (P)	0,816 (P)	0,283 (P)	0,581 (P)	0,542 (P)	10/10
Test nepřekrývajících se vzorů	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	10/10
Test sérií	0,572 (P)	0,294 (P)	0,212 (P)	0,774 (P)	0,189 (P)	0,121 (P)	0,192 (P)	0,698 (P)	0,346 (P)	0,189 (P)	10/10
Test přibližné entropie	0,294 (P)	0,258 (P)	0,044 (P)	0,799 (P)	0,526 (P)	0,044 (P)	0,482 (P)	0,299 (P)	0,324 (P)	0,063 (P)	10/10
Test kumulativních součtů	0,269 (P)	0,179 (P)	0,122 (P)	0,881 (P)	0,590 (P)	0,044 (P)	0,606 (P)	0,364 (P)	0,058 (P)	0,447 (P)	10/10
Test binárních matic	0,626 (P)	0,898 (P)	0,381 (P)	0,023 (P)	0,957 (P)	0,728 (P)	0,346 (P)	0,708 (P)	0,608 (P)	0,824 (P)	10/10

Tab. B.4: Výsledky testů generátoru Xorshift na sekvencích 100 000 bitů v 10 iteracích.

Název testu	**Iterace testu**										**Úspěšnost**
Poker test	0,330 (P)	0,408 (P)	0,684 (P)	0,320 (P)	0,657 (P)	0,618 (P)	0,008 (N)	0,685 (P)	0,761 (P)	0,004 (N)	8/10
Test dvojic bitů	0,824 (P)	0,955 (P)	0,565 (P)	0,921 (P)	0,671 (P)	0,947 (P)	0,256 (P)	0,672 (P)	0,485 (P)	0,497 (P)	10/10
Test mezer	0,085 (P)	0,102 (P)	0,997 (P)	0,000 (N)	0,983 (P)	0,929 (P)	0,632 (P)	0,780 (P)	0,149 (P)	0,203 (P)	9/10
Test bodů zvratu	0,687 (P)	1,000 (P)	0,279 (P)	0,001 (N)	0,553 (P)	0,062 (P)	0,373 (P)	0,384 (P)	0,219 (P)	0,155 (P)	9/10
Test autokorelace	0,467 (P)	0,009 (N)	0,440 (P)	0,433 (P)	0,415 (P)	0,588 (P)	0,464 (P)	0,386 (P)	0,420 (P)	0,647 (P)	9/10
Test Hammingovy váhy	0,182 (P)	0,848 (P)	0,196 (P)	0,957 (P)	0,219 (P)	0,075 (P)	0,735 (P)	0,895 (P)	0,645 (P)	0,408 (P)	10/10
Frekvenční test	0,640 (P)	0,990 (P)	0,649 (P)	0,704 (P)	0,649 (P)	0,795 (P)	0,519 (P)	0,631 (P)	0,297 (P)	0,306 (P)	10/10
Blokový frekvenční test	0,867 (P)	0,902 (P)	0,501 (P)	0,286 (P)	0,051 (P)	0,382 (P)	0,290 (P)	0,977 (P)	1,000 (P)	0,351 (P)	10/10
Test sekvencí bitů	0,056 (P)	0,491 (P)	0,532 (P)	0,478 (P)	0,586 (P)	0,561 (P)	0,164 (P)	0,658 (P)	0,329 (P)	0,932 (P)	10/10
Test nejdelší sekvence jedniček v bloku	0,550 (P)	0,703 (P)	0,840 (P)	0,498 (P)	0,376 (P)	0,984 (P)	0,416 (P)	0,104 (P)	0,711 (P)	0,494 (P)	10/10
Test diskrétní Fourierovy transformace	0,954 (P)	0,794 (P)	0,908 (P)	0,581 (P)	0,816 (P)	0,642 (P)	0,642 (P)	0,839 (P)	0,908 (P)	0,368 (P)	10/10
Test nepřekrývajících se vzorů	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	10/10
Test sérií	0,693 (P)	0,170 (P)	0,263 (P)	0,860 (P)	0,360 (P)	0,591 (P)	0,029 (P)	0,997 (P)	0,926 (P)	0,671 (P)	10/10
Test přibližné entropie	0,331 (P)	0,359 (P)	0,229 (P)	0,772 (P)	0,940 (P)	0,294 (P)	0,027 (P)	0,926 (P)	0,709 (P)	0,711 (P)	10/10
Test kumulativních součtů	0,761 (P)	0,674 (P)	0,630 (P)	0,762 (P)	0,899 (P)	0,786 (P)	0,699 (P)	0,597 (P)	0,389 (P)	0,242 (P)	10/10
Test binárních matic	0,270 (P)	0,253 (P)	0,177 (P)	0,430 (P)	0,786 (P)	0,357 (P)	0,053 (P)	0,400 (P)	0,608 (P)	0,202 (P)	10/10

Tab. B.5: Výsledky testů generátoru LFSR na sekvencích 100 000 bitů v 10 iteracích.

Název testu	**Iterace testu**										**Úspěšnost**
Poker test	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test dvojic bitů	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test mezer	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test bodů zvratu	1,000 (P)	1,000 (P)	1,000 (P)	1,000 (P)	0,983 (P)	0,983 (P)	0,983 (P)	1,000 (P)	1,000 (P)	0,983 (P)	10/10
Test autokorelace	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test Hammingovy váhy	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Frekvenční test	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Blokový frekvenční test	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test sekvencí bitů	-	-	-	-	-	-	-	-	-	-	-
Test nejdelší sekvence jedniček v bloku	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test diskrétní Fourierovy transformace	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test nepřekrývajících se vzorů	1,000 (P)	0,341 (P)	1,000 (P)	1,000 (P)	1,000 (P)	0,272 (P)	0,167 (P)	1,000 (P)	1,000 (P)	1,000 (P)	10/10
Test sérií	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test přibližné entropie	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test kumulativních součtů	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10
Test binárních matic	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0,000 (N)	0/10

C Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `generated_data` ..... složka obsahující programem vygenerované sekvence
  - `gui` ..... zdrojové kódy grafického uživatelského prostředí aplikace
    - `eligible_tests.py`
    - `generate_numbers.py`
    - `choose_file.py`
    - `main_window.py`
    - `results.py`
    - `logos` ..... logo aplikace
      - `dice.png`
    - `ui` ..... soubory `.ui` definující vzhled aplikace
      - `EligibleTests.ui`
      - `GenerateNumbers.ui`
      - `ChooseFile.ui`
      - `MainWindow.ui`
      - `Results.ui`
  - `source` .. zdrojové kódy aplikace obsahující testy, generátory, další funkce aplikace
    - `tests` ..... zdrojové kódy vlastních testů
      - `graphical_test.py`
      - `test_autocorrelation.py`
      - `test_gap.py`
      - `test_hamming_weight.py`
      - `test_lempel_ziv_compression.py`
      - `test_nist_serial_mod.py`
      - `test_poker.py`
      - `test_turning_point.py`
      - `test_two_bit.py`
    - `create_battery_of_tests.py`
    - `file_reader.py`
    - `generators.py`
    - `test_data.py`
  - `test_results` ..... složka pro ukládání výsledků testování
  - `app.py`
  - `BP.pdf` ..... text bakalářské práce
  - `README.md` ..... postup spuštění aplikace
  - `requirements.txt` ..... požadované knihovny pro spuštění aplikace