Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

ŠTÁTNY PEDAGOGICKÝ ÚSTAV
NATIONAL INSTITUTE FOR EDUCATION

Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Kapitoly z informatiky 3

Predmet: Kapitoly z informatiky

Línia: Vlastný odborový kontext informatiky a informatickej výchovy

Operačný program VZDELÁVANIE

EURÓPSKA ÚNIA
Európsky sociálny fond

Európska únia
Európsky sociálny fond

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov ES

Kapitoly z informatiky 3

Identifikácia modulu

Aktivita projektu:	1.2 Vzdelávanie nekvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Vlastný odborový kontext informatiky a informatickej výchovy
Predmet:	Kapitoly z informatiky

Garant predmetu:

Doc. RNDr. Gabriela Andrejková, CSc.,
ÚI PF UPJŠ, Košice
Gabriela.Andrejkova@upjs.sk

Autori:

RNDr. Michal Winczer, PhD.
KZVI FMFI UK, Bratislava
winczer@fmph.uniba.sk

RNDr. František Galčík, PhD.
ÚINF PF UPJŠ, Košice
frantisek.galcik@upjs.sk

RNDr. Michal Forišek, PhD.
KI FMFI UK, Bratislava
misof@ksp.sk

Zaradenie modulu

Modul *Kapitoly z informatiky 3* patrí do série troch modulov *Kapitoly z informatiky*, ktoré majú za úlohu priniesť frekventantom základy informatiky ako vedy. Modul predpokladá absolvovanie všetkých programátorských modulov *Programovanie 1-9* a tiež modulov *Matematika pre učiteľov informatiky 1-3, Algoritmy a údajové štruktúry 1, 2* a *Počítačové systémy 1-5*.

Modul je rozdelený na dve témy: kryptografiu a pravdepodobnostné algoritmy.

Abstrakt modulu

V module zoznámime čitateľov s dvoma oblasťami informatiky, v ktorých teoretické výsledky majú početné aplikácie v praxi. Prvou oblasťou je kryptografia. Podáme prehľad toho, ako sa kryptografia vyvíjala a zároveň si predstavíme niektoré aplikácie modernej kryptografie v praxi. Druhou oblasťou sú pravdepodobnostné algoritmy, ktoré sú pekným príkladom toho, ako s využitím náhody skonštruovať efektívne algoritmy no na úkor toho, že budeme musieť akceptovať to, že niekedy dostaneme nesprávny výsledok. Použitím istých techník konštrukcie algoritmov opierajúcich sa o teóriu pravdepodobnosti však dokážeme pravdepodobnosť nesprávneho výsledku v niektorých prípadoch spraviť ľubovoľne malou. Predstavíme si výpočty s použitím pravdepodobnosti, Las Vegas i Monte Carlo pravdepodobnostné algoritmy. Na záver si ukážeme ako sa generujú (pseudo)náhodné čísla v počítačoch.

Obsah

Úvod

Milá čitateľka, milý čitateľ, v učebných materiáloch *Kapitoly z informatiky 1* ste sa zoznámili s pojmom efektívny algoritmus, priblížili ste si oblasti teoretickej informatiky: analýza algoritmov a výpočtová zložitosť. *Materiál Kapitoly z informatiky 2* predstavil ďalšiu oblasť teoretickej informatiky: vypočítateľnosť, konkrétne dva výpočtové modely – konečný automat a Turingov stroj, na ktorých sme si ukazovali, čo je to vôbec výpočet a ako charakterizovať, čo konkrétny model vôbec „dokáže“ vypočítať. Zaviedol sa tu aj pojem nedeterminizmus a ukázali sme si, čo znamená v prípade spomínaných dvoch výpočtových modelov.

V prekladanom treťom module *Kapitoly z informatiky 3* sa venujeme dvom oblastiam: *kryptografii* a *pravdepodobnostným algoritmom*.

Neznamená to, že informatika už nemá iné oblasti, ale vzhľadom na priestor, ktorý máme k dispozícii, sme sa pokúsili vybrať reprezentatívne také, ktoré sú prakticky dôležité a často používané, hoci si túto skutočnosť vôbec nemusíme uvedomovať. Druhý dôvod na ich výber bol, že ich považujeme za potenciálne atraktívne aj pre žiakov či študentov a navyše aj za realizovateľné vo Vašej výučbe.

Pozývame vás na výlet do krajín šifier a náhod, spoliehame sa na Vašu kreatívnosť a schopnosti improvizovať.

Cieľ modulu

Cieľom modulu je predstaviť základné princípy šifrovania, úlohu informatiky pri definovaní pojmu bezpečná šifra a ukázať základné myšlienky šifrovania, ktoré sú základom bezpečnej komunikácie na internete.

Druhý cieľ modulu je priblížiť poslucháčom náhodu a jej využitie pri riešení niektorých informatických problémov. Chceme tiež upozorniť na to, že náhoda je silný pomocník informatikov a pomáha prakticky riešiť úlohy, ktoré inak nie sme schopní riešiť.

Vstupné vedomosti

Požadované prerekvizity

Modul predpokladá absolvovanie všetkých programátorských modulov *Programovanie 1—9* a tiež modulov *Matematika pre učiteľov informatiky 1—3*, *Algoritmy a údajové štruktúry 1*, *2* a *Počítačové systémy 1—5*.

Predpokladané vstupné vedomosti, skúsenosti a zručnosti

Predpokladáme, že účastník vzdelávania:

- vie napísať jednoduchý program a ovláda údajové štruktúry prebrané v moduloch *Programovanie 1* až *9*,
- má pomerne detailnú predstavu o fungovaní počítača,
- ovláda matematiku na bežnej stredoškolskej úrovni.

Preverenie vstupných vedomostí

Účastník vzdelávania vie použiť na šifrovanie a dešifrovanie textu viaceré jednoduché šifry. Pozná slabé a silné stránky jednoduchých šifier a niektoré prístupy ako ich prelomiť.

Poznámka

Všetky projekty spomínané v texte sú dostupné v systéme Moodle projektu DVUi medzi súbormi k tomuto modulu.

Kryptografia

Veľmi zjednodušene môžeme povedať, že šifrovanie je zapisovanie textu v takej podobe, aby mu nepovolaný čitateľ nemal šancu porozumieť. Najčastejšie sa šifrovanie v jeho rôznych podobách používa pri komunikácii: jeden človek má informáciu, ktorú potrebuje poslať inému; nechce však, aby si ju mohol cestou prečítať ktokoľvek ďalší.

Slovo kryptografia pochádza z gréckeho *kryptos* (skrytý, utajený) a *grafein* (písať). Niekedy sa ako približné synonymum používa tiež slovo kryptológia.

Práve šifrovanie patrí k témam, ktorými sa budeme zaoberať v nasledujúcom texte. Presnejšie, jeho témou bude **kryptografia**: vedná oblasť zaoberajúca sa ukrývaním obsahu textu pred nepovolanými osobami.

S kryptografiou súvisí niekoľko ďalších vedných oblastí. V prvom rade ide o opačnú stranu mince, **kryptoanalýzu**: vednú oblasť zaoberajúcu sa štúdiom metód, ktoré môže použiť útočník pri snahe získať zo zašifrovaného textu ukryté informácie. Taktiež pár vetami spomenieme **steganografiu**, ktorá sa zaoberá utajovaním samotného prenosu správ. Tieto a všetky súvisiace oblasti niektorí autori označujú súhrnným pojmom **kryptológia**. (Pri používaní tohto pojmu však nie je úplný konsenzus, viaceré zdroje používajú pojmy kryptografia a kryptológia ako približné synonymá.)

Text je členený do niekoľkých samostatných častí: Na úvod uvedieme stručnú históriu kryptografie a pri jej výklade sa oboznámime so všetkými potrebnými odbornými pojmami. V druhej časti si v nutne zjednodušenej podobe vysvetlíme princípy fungovania kryptografických algoritmov v modernej, počítačovej dobe. V záverečnej tretej časti sa podrobnejšie zaoberáme praktickými aplikáciami výsledkov kryptografie, na ktoré používateľ počítača narazí v bežnom živote. Ide napríklad o elektronické podpisy a zabezpečené web stránky.

Kryptografia v historických dobách

Antika a doby ešte staršie

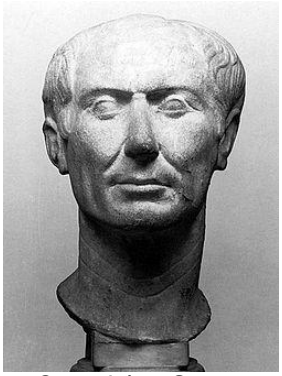
Prvé zachované snahy o šifrovanie textu pochádzajú zo starého Egypta a z Mezopotámie. Zväčša išlo o jednoduchú zámenu symbolov textu, tzv. **substitúciu**: každý symbol pôvodného, tzv. **otvoreného textu** je konzistentne nahradený iným symbolom, čím vzniká **šifrový text**, nečitateľný pre nezasväteného. Napríklad sa zachovali prípady, kedy „pisár“ použil pozmenené formy hieroglyfov – je však možné, že pri spomínaných nápisoch na hrobkách nešlo ani tak o snahu ukryť text pred čitateľom, ako o dramatický efekt.

Treba si však uvedomiť, že nie len v tejto dobe, ale ešte aj dlho potom bolo už samotné napísanie textu akousi formou šifrovania – drvivá väčšina ľudí totiž nevedela čítať, a tak bol obsah správy pred nimi ukrytý. Často krát samotný posol prenášajúci správu nevedel čítať a významu správy nerozumel.

Jednoduché substitučné šifry ostávajú v obľube ešte dlhé storočia. Známym príkladom je hebrejská šifra atbaš, ktorej názov popisuje samotný postup šifrovania: prvé písmeno abecedy sa nahradí posledným, druhé predposledným, a tak ďalej. Pri použití dnešnej anglickej abecedy by sme namiesto každého `A` písali `Z`, namiesto každého `B` by bolo `Y`, namiesto `C` by sme použili `W` a tak ďalej. Napríklad zo slova `ZABA` dostali zašifrované slovo `AZYZ`. (Za povšimnutie stojí, že na dešifrovanie sa pri tejto šifre použije ten istý postup ako na šifrovanie.)

Úloha 1.

Šifrou atbaš sme zašifrovali krátky text. Tu je výsledok: `KLPOZW QV FPIBGB KLW HPZOLF`. Viete zistiť znenie zašifrovanej správy? (Pomôcka: Použitá je anglická 26-znaková abeceda – tá, ktorú nájdete na klávesnici.)



Gaius Julius Caesar
(Obrázok je prevzatý z wikipedia.org.)

Asi najznámejšou z týchto jednoduchých šifier je Cézarova šifra, ktorú používal Gaius Július Cézar v správach posielaných svojim vojvodcom: každé písmeno správy posunul v abecede o tri miesta. (Samotný Cézar aj jeho nasledovníci občas používali aj iné posuny ako o tri, v súčasnosti sa preto pojem „Cézarova šifra“ bežne používa aj v prenesenom význame na označenie ľubovoľného posunu abecedy.)

V časoch antického Grécka sa prvýkrát objavuje použitie mechanickej šifrovacej pomôcky: Išlo o drevený kolík, v gréčtine nazývaný skytalé (obrázok 1.1). Na ten sa navinul prúžok pergamenu a po riadkoch sa naň napísala správa. Po rozvinutí zostala na pergamene nezmyselná postupnosť písmen. Príjemca potom zobral rovnako hrubý kolík a znova naň pergamen navinul, čím sa písmená správy správne zarovnali do riadkov.

Obrázok 1.1: Čítanie dešifrovanej správy pomocou kolíka skytalé. (Obrázok je prevzatý z wikipedia.org.)

Stredoveká Arábia a Európa

Keďže v Európe po konci antiky sa rozvoj vied na dlhé storočia spomaľuje, ďalší historický medzník musíme hľadať inde. Do popredia sa presúva arabská civilizácia.

Spolu s rozvojom rôznych metód šifrovania samozrejme prichádza aj k snahe nepovolaných osôb šifrovaný text rozlúštiť. Výsledkom týchto snáh je dnes vedná oblasť nazývaná **kryptoanalýza**. Jednou z prvých písomných zmienok je text z deviateho storočia nášho letopočtu, ktorého autorom bol Al-Kindí. V tomto texte popisuje viacero metód lúštenia šifrových textov. Okrem iného ide o prvú písomnú zmienku o metóde nazývanej **frekvenčná analýza**: Útočník (teda nepriateľ, ktorý odchytil zašifrovanú správu a snaží sa ju dešifrovať) si spočíta, ktoré znaky sa v správe ako často vyskytujú, a potom pri lúštení použije predpoklad, že často sa vyskytujúce znaky zodpovedajú najbežnejším písmenám.

Úloha 2.	Našli ste papierik s textom `VCVQ URTCXC UC FC NCJMQ QFUKHTQXCV`. Ide o šifru podobnú Cézarovej, ale odosielateľ použil iný posun v abecede, nie o tri znaky. Viete určiť, aký posun si zvolil, a tento text rozšifrovať? (Opäť je použitá 26-písmenová abeceda z klávesnice. Zašifrovaný text je v slovenčine. Napovieme, že najčastejšie písmeno v slovenskom texte je A. Aké je najčastejšie písmeno v šifrovom texte? A aký posun by tomu zodpovedal?)
Úloha	Predstavte si, že ste v starom Grécku. Dostal sa vám do rúk prúžok pergamenu s napísanou správou, nemáte však správny kolík – skytalé, ani neviete, aký je hrubý. Vedeli by ste sa aj tak dostať k tajnej správe? Ako by ste pri tom postupovali?

Na rozlúštenie Cézarovej šifry ešte frekvenčnú analýzu nepotrebujeme. Možných posunov v abecede je len pár, stačí použiť **hrubú silu**, vyskúšať ich všetky a nájsť ten, ktorý vedie k zmysluplnému textu. Existujú však mnohé zložitejšie šifry, pri ktorých lúštení je frekvenčná analýza cenným nástrojom.

Zjavným vylepšením Cézarovej šifry je **všeobecná substitučná šifra**: každé písmeno bolo konzistentne nahradené iným symbolom, napríklad namiesto A srdiečko, namiesto B trojuholník, a tak ďalej. Samozrejme, nové symboly mohli byť opäť písmená, len v inom poradí – autor správy mohol namiesto každého A písať C, namiesto každého B písať W, namiesto každého C písať Q atď.

Substitučná šifra sa dá ľahko rozlúštiť práve použitím frekvenčnej analýzy. Kvôli svojej jednoduchosti sú však substitučné šifry obľúbené dodnes, často sa používajú napríklad pri rôznych hrách. Stretneme sa s nimi aj v krásnej literatúre, napríklad v poviedke *Zlatý chrobák* od Edgara Allana Poea (1843) a v poviedke *Tancujúce figúrky* od Sira Arthura Conana Doyla (1903).

obrázok 1.2: Ukážka zašifrovaného textu, ktorý rieši Sherlock Holmes v poviedke A. C. Doyla. (Obrázok je prevzatý z wikipedia.org.)

Európa v období renesancie

Príchod renesancie do Európy so sebou nesie aj výrazný rozvoj prírodných vied, vrátane kryptografie. V neprestajne sa odohrávajúcich bojoch o moc sa často stáva, že práve šifrovanie poskytne jednej strane potrebnú výhodu. Prichádza teda k rozvoju rôznych druhov čoraz dokonalejších šifier.

Obľube sa vďaka svojej jednoduchosti stále teší substitučná šifra. Tú však už mnohí vedia vďaka frekvenčnej analýze riešiť, a tak je nutné ju rôznymi spôsobmi vylepšovať. V priebehu desaťročí vznikajú rôzne jej variácie, ktoré sa riešia ťažšie. Populárne je používať **nomenklátory**: pre najpoužívanejšie slová (ako napríklad „kráľ“, „pápež“, „vražda“, či rôzne mená) si komunikujúce strany dohodnú „prekladový slovník“. Napríklad namiesto „kráľ“ sa bude písať „XY“. Iní autori používajú **klamače**: do šifrového textu vkladajú znaky navyše, ktoré nemajú žiaden význam – príjemca ich pri dešifrovaní jednoducho preskočí.

Treťou obľúbenou metódou sú **homofóny**: Aby autor sťažil dešifrovanie pomocou frekvenčnej analýzy, pre najčastejšie písmená abecedy bude používať viacero rôznych znakov. Obľúbené je napríklad šifrovanie, kde otvorený text môže obsahovať len písmená, ale šifrový text môže obsahovať aj cifry. V takejto šifre by autor mohol napríklad niektoré E nahradiť znakom X a ostatné E znakom 7. Aj znaku X, aj znaku 7 bude vo výslednej šifre menej, a vďaka tomu je ťažšie spoznať, že práve tieto dva znaky predstavujú samohlásku E.

Význam používania dobrých šifier nemožno podceňovať – často na tom záviseli ľudské životy. A to nie len priamo pri vojenských konfliktoch. V roku 1587 bola popravená Mária Stuartová, škótska kráľovná, za účasť na Babingtonovom sprisahaní – snahe katolíkov o zavraždenie anglickej kráľovnej Alžbety I. Dôkazom boli jej šifrované listy, ktoré si posielala práve s Babingtonom. Tieto listy sa ale cestou dostali do rúk tzv. čiernej komory zriadenej sirom Francisom Walsinghamom a jeden z jej členov, Thomas Philips, dokázal šifru Márie Stuartovej rozlúštiť. Išlo práve o substitučnú šifru s nomenklátorom.

V šestnástom storočí už prichádzajú k slovu aj iné metódy šifrovania. Giovanni Battista Bellaso v roku 1553 prvýkrát popisuje šifru používajúcu **periodické** heslo. V neskorších rokoch sa objav tejto šifry omylom pripíše inému kryptografovi: Blaisovi Vigenèrovi, podľa neho je dnes táto šifra známa ako **Vigenèrova šifra**.

Princíp je opäť veľmi jednoduchý – dostatočne jednoduchý na to, aby sa šifrovať

dalo len pomocou pera a papiera. Odosielateľ a adresát sa dohodnú na spoločnom hesle, napríklad „JAHODA“. Toto heslo, presnejšie spoločnú tajnú informáciu zdieľanú odosielateľom a adresátom, v kryptografii označujeme tiež pojmom **kľúč**. Šifrovanie teraz prebehne nasledovne: Odosielateľ si napíše text, ktorý chce poslať. Pod každé jeho písmeno napíše jedno písmeno hesla. Celé to teda môže vyzerať napríklad nasledovne:

```
STRETNEME SA O POLNOCI
JAHODAJAH OD A JAHODAJ
```

V každom stĺpci teraz máme dve písmená. Tieto „sčítame“ (ako keby to boli čísla, teda napríklad v anglickej abecede by mohlo byť A=1, B=2, ..., Z=26), a tým dostaneme jedno písmeno šifrového textu.

Príklad: S=19, J=10, teda S+J=19+10=29. Keďže 26=Z, 27 je opäť A, 28 je B a 29 je C. Dostávame teda, že vo vyššie uvedenom príklade by prvým písmenom šifrového textu bolo C.

Ďalej dostávame T+A=20+1=21=U, R+H=18+8=26=Z, E+O=5+15=20=T, atď.

Výsledný šifrový text: CUZTXOONM HE P ZPTCSDS. Všimnite si, že na rozdiel od substitučnej šifry sa tu môže stať, že rôzne písmená otvoreného textu sa zašifrujú na rovnaké písmená. Pekne to vidieť na dvoch po sebe idúcich O v prvom slove šifrového textu.

Úloha 4.

Heslom PES sme zašifrovali jedno slovenské slovo. Dostali sme takto šifrový text HJOEQNSNT. Viete ho dešifrovať?

Vigenèrova šifra pôsobila natoľko neprelomiteľne, že si vyslúžila pomenovanie „le chiffre indéchiffrable“, nevylúštiteľná šifra. Opak je však pravdou, jej rozlúštenie je len o trošku ťažšie ako rozlúštenie Cézarovej šifry. Napriek tomu je efektívny postup jej lúštenia prvýkrát verejne publikovaný až v roku 1863, teda vyše tristo rokov po jej objave. Vtedy Friedrich Kasiski prvýkrát verejne popisuje univerzálnu metódu na dešifrovanie bez znalosti hesla. (Zachovalo sa však viacero dôkazov o tom, že mnohí učenci podobné postupy poznali omnoho skôr. Zo zjavných dôvodov si ich však všetci ponechali pre seba.)

Myšlienka tohto útoku je geniálne jednoduchá: Predstavme si, že sme uhádli dĺžku kľúča. (V praxi sa tou dobou ako kľúč používalo slovo, prípadne krátka fráza, takže možných dĺžok kľúča bolo málo a útočníkovi nič nebránilo vyskúšať ich postupne všetky.) Napríklad sme zachytili vyššie uvedený šifrový text CUZTXOONM HE P ZPTCSDS a vieme, že kľúč má dĺžku 6. To, čo teraz môžeme spraviť, je rozdeliť správu na šestice písmen a tie napísať pod seba:

```
CUZTXO
ONMHEP
ZPTCSD
S
```

No a teraz vieme, že v každom stĺpci boli všetky písmená zašifrované tým istým písmenom hesla. Inými slovami, v každom stĺpci bola použitá jedna z 26 možných „Cézarovych“ šifier, teda posunov v abecede. Ak teda máme dostatočne dlhú správu, ľahko v každom stĺpci (napríklad frekvenčnou analýzou) nájdeme ten správny posun, a potom už len dešifrovanú správu po riadkoch prečítame. (V praxi často stačilo správne odhadnúť niekoľko po sebe idúcich znakov hesla, zvyšok sa už ľahko doriešil ručne.)

Steganografia

Jednou samostatnou oblasťou kryptológie je steganografia, veda o tom, ako informácie ukrývať. Oproti klasickému šifrovaniu je tu jeden veľmi podstatný rozdiel: Pri posielaní šifrovanej správy je nepriateľovi jasné, že autor a adresát spolu komunikujú, nevie však zistiť obsah tejto komunikácie. Pri steganografii je hlavným cieľom utajiť, že k akejkoľvek komunikácii došlo. Ľudovo môžeme povedať, že sa snažíme prenášané informácie ukryť na miesto, kde by ich nik nehľadal.

Prvé pokusy ukrývania správ, ktoré môžeme zahrnúť do oblasti steganografie, sa datujú do antických dôb, napríklad sa dochovali prípady, kedy odosielateľ napísal text na drevenú dosku, tú potom pokryl voskom (čím správu ukryl) a do vosku napísal inú, nevinnú správu. Do steganografie tiež patria napríklad rôzne „neviditeľné“ atramenty, či vtipné triky typu „do obálky vložím falošný list, skutočnú správu napíšem na obálku a prelepím ju známkou“.

Asi prvú steganografickú metódu fungujúcu čisto v textovej podobe vymýšľa začiatkom 17. storočia Francis Bacon. Spočíva v tom, že autor použil dva rôzne atribúty písma (napríklad v tlačenej podobe mohlo ísť o dva rôzne rezy písma). Do textu sa potom tajná správa ukryje tak, že každá pätica znakov otvoreného textu kóduje jeden znak textu ukrytého systémom podobným dvojkovej sústave.

Aj v súčasnosti má steganografia mnoho praktických využití. Jedným z nich je boj s cenzúrou. Vo viacerých krajinách je preto stále nelegálne používať šifrovanie. Zatiaľ čo použitie šifry môže disidenta rovno dostať do väzenia, v prípade úspešného použitia steganografie je pred nepovolanými utajené, že vôbec nejaká komunikácia prebehla. Iným, menej dramatickým využitím, je podpisovanie elektronických diel (tzv. digitálna vodotlač), ktorá autorovi umožní dokázať, že práve on je autorom dotyčného diela.

Mechanické šifrovacie a dešifrovacie stroje

V 19. až 20. storočí prichádzajú do módy rôzne mechanické šifrovacie pomôcky, neskôr aj takmer samočinné šifrovacie stroje. Zo začiatku sú zväčša jednoduché, ako napríklad dve otáčajúce sa kolesá uľahčujúce používateľovi šifrovanie. Nájdu sa však aj originálne nápady, napríklad koncom 19. storočia popisuje Edouard Fleissner von Wostrowitz variáciu mriežkovej šifry, pri ktorej vhodnú mriežku štyrikrát otočíme a zakaždým do jej políčok vyplníme štvrtinu šifry. Túto šifru preslávil Jules Verne vo svojej knihe Nový gróf *Monte Christo* (*Matyáš Sandorf*). Niekoľko mesiacov sa používala počas prvej svetovej vojny.

Viac detailov o Fleissnerovej mriežke nájdete na stránke http://en.wikipedia.org/wiki/Grille_(cryptography)

Po prvej svetovej vojne prichádzajú do módy komplexnejšie šifrovacie stroje. Tie zväčša obsahujú klávesnicu a niekoľko vymeniteľných rotačných valcov, ktoré treba pre dešifrovanie správy nastaviť rovnako ako pri šifrovaní. Najznámejším príkladom je nemecká Enigma, používaná počas druhej svetovej vojny. Tento nútený rozvoj kryptografie a obzvlášť kryptoanalýzy (hlavne v anglickom Bletchley Parku, kde okrem iných pôsobil aj Alan Turing) priniesol významné kroky smerom k informatike, ako ju dnes poznáme.

Obrázok1.3: Šifrovací stroj Enigma. V hornej časti sivej plochy sú vidieť tri výmenné rotačné valce (k nám otočené bočnou stranou), ktoré spolu s ich začiatočným natočením tvoria šifrovací kľúč. V spodnej časti je klávesnica, na ktorej operátor zadával text. Uprostred sa po každom stlačení klávesy rozsvietila žiarovka pod výstupným písmenom. (Obrázok je prebraný z enigma.wikispaces.com.)

Moderná kryptografia a bezpečnosť šifier

Hľadanie absolútne bezpečnej šifry

Ako sme sa dozvedeli v našom stručnom prehľade histórie šifrovania, základný princíp šifrovania bol spočiatku v tom, že tajomstvom bola samotná **metóda**, ktorou sa z otvoreného textu stával šifrový text. Dnes by sme povedali, že nutnou požiadavkou na bezpečnosť šifry bolo utajenie samotného šifrovacieho algoritmu.

Takéto šifrovanie však malo obrovskú nevýhodu: len čo sa tento algoritmus prezradil, šifra sa okamžite stala úplne nepoužiteľnou. A prax ukázala, že každý algoritmus sa skôr či neskôr prezradí. Už v Cézarovom prípade sa to stalo, a to pravdepodobne vtedy, keď sa Cicero pridal k jeho odporcom.

V súčasnosti sa pre šifrovacie systémy, ktorých bezpečnosť je postavená len na tom, že je utajené, ako fungujú, používa technický termín **security through obscurity** (v preklade „bezpečnosť (len) vďaka neprehľadnosti“). Veľká väčšina bezpečnostných expertov sa zhoduje v tom, že takéto systémy nemajú byť nikdy používané v situáciách, keď na bezpečnosti šifrovaných údajov skutočne záleží. Napriek tomu sa v praxi dodnes stretáme s takýmito systémami – a to zväčša v kontexte správ o ich neúspechu. Dôvodov môže byť viacero, od nedostatočného vzdelania autorov systému až po lobistické tlaky skupín, ktorým skutočne zabezpečený systém nevyhovuje. Nedávnym príkladom je použitie nedostatočne zabezpečených hlasovacích zariadení firmy Diebold v amerických prezidentských voľbách v roku 2004.

V renesančnej Európe potom prišlo k významnému posunu v prístupe k šifrovaniu: oddelil sa od seba šifrovací algoritmus, ktorý bol častokrát známy mnohým ľuďom, a **kľúč** – tajná informácia, ktorú zdieľali len odosielateľ a adresát, a ktorá mala len tomu správnemu adresátovi umožniť prečítanie zašifrovanej správy.

A práve o takýchto podmienkach uvažuje moderná kryptografia: Predpokladá sa, že šifrovací algoritmus je útočníkovi známy. Na to, aby bola šifra považovaná za bezpečnú, musí byť aj vtedy ťažké ju rozlúštiť – ak útočník nepozná použitý šifrovací kľúč.

Ide to však vôbec? Existuje bezpečná šifra? V roku 1841 napísal Edgar Allan Poe vo svojom diele *Pár slov o tajnom písaní* vetu: „Yet it may be roundly asserted that human ingenuity cannot concoct a cipher which human ingenuity cannot resolve.“ Vo voľnom preklade: „Môžeme smelo prehlásiť, že ľudský dôvtip nie je schopný vymyslieť šifru, ktorú by ľudský dôvtip nedokázal vyriešiť.“ Toto prehlásenie však prišlo ešte v dobe klasickej kryptografie, kedy sa všetko šifrovanie dalo robiť s perom a papierom, prípadne pomocou jednoduchých mechanických strojov, ktoré však len uľahčovali zdĺhavú prácu človeka.

Ďalekopis je zariadenie, ktoré vedelo po telekomunikačných linkách posielať textové správy. Tie sme u nás poznali pod názvom telegramy. Za povšimnutie stojí, že išlo vlastne o priameho predchodcu dnes populárnych textových správ, tzv. esemesiek.

To, že Edgar Allan Poe nemal pravdu, sa ukázalo až o niekoľko desaťročí neskôr. V roku 1917 si Gilbert Vernam dal patentovať vynález, ktorý ku ďalekopisu pripojil vopred pripravený kľúč na diernej páske. Potom vždy, keď odosielateľ zadal písmeno správy, prečítal ďalekopis ďalšie písmeno kľúča a z nich pomocou vhodnej jednoduchej operácie vypočítal zašifrované písmeno, ktoré potom odoslal.

> Spomínanou jednoduchou operáciou mohlo byť napríklad sčítanie: Ak si povieme, že A=1, B=2, C=3, ..., Z=26, tak môžeme sčítať ľubovoľné dve písmená, a dostať tak tretie. Napríklad sčítaním C=3 a D=4 by sme dostali 7=G. Ak by výsledok prekročil 26, vrátime sa na začiatok abecedy – teda 27 bude opäť A, 28 bude B, a tak ďalej. (Pre zaujímavosť, v pôvodnom Vernamovom stroji bol namiesto sčítania použitý tzv. bitový xor: v posielanom znaku sa znegovali bity zodpovedajúce predierovaným pozíciám na páske.)

Až v roku 1949 publikoval Claude Shannon vedecký článok, ktorým v princípe začína éra modernej kryptografie. Shannon v ňom uvádza princíp, ktorý sme si tu už my

uviedli: Šifra musí byť bezpečná, teda odolná voči útočníkom, aj vtedy, keď títo poznajú jej algoritmus. Samotný princíp nie je pre kryptografiu ničím novým: prvá jeho známa publikácia pochádza z roku 1883 a podľa jej autora sa mu dodnes hovorí Kerckhoffsov princíp.

Shannon však pojem bezpečnosti ako prvý dokáže exaktne matematicky definovať: šifra je **absolútne bezpečná** vtedy, ak sú pre daný šifrový text bez znalosti kľúča všetky otvorené texty rovnako pravdepodobné. Pre útočníka, ktorý má zašifrovanú správu, ale nemá kľúč, to teda znamená, že netuší vôbec nič o prenášanej správe.

Shannon následne dokazuje, že za jedného jednoduchého predpokladu je Vernamova šifra absolútne bezpečná. Spomínaným predpokladom je to, že použitý kľúč musí byť taký dlhý ako šifrovaná správa a jeho znaky musia byť volené náhodne.

Uvedieme jeden príklad, ako sa v súčasnosti môže používať Vernamova šifra. Pri návšteve rodnej krajiny sa veľvyslanec zastaví na ministerstve zahraničných vecí. Tam im počítačový program vygeneruje niekoľko gigabajtov (pseudo)náhodných znakov. Túto postupnosť napália na dve DVD. Jedno z týchto dvoch identických DVD zostane zatvorené v trezore na ministerstve, druhé si veľvyslanec zoberie so sebou do sveta. A následne vždy, keď potrebujú spolu bezpečne komunikovať, zoberie odosielateľ z DVD toľko znakov, ako dlhú správu posiela, a použije ich ako kľúč na jej zašifrovanie. Kým sú obe DVD bezpečne utajené pred svetom, majú komunikujúce strany úplnú istotu, že nik nepovolaný ich šifrovanú komunikáciu nedokáže rozlúštiť.

Princíp asymetrického šifrovania

Ďalší zlomový okamih kryptografie prišiel v roku 1976, kedy Diffie a Hellmann prišli s prevratným nápadom: asymetrickým šifrovaním. Tento ich nápad totiž riešil problém, ktorým dovtedy trpeli úplne všetky šifrovacie systémy: potrebu vopred sa stretnúť. Staré šifrovacie systémy boli založené na myšlienke, že sa odosielateľ a adresát vopred dohodnú na šifrovacom systéme. V neskorších rokoch mohol síce byť šifrovací systém verejne známy, ale aj tak sa odosielateľ a adresát potrebovali vopred dohodnúť na kľúči, ktorý nik iný nepoznal.

Čo však, ak sa ocitnem v situácii, keď zrazu potrebujem niekomu novému poslať utajenú správu? Spoločný kľúč dohodnutý, samozrejme, nemáme a dohodnúť sa na nejakom nemáme možnosť. Čo v takejto situácii? (Uvedomte si, že by sme sa mohli stretnúť, či inak si dohodnúť kľúč, tak by som mu rovno mohol oznámiť tú správu, a nepotrebovali by sme sa dohadovať.)

Jedno možné riešenie tejto zdanlivo neriešiteľnej situácie si môžeme priblížiť vtipnou logickou úlohou: Známy archeológ Indiana Jones našiel prastarú truhlicu plnú artefaktov. Chcel by ju poslať svojmu otcovi do Anglicka. Ako to však spraviť? Ak ju pošle nezamknutú, po ceste ju určite niekto vykradne. Ale ak ju pošle poriadne zamknutú, nedostane sa k jej obsahu ani jeho otec. Mohol by samozrejme poslať zamknutú truhlicu a nejakou inou cestou aj kľúč od nej, čo ak ho však niekto sledoval a odchytí na pošte aj truhlicu, aj kľúč?

Indiana Jones vymyslel nasledovné riešenie: Kúpi si poriadny zámok a zamkne truhlicu. Kľúč si nechá, truhlicu pošle svojmu otcovi. Ten ju síce nevie otvoriť, vie však spraviť niečo iné: aj on si kúpi zámok a tiež ním truhlicu zamkne. (Pozri obrázok nižšie.) Dvakrát zamknutú truhlicu, teda stále bezpečne zatvorenú, odošle späť svojmu synovi. Keď Indiana Jones dostane truhlicu späť, zoberie svoj kľúč, odomkne svoj zámok a odstráni ho z truhlice. Zostane mu truhlica, ktorá je naďalej zamknutá – teraz už však len zámkom jeho otca. Truhlicu teraz Indiana Jones opäť pošle svojmu otcovi. No a keď ju ten dostane, odomkne svoj zámok a má pred sebou otvorenú truhlicu plnú artefaktov.

obrázok 1.4: Ilustrácia zamknutia truhlice pomocou dvoch nezávislých zámkov. (Obrázok je prevzatý z dreamstime.com.)

Príklad s truhlicou nám teda ukazuje jeden veľmi dôležitý poznatok: v niektorých situáciách môže existovať spôsob bezpečnej komunikácie, ktorý nebude potrebovať žiadne vopred zdieľané spoločné tajomstvo! A práve takéto úvahy umožnili neskôr objav asymetrickej kryptografie.

Existujú aj priame kryptografické analógie vyššie popísaného postupu s truhlicou. V praxi má ale tento postup jeden zjavný nedostatok: správu (truhlicu) bolo potrebné poslať až trikrát a museli sa postupne zapojiť obaja aktéri. Diffie s Hellmannom prišli na ešte efektívnejší spôsob. Ten si môžeme „sedliacky“ priblížiť nasledovne: Predstavme si, že existuje uzol, ktorý je veľmi ťažké rozviazať. Ak je náš adresát jediným na svete, kto ho rozviazať dokáže, môžeme tento uzol použiť na zabezpečenie truhlice. Trik je v tom, že ani my ho nemusíme vedieť rozviazať – stačí, keď ho vieme zaviazať!

Šifra RSA

Už rok po Diffieho a Hellmannovom nápade, v roku 1978, prišli Rivest, Shamir a Adleman s prakticky použiteľnou realizáciou takejto šifry. Šifra RSA (nazvaná podľa prvých písmen ich priezvisk) sa v praxi používa odvtedy až dodnes.

Táto šifra je založená na veľmi jednoduchom pozorovaní. Viete vypočítať, koľko je 479-krát 521? Určite. S kusom papiera a perom túto úlohu zvládne za minútu aj bežný základoškolák. No viete povedať, aké dve prvočísla treba vynásobiť, aby sme dostali výsledok 252 179? Asi by trvalo dosť dlho, kým by ste sa prepracovali k odpovedi – ak by ste ju vôbec v rozumnom čase našli.

Náš popis šifry RSA je nutne zjednodušený, v skutočnosti súkromný aj verejný kľúč obsahujú aj ďalšie údaje potrebné na technickú realizáciu algoritmov. Naším cieľom nie je odborná presnosť tohto textu, ale zvýraznenie hlavnej myšlienky.

Podobne ako pre nás je táto úloha ťažká aj pre počítač: násobiť čísla dokáže veľmi rýchlo, no rozkladať číslo na súčin (odborne sa tejto úlohe hovorí faktorizácia) je ťažké, dodnes nie je známy žiaden algoritmus, ktorý by bol porovnateľne efektívny ako tie na násobenie.

Presná zložitosť problému faktorizácie nie je známa, je predmetom intenzívneho vedeckého výskumu. V súčasnosti najlepšie faktorizačné algoritmy sú variácie algoritmu general number field sieve. Súčasným rekordom je faktorizácia 232-ciferného čísla, tzv. čísla RSA-768, na súčin dvoch prvočísel. Tento výpočet reálne trval približne rok, pričom potreboval 50 tisíc rokov strojového času. Pre porovnanie, bežný súčasný počítač zvláda pri použití efektívneho algoritmu za sekundu vynásobiť miliónciferné čísla.

Ako však na tomto pozorovaní založiť šifru?

> Každý človek, ktorý chce prijímať správy šifrované pomocou RSA, si musí vygenerovať svoj kľúč. Toto spraví tak, že si zvolí dve prvočísla p a q. (V praxi majú tieto prvočísla medzi sto a tisíc cifier. A, samozrejme, človek ručne nerobí nič, všetko sa udeje vnútri v počítači.) Tieto prvočísla si zapamätá, on musí byť jediný, kto ich pozná. Hovoríme, že tieto prvočísla tvoria jeho **súkromný kľúč**. Ďalej vypočíta ich súčin $n = pq$ a ten zverejní. Tejto hodnote hovoríme **verejný kľúč**.

Verejný kľúč, ako už naznačuje jeho názov, môže poznať každý. Existujú dokonca stránky, ktoré slúžia ako „telefónny zoznam“: podľa e-mailovej adresy adresáta tam nájdete jeho verejný kľúč. (Presnejšie, vhodný program ho tam nájde za vás.)

Prínos Rivesta, Shamira a Adlemana bol v tom, že vymysleli algoritmy na šifrovanie a dešifrovanie, ktoré majú nasledovnú dôležitú vlastnosť: **Na šifrovanie stačí poznať verejný kľúč, na dešifrovanie je nutné poznať súkromný kľúč.**

Dôležité je uvedomiť si, že zverejnenie verejného kľúča nijak neohrozuje bezpečnosť šifry. Ak nepriateľ pozná verejný kľúč, nemá ako zistiť z neho súkromný kľúč: číslo *n* je totiž také obrovské, že na nájdenie *p* a *q* by aj najlepší známy program potreboval miliardy rokov.

Ukážme si to na príklade. Alica má na svojej webstránke zverejnený svoj verejný kľúč. Keď jej chce Boris poslať dôvernú správu, môže si tento kľúč stiahnuť na svoj počítač a pomocou neho šifrou RSA posielanú správu zašifrovať. Takto dostane zašifrovanú správu. Tá môže vyzerať približne nasledovne:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.7 (MingW32) - WinPT 1.2.0
hQIOA9arcc/jGAv/EAf/cP0gFDneg5BKr7AzLqxgpkWeqUUKj83Db4H/Zd5EnArh
VxzDXs0ZQcG/e438tdaN1Bh17XYmdJL/kqlo47vBeH2b09CTkGNHejv+r25mf1+N
j4991ixWwHNqHcDJiir1Ncaog5BOmKRMNnu2ALL0/thXsG7+2KL0vaAqXjRUq6V8
ncuwS9N75ov76zBSjVTwKdKHUh31YaZRQDDQojAnNl+wLS1DzkMReLjHCtd88KgK
utaHA9g5GONTLmSdcLwU7v/V2ysVP4kifRAcSRy2nL1qqoeSmkK4f3ATnAu5SUKP
djcq+X5QAoOSr5zmR7/xlsEuycg/k2whZLNmU/qf7wf/TjQp/3cExvc/aAkpO58h
cIJNm7ZhJ/pS26f
-----END PGP MESSAGE-----
```

Túto správu teraz môže Boris pokojne poslať Alici e-mailom. Aj keby ju totiž niekto po ceste odchytil, neprečíta ju. Na dešifrovanie totiž treba **Alicin** súkromný kľúč – ten, ktorý „pasuje“ k verejnému kľúču použitému pri šifrovaní. A Alica je jediná, kto tento kľúč má. Ona jediná sa teda k obsahu Borisovej správy vie dostať.

Všimnite si, že Alica a Boris sa nikdy nemuseli stretnúť a dohodnúť na spoločnom tajomstve. A v tom je práve hlavný praktický význam tejto šifry.

Ako funguje elektronický podpis

Na šifrovanie, napríklad šifrou RSA, sa môžeme dívať ako na dvojsmerný proces: hocikto vie zobrať reťazec a zašifrovať ho, ale len vlastník súkromného kľúča vie zobrať reťazec a odšifrovať ho. Šifra RSA má navyše dve užitočné vlastnosti. Prvou je, že nie len zašifrovať, ale aj odšifrovať sa dá ľubovoľný reťazec. (Samozrejme, ak odšifrujeme nejaký nezmysel, dostaneme ako výsledok iný nezmysel.) Druhou vlastnosťou, ktorú teraz využijeme, je, že pri šifre RSA sú šifrovanie a dešifrovanie komutatívne: ak zoberieme správu, **najskôr** ju odšifrujeme a **potom** ju zašifrujeme, dostaneme späť pôvodnú správu.

A presne na týchto pozorovaniach je založené fungovanie elektronického podpisu.

Predstavme si napríklad, že Alica má v elektronickej forme vyplnené daňové priznanie. Chcela by ho podpísať, a tak potvrdiť údaje v ňom uvedené. Ako to spraví? Tak, že zoberie dotyčný súbor a **odšifruje** ho pomocou svojho súkromného kľúča. Výsledkom tejto operácie je reťazec predstavujúci jej podpis. Ten Alica pripojí k samotnému dokumentu. Keď chce Karol **overiť pravosť** tohto **podpisu**, môže zobrať Alicin verejný kľúč, pomocou neho **zašifrovať** jej podpis a overiť, že sa výsledok rovná pôvodnému dokumentu.

Opäť situáciu nutne zjednodušujeme. V praxi sa kvôli rýchlosti a tiež úspore miesta tieto operácie nerobia s celým dokumentom, ale iba s jeho tzv. kryptografickou hešovacou hodnotou - krátkym reťazcom, ktorý je akýmsi „odtlačkom“ celého dokumentu. S podobnou témou sa neskôr stretneme pri pravdepodobnostných algoritmoch.

Prečo to celé funguje? Pretože Karol vie, že Alica je jediná, kto vie pomocou jej kľúča aj dešifrovať – a teda dotyčný podpis musela vyrobiť ona, nik iný na to nemá dostatok informácií.

Zaručený elektronický podpis a certifikačné autority

Ak by Alica teraz chcela takto podpísané daňové priznanie z pohodlia svojho domova podať, nefungovalo by to. Chýba totiž jeden dôležitý článok: prepojenie medzi verejným kľúčom (súborom získateľným kdesi na internete) a Alicou ako fyzickou osobou. Štát nemá prečo len tak uveriť, že Alicin verejný kľúč je zrovna tento a nie jeden zo sta iných.

Aj na toto sa však myslelo. U nás na Slovensku túto problematiku rieši Zákon 215/2002 Z.z. o elektronickom podpise. Tento definuje tzv. **zaručený elektronický podpis**, ktorý má, zjednodušene povedané, rovnakú právnu váhu ako fyzický podpis na klasickom dokumente. Ako vieme takýto podpis vyrobiť? Chýbajúci medzičlánok – záruku korešpondencie medzi verejným kľúčom a fyzickou osobou – zabezpečujú inštitúcie nazývané **akreditované certifikačné autority**. Celý proces si môžeme jednoducho popísať nasledovne:

- Alica zoberie svoj občiansky preukaz a navštívi niektorú akreditovanú certifikačnú autoritu (CA).
- Pracovník CA si overí jej totožnosť a následne jej vygeneruje súkromný a verejný kľúč.
- Ku kľúču jej navyše vydá **certifikát**, teda potvrdenie, že ide naozaj o Alicin kľúč. (Tento certifikát je vlastne elektronický dokument obsahujúci potrebné údaje a podpísaný kľúčom dotyčnej certifikačnej autority.)
- Od tejto chvíle vie Alica vyrábať zaručených elektronických podpisov, koľko len chce. Ku podpísanému dokumentu následne priloží aj certifikát, ktorý príjemcovi umožní overiť totožnosť odosielateľa.

Praktické aplikácie kryptografie

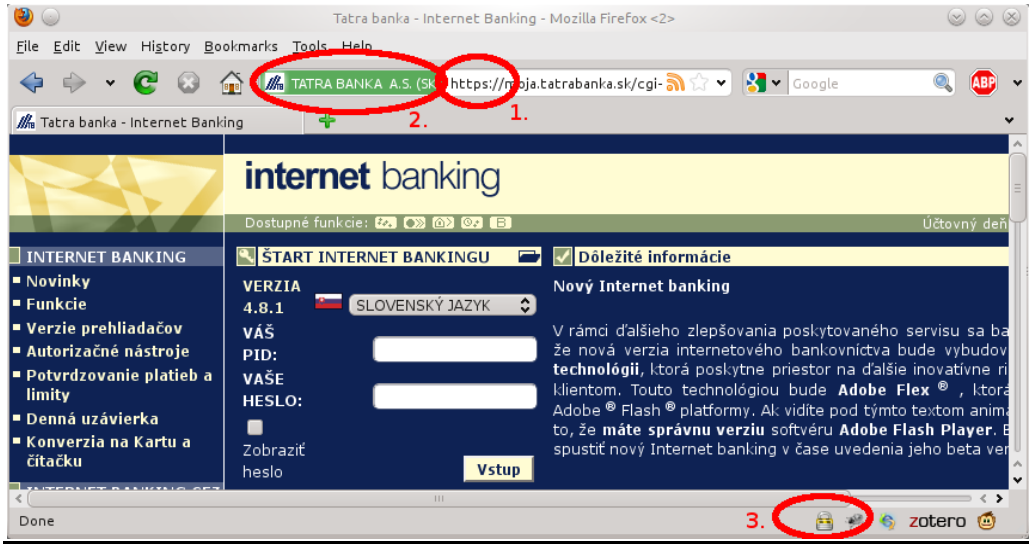
Zabezpečené webstránky

Máloktorý používateľ počítača si uvedomuje, že keď si prezeráte webstránky, skoro všetky údaje posielané medzi vaším počítačom a servermi sa prenášajú (pomocou protokolu HTTP) v nešifrovanej podobe. Hocikto teda môže túto komunikáciu odpočúvať.

Sú však situácie, kedy toto nechceme: napríklad také čítanie súkromnej elektronickej pošty, alebo platenie pomocou kreditnej karty. Na tieto účely bol vymyslený protokol HTTPS („S“ ako „secure“, presnejšie ide o kombináciu protokolov HTTP a SSL/TLS). Pri tomto protokole komunikácia prebieha v zašifrovanej podobe.

Dôveryhodné webstránky, ako napríklad internet banking vašej banky, vás pri návšteve zväčša automaticky presmerujú na HTTPS verziu prihlasovacej stránky. To, že na stránku pristupujete prostredníctvom šifrovanej komunikácie, spoznáte

ľahko priamo v prehliadači. V prvom rade podľa protokolu uvedeného v URL adrese stránky: tá nebude začínať znakmi „http://“, ale znakmi „https://“. Väčšina prehliadačov tiež zobrazí ikonku zámku, ak je obsah prenášaný šifrovane.



obrazok: 1.5: Základné upozornenia na šifrovanú stránku.

Na obrázku 1.5 vidíme príklad zabezpečenej stránky otvorenej v prehliadači Firefox. Číslom 1 je označený začiatok URL adresy, číslom 3 je označená ikonka zámku.

Samotné zabezpečenie komunikácie však ešte nestačí – ako vieme, že tá webstránka skutočne patrí našej banke, a nie nejakému zlodejovi? Tu, podobne ako pri zaručených elektronických podpisoch, musia prísť k slovu certifikáty. Niekto dostatočne dôveryhodný (certifikačná autorita) vydal banke certifikát potvrdzujúci jej totožnosť. A práve týmto certifikátom sa webserver banky potom preukáže vášmu prehliadaču. Prehliadač si overí, že certifikát banky je podpísaný certifikačnou autoritou, ktorej dôveruje, a výsledkom je to, čo vidíme na obrázku označené číslom 2: v adrese sa objaví farebne zvýraznená informácia o tom, že bol úspešne overený certifikát stránky, a tiež to, kto je jeho vlastníkom. Ak teda všetko prebehlo ako sme očakávali, môžeme pokojne zadať naše prihlasovacie údaje – máme dostatočnú dôveru, že ich skutočne posielame našej banke.

obrázok 1.6: Varovanie prehliadača. (Obrázok je prevzatý z seedbox.me.)

Ak overenie certifikátu neprebehlo úspešne, prehliadač vás presmeruje na stránku

podobnú tej na obrázku. Ak sa vám toto zrazu stane pri prihlasovaní sa na stránku, ktorá „dovtedy fungovala“, je veľmi pravdepodobné, že ste práve obeťou nejakého útoku. V takomto prípade rozhodne **nič neodsúhlasujte** a nikam nepíšte svoje heslo. Pokúste sa pripojiť ešte raz, ideálne tak, že priamo zadáte adresu do prehliadača. V prípade neúspechu kontaktujte správcu stránky (teda napríklad svoju banku).

Niekedy sa môže stať, že takéto varovanie uvidíte aj na legitímnej stránke. Uznávanou certifikačnou autoritou podpísané certifikáty sú totiž drahé, a tak webstránky prevádzkované neziskovými inštitúciami často takéto certifikáty nemajú. Aby fungovalo HTTPs spojenie na ne, vygenerujú si teda certifikát samy. To však vedie k varovaniu ukázanému na obrázku: prehliadač nás upozorňuje, že síce mu stránka ukázala certifikát, ale nepodarilo sa mu overiť jeho pravosť.

Ak pristupujete na takúto webstránku, skontrolujte si, či jej certifikát obsahuje správne údaje. Potom môžete urobiť výnimku a pridať tento certifikát medzi tie, ktorým má váš prehliadač dôverovať. (Na obrázku je šípkou označený príklad miesta, kde môžete toto schválenie spraviť v jednom z bežných prehliadačov.) Pri nasledujúcich návštevách dotyčnej stránky sa už varovanie zobrazovať nemalo. Je samozrejme odporúčané k tomuto kroku pristúpiť len vtedy, keď dotyčnej stránke dostatočne dôverujete. Autori tohto textu to nikdy nerobia na stránkach, ktoré by ich mohli napríklad pripraviť o peniaze alebo zneužiť ich dôverné osobné údaje.

Šifrovanie a podpisovanie e-mailov a súborov na disku

Štandardným programom používaným na asymetrické šifrovanie a dešifrovanie pomocou verejných a súkromných kľúčov je program PGP (Pretty Good Privacy, teda po našom „celkom slušné súkromie“). Ide o program pre OS Windows, ktorý v roku 1991 napísal Philip Zimmermann. Podľa jeho formátu vstupu a výstupu bol neskôr vytvorený otvorený štandard OpenPGP (RFC 4880), ktorý dnes implementujú aj viaceré iné programy, softvéry. Slobodnou a open source alternatívou k PGP je program gpg (GnuPG, GNU Privacy Guard). Ten nájdete v každej distribúcii Linuxu, nainštalovať sa však samozrejme, dá aj pod Windows.

Niektoré programy na prácu s elektronickou poštou majú priamo v sebe zabudovanú podporu šifrovania. Napríklad vo v súčasnosti obľúbenom programe Thunderbird sa dá nainštalovať rozšírenie s názvom Enigmail, ktoré vám následne umožní šifrovať a podpisovať posielané e-maily.

Existujú tiež iné spôsoby šifrovania dát. V niektorých moderných notebookoch sa napríklad stretneme s hardvérovou podporou šifrovania celého disku. Existujú aj viaceré softvérové produkty, ktoré umožňujú mať šifrované súbory, partície pevného disku či celé disky. Za všetky spomeňme slobodný open source program TrueCrypt, ktorý funguje na všetkých bežných platformách. (Hardvérové šifrovacie metódy aj TrueCrypt však používajú moderné verzie symetrických šifier, keďže sú rýchlejšie a asymetriu pri ich bežnom použití aj tak nie je kde využiť.)

Pravdepodobnostné algoritmy

V tejto časti sa budeme zaoberať využitím náhody pri riešení úloh. Ukazuje sa, že šikovné využitie náhody nám môže pomôcť vyriešiť úlohy, ktoré „klasickými“ technikami nevieme prakticky riešiť. Cena, ktorú za to zaplatíme, je zvyčajne v strate istoty, že vypočítané riešenie je optimálne, resp. správne. Prekvapujúce je, že túto neistotu vieme často ľubovoľne zmenšiť až na požadovanú mieru, ktorú sme ochotní akceptovať.

Tradičné algoritmy, s ktorými ste sa väčšinou stretávali v doterajšom vzdelávaní, sa označujú aj ako *deterministické*. V každom kroku výpočtu algoritmu je jednoznačne určené, aký bude nasledujúci krok a hodnoty premenných. Úsilie tvorcu algoritmu je dokázať, že algoritmus daný problém (vždy) vyrieši správne a rýchlo (to väčšinou znamená, že počet operácií, ktoré algoritmus vykoná na nájdenie riešenia, závisí polynomiálne od veľkosti vstupných údajov).

Pravdepodobnostné algoritmy okrem vstupných údajov používajú ešte aj „skrytý“ vstup – zdroj náhodných čísiel, na základe ktorého robia počas behu rozhodnutia. Beh pravdepodobnostného algoritmu na rovnakých vstupných údajoch môže byť vždy iný. Pri návrhu pravdepodobnostných algoritmov sa usilujeme ukázať, že sa budú pravdepodobne správať dobre pre ľubovoľné vstupné údaje (pravdepodobnosť závisí len od náhodných čísiel, ktoré používa algoritmus, ale nie od vstupných údajov).

Výhodou mnohých pravdepodobnostných algoritmov je ich jednoduchosť a efektívnosť. Pre mnoho problémov je pravdepodobnostný algoritmus najjednoduchší alebo najrýchlejší, alebo oboje súčasne. Na druhej strane, ak chceme o pravdepodobnostnom algoritme ukázať nejaké vlastnosti (napríklad, že pravdepodobnosť zlého výsledku bude nízka), vyžaduje si to mnohokrát použitie zložitejšej matematiky a vedomostí z teórie pravdepodobnosti. V prvej podkapitole preto uvádzame stručný úvod do pravdepodobnosti, ktorý rekapituluje potrebné poznatky a na ktorý sa budeme odvolávať pri analýze prezentovaných algoritmov. Našou snahou bolo vybrať také príklady algoritmov, ktoré dostatočne vystihujú základné myšlienky používané pri konštrukcii pravdepodobnostných algoritmov, a zároveň si pri ich analýze vystačíme len s relatívne jednoduchou teóriou pravdepodobnosti. Pri algoritmoch sme sa usilovali vždy uviesť potrebné argumenty, ktoré zdôvodňujú uvádzané postupy. Inak by hrozilo, že uvedieme len akúsi zbierku, možno zaujímavých, postupov, ktorým môžeme, ale nemusíme veriť.

Pravdepodobnostné algoritmy sa používajú v algoritmoch z teórie čísiel, v údajových štruktúrach, testovaní zhody, matematickom programovaní, grafových algoritmoch, v zisťovaní počtu určitých štruktúr, paralelných a distribuovaných výpočtoch, pravdepodobnostných dôkazoch existencie a ďalších.

Zaujímavou a ťažkou oblasťou je tzv. *derandomizácia* – úprava pravdepodobnostných algoritmov na rovnako efektívne deterministické.

V nasledujúcom texte si uvedieme príklady viacerých pravdepodobnostných algoritmov a ukážeme si, že sú dva základné typy pravdepodobnostných algoritmov: *Monte Carlo* a *Las Vegas*.

Čo myslíte, podľa čoho boli tieto algoritmy pomenované a prečo?

1. Úvod do pravdepodobnosti

Slovíčka „pravdepodobne“ a „nepravdepodobne“ používame v bežnom živote často. S nimi úzko súvisí pojem *náhoda*. Veď koľkokrát sme už povedali, že náhodou sa niečo stalo. Zvyčajne to myslíme vo význame, že sa niečo udialo neplánovane. Ak sa udeje niečo pravdepodobné, nijako sa tomu nedivíme. Naopak, ak sa udeje niečo nepravdepodobné, zdá sa nám to zvláštne. I keď veľmi dobre vieme, že nepravdepodobné ešte neznamená nemožné. Pretože ak by bolo niečo nemožné, tak sa to nikdy nestane. Čo si myslíte, je pravdepodobné alebo nepravdepodobné, že ak by sme si dnes podali lotériový tiket, vyhráme jackpot?

Aktivita 1

Diskutujte o tom, čo je to pravdepodobnosť. Kde v bežnom živote sa s ňou môžeme stretnúť?

Otázky náhody (a s ňou súvisiacej pravdepodobnosti) od nepamäti fascinovali ľudí a venovali sa im veľkí filozofi svojej doby. Je osud človeka vopred určený alebo celý život je plný náhod? Nie je náhoda len dôsledkom neschopnosti človeka dobre popísať svet okolo seba? Veď vari nie je výsledok takej zdanlivo náhodnej udalosti, akou je hod hracou kockou, presne predurčený silou a smerom hodenia kocky, fyzikálnymi vlastnosťami kocky a okolitého prostredia? Ak aj áno, tento jav je natoľko komplexný, že človek ho len ťažko dokáže deterministicky a presne popísať. Ľudia zvyčajne nemajú radi náhodu, pretože sa spája s chaosom a neporiadkom. My si však ukážeme, že aj náhoda má svoje zákonitosti a svoj poriadok. V ďalších podkapitolách si potom tiež ukážeme, ako tieto zákonitosti a náhodu využiť na efektívne riešenie informatických problémov.

Zákonitosti „náhody“ – t.j. teóriu pravdepodobnosti využívajú nielen informatici. Na jej základoch kasína vytvárajú nové hry a s jej využitím dokážu presne vypočítať očakávané výnosy z hry.

Aktivita 2

Diskutujte o hode kockou. Ak je kocka dokonalá – spravodlivá (čo reálna nikdy nie je), je pravdou, že každé číslo od 1 po 6 je rovnako očakávané – rovnako pravdepodobné? Aké to má dôsledky?

To, že každý z výsledkov od 1 po 6 je pri hode kockou rovnako pravdepodobný, možno interpretovať tak, že ak zrealizujeme 60 hodov, tak očakávame, že číslo 1 by malo padnúť približne 10-krát, číslo 2 rovnako približne 10-krát, atď. Samozrejme, dôležité je slovíčko *„očakávame“*. Keďže náhoda nie je presne predurčená, vôbec to tak nemusí byť. Ale asi by sme sa veľmi divili, ak by trebárs pri 60-tich hodoch číslo 1 nepadlo ani raz (i keď to nie je vylúčené).

Aktivita 3

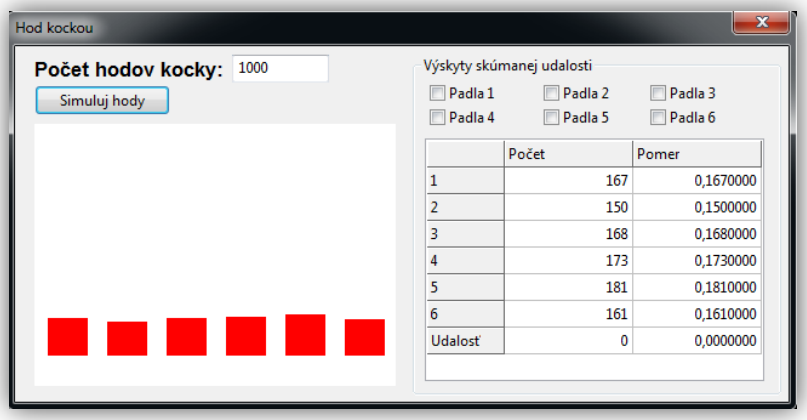
S využitím pripraveného softvéru (aktivita „Hod kockou“) simulujúceho hádzanie hracou kockou overte, či sa napĺňajú vyššie uvedené očakávania. Aký je pomer počtu hodov, pri ktorých padlo číslo 1, k celkovému počtu hodov? Ako závisí tento pomer od uskutočneného počtu simulovaných hodov?

Upozornenie: Pred použitím softvéru stlačte v úvodnom okne tlačidlo „Randomize“. V opačnom prípade budete pri každom spustení programu dostávať rovnaké výsledky. Prečo je to tak, bude vysvetlené v závere tejto kapitoly.

V programe si najprv zvolíme, koľko hodov kocky chceme simulovať. Následne kliknutím na tlačidlo „Simuluj hody“ spustíme simuláciu. Pri každom hode sa výsledok hodu kocky určí takto:

$$Random(6) + 1$$

Po skončení simulácie môžeme v tabuľke v pravej časti okna vidieť pre každé číslo na kocke údaj, pri koľkých simulovaných hodoch padla daná hodnota. Okrem absolútnej početnosti v stĺpci *Počet*, v stĺpci *Pomer* nájdeme relatívne početnosti padnutia jednotlivých čísel. V ľavej časti okna sú relatívne početnosti znázornené vizuálne vo forme stĺpcového grafu.



Obrázok 2.1: Simulácia hádzania hracej kocky

Všimnime si, že čím väčší počet hodov kocky simulujeme, tým viac sa blíži výsledok simulácie k očakávaniu, t. j. k tomu, že každé číslo na kocke padne približne rovnaký počet ráz. Zo samotných početností padnutia jednotlivých čísiel na kocke je však ťažšie vidieť to, ako veľmi sa líši očakávaný počet od skutočného počtu. Ak sa očakávaný a skutočný počet padnutí nejakého čísla líši o 20, je to veľa alebo málo? Intuitívne je nám jasné, že to záleží od toho, aký bol celkový uskutočnený počet hodov. Trebárs pri 100 hodoch je rozdiel 20 naozaj dosť veľký, no pri 10000 hodoch je tento rozdiel zanedbateľný. Preto bude pre nás oveľa cennejšia informácia o *relatívnej početnosti* vzniku nejakého „skúmaného javu“ – v našom prípade padnutia nejakého konkrétneho čísla na kocke. Relatívna početnosť je vyjadriteľná vzťahom:

$$\frac{\text{počet pokusov (hodov), pri ktorých nastalo to, čo nás zaujíma}}{\text{celkový počet pokusov (hodov)}}$$

Jednoduchými logickými úvahami dostaneme, že relatívna početnosť je vždy číslo z intervalu $\langle 0, 1\rangle$. Ak toto číslo vynásobíme číslom 100, dostaneme, v koľkých percentách prípadov nastalo to, čo nás zaujíma. S využitím relatívnej početnosti (resp. percent) už dokážeme lepšie povedať, ako veľmi sa očakávaná hodnota líši od skutočnej hodnoty.

Proces, ktorého výsledok je náhodný, nazývame *experiment* alebo *pokus*. V našom prípade je pokusom jeden hod kockou. To, že dostaneme nejaký výsledok, nazývame *výskytom javu* alebo *udalosti*. Udalosťou je napríklad to, že padlo číslo 1. Inou udalosťou je, že padlo číslo 2. Ale udalosťou môže byť aj to, že padlo párne číslo. Niektoré udalosti sú akýmsi spôsobom základné – „nedeliteľné“. Pri hode kockou je to padnutie konkrétneho čísla. Tieto udalosti nazývame *elementárnymi*. Iné udalosti môžu byť vyjadrené pomocou elementárnych udalostí – napríklad padnutie párneho čísla znamená, že padlo číslo 2, číslo 4, alebo číslo 6, t. j. udalosť padnutia párneho čísla vieme vyjadriť pomocou výskytu troch elementárnych udalostí.

Aktivita 4

Pripravený softvér simulujúci hádzanie kockou (aktivita „Hod kockou“) umožňuje pomocou zaškrtávacích políčok vyšpecifikovať ľubovoľnú udalosť a následne skúmať, pri akom počte hodov táto udalosť nastala. Napríklad, ak chceme skúmať padnutie párneho čísla, označíme políčka „Padla 2“, „Padla 4“ a „Padla 6“. V poslednom riadku tabuľky vidíme pre zvolenú udalosť absolútny, ale aj relatívny počet výskytov danej udalosti.

Pre zvolenú udalosť (napr. padlo nepárne číslo, padlo číslo deliteľné 3, atď.) určte, z akých elementárnych udalostí sa táto udalosť skladá. Odhadnite očakávanú relatívnu početnosť výskytov tejto udalosti a porovnajte ju s výsledkami simulácie.

Určite si pamätáte vzťah, že pravdepodobnosť náhodnej udalosti je definovaná ako podiel počtu elementárnych udalostí, pri ktorých táto udalosť nastane, k celkovému počtu všetkých elementárnych udalostí. Ak teda máme n elementárnych rovnako očakávaných (pravdepodobných) udalostí, pravdepodobnosť výskytu každej z nich je $1/n$. Ak udalosť A nastane pri m z nich, pravdepodobnosť výskytu udalosti A, označujeme ju ako $P(A)$, je definovaná výrazom:

$$P(A) = m/n$$

V prípade hodu kockou máme 6 elementárnych udalostí, a tak pravdepodobnosť elementárnej udalosti je $1/6$. Ak nás zaujíma pravdepodobnosť toho, že padne párne číslo, dostávame pravdepodobnosť $1/2 = 3/6$ (pri 3 zo 6 elementárnych udalostí máme párne číslo). Všimnime si, že čím väčší počet hodov kocky

Poznamenajme, že táto definícia platí iba v prípade, že každá elementárna udalosť je rovnako očakávaná – pravdepodobná. „Falošná“ hracia kocka je príkladom kocky, v ktorej elementárne udalosti nemajú rovnakú pravdepodobnosť. Čo však platí, je to, že súčet pravdepodobností elementárnych udalostí je vždy 1.

simulujeme, tým viac sa relatívna početnosť (nazývaná aj *relatívna frekvencia*) blíži k pravdepodobnosti. Toto pozorovanie nám ponúka pohľad na inú definíciu pravdepodobnosti, ktorá hovorí, že pravdepodobnosť výskytu udalosti je relatívna početnosť pokusov (relatívna frekvencia), pri ktorých sa daná udalosť vyskytla, pri nekonečnom opakovaní pokusov.

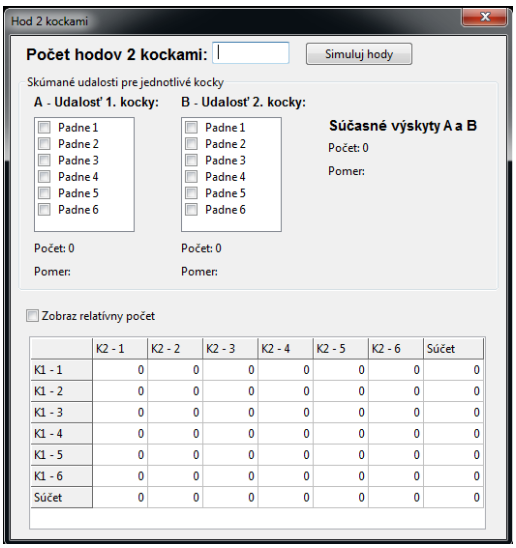
Použitím vzťahu ľahko zrátame, že pravdepodobnosť padnutia čísla menšieho alebo rovného ako 2 (teda čísla 1 alebo 2) je $2/6 = 1/3$. Aká je však pravdepodobnosť, že táto udalosť nenastane? Na základe vzorca dostaneme, že je to $2/3$ (máme $4 = 6 - 2$ elementárne udalosti zo 6, kedy udalosť nenastane). Vo všeobecnosti, ak udalosť A má pravdepodobnosť $P(A)$, tak pravdepodobnosť toho, že udalosť A nenastane (t.j. nastane doplnková udalosť k A - označujeme A^c) je $1 - P(A)$. Presnejšie vždy platí, že $P(A) + P(A^c) = 1$ (vždy udalosť A alebo nastane alebo nenastane, nikdy nenastanú A a A^c súčasne). Z toho môžeme ľahko odvodiť aj ďalší dôležitý vzťah. Ak $P(A) \leq p$, tak $P(A^c) \geq 1 - p$, resp. ak $P(A) \geq p$, tak $P(A^c) \leq 1 - p$.

Úloha 1

Vo vreci máme 1000 (resp. n) kartičiek s číslami, pričom na 168 (resp. m) z nich sú napísané prvočísla. Vypočítajte, aká je pravdepodobnosť toho, že vytiahneme, resp. nevytiahneme kartičku s prvočíslom?

Skúsme teraz analyzovať situáciu, keď namiesto jednej hracej kocky hádžeme dvoma. Je jasné, že to, čo nám padne na jednej kocke, nijako nesúvisí s tým, čo nám padne na druhej kocke. Trebárs udalosť, že na prvej kocke padlo číslo 1, a udalosť, že na druhej kocke padlo číslo 6, nijako nesúvisia. Takéto udalosti nazývame *nezávislými*. Pozor však na jeden malý, ale podstatný detail, ktorý si musíme uvedomiť, ak chceme skúmať pravdepodobnosti pri hode dvoma kockami. V prípade hodu dvoma kockami máme až $36 = 6 \cdot 6$ elementárnych udalostí. Toľko je všetkých možných rôznych výsledkov vykonania pokusu (hodu): [1, 1], [1, 2], ..., [1, 6], [2, 1], [2, 2], ... Ak by sme sa rozhodli aj v tomto prípade spočítať pravdepodobnosť toho, že na prvej kocke padne číslo 1, opäť dostaneme, že je to $1/6 = 6/36$ ([1,1], [1, 2], ..., [1, 6]). Namiesto hádzania dvomi kockami môžeme dvakrát hádzať jednou kockou. Výsledok prvého hodu bude zodpovedať prvej kocke, výsledok druhého hodu druhej kocke. Tak, ako v prípade s dvoma kockami, aj teraz sú výsledky jednotlivých hodov nezávislé. Aká je pravdepodobnosť toho, že nám pri oboch hodoch padne číslo 6? Máme 36 elementárnych udalostí, z nich udalosť, ktorá nás zaujíma, vznikne len pri jednej: pravdepodobnosť je teda $1/36$.

Možné výsledky hodu dvomi hracími kockami:
[1, 1], [1, 2], [1, 3],
[1, 4], [1, 5], [1, 6],
[2, 1], [2, 2], [2, 3],
[2, 4], [2, 5], [2, 6],
[3, 1], [3, 2], [3, 3],
[3, 4], [3, 5], [3, 6],
[4, 1], [4, 2], [4, 3],
[4, 4], [4, 5], [4, 6],
[5, 1], [5, 2], [5, 3],
[5, 4], [5, 5], [5, 6],
[6, 1], [6, 2], [6, 3],
[6, 4], [6, 5], [6, 6]



Obrázok 2.2: Simulácia dvoch hodov kocky

Aktivita 5

Pripravený softvér v aktivite „Hod 2 kockami“ umožňuje simuláciou preskúmať pravdepodobnosti pri hádzaní dvomi hracími kockami. Po zadaní počtu hodov dvomi kockami (pri každom hádžeme oboma kockami) kliknutím na tlačidlo „Simuluj hody“ spustíme simuláciu. Koľkokrát nastali jednotlivé udalosti, si môžeme pozrieť v tabuľke v spodnej časti okna. Všimnime si, že súčet hodnôt v tabuľke je vždy rovný počtu hodov, resp. číslu 1 pri relatívnej početnosti. Pomocou zaškrtávacích políčok môžeme špecifikovať udalosť, ktorá nás zaujíma na prvej kocke, a udalosť, ktorej výskyt nás zaujíma na druhej kocke.

Zodpovedajú softvérové simulácie tomu, čo „predpovedajú“ matematické odvodenia? Nech $P(A)$ je pravdepodobnosť výskytu udalosti A na prvej kocke a $P(B)$ je pravdepodobnosť výskytu udalosti B na druhej kocke. Experimentálne skúste zistiť, aká bude pravdepodobnosť toho, že sa obe udalosti vyskytnú súčasne.

Aká je pravdepodobnosť, že pri oboch hodoch (na oboch kockách) padne párne číslo? Nahliadnutím do zoznamu všetkých možných výsledkov (elementárnych udalostí) zistíme, že párne číslo na oboch kockách máme v 9-tich prípadoch. Teda pravdepodobnosť je $9/36 = 1/4$. Poďme to zovšeobecniť. Nech $P(A)$ je pravdepodobnosť výskytu udalosti A pri hode jednou hracou kockou. Aká je pravdepodobnosť toho, že ak pokus zopakujeme dvakrát (resp. hodíme dvomi kockami), udalosť A nastane v oboch prípadoch (resp. na oboch kockách)? Nech $P(A) = m/n$. Pri dvoch opakovaniach máme $n \cdot n$ možných rôznych elementárnych udalostí. Z nich je pre nás priaznivých $m \cdot m$ udalostí. (Prečo?) Dostávame tak, že pravdepodobnosť toho, že udalosť nastane pri oboch opakovaniach, je $\frac{m \cdot m}{n \cdot n} = P(A) \cdot P(A)$. Podobným argumentom ľahko ukážeme, že ak pravdepodobnosť nastania udalosti A je $P(A)$ a pravdepodobnosť nastania udalosti B je $P(B)$, tak pravdepodobnosť toho, že na prvej kocke (resp. pri prvom hode) sa vyskytne udalosť A a súčasne na druhej kocke B (resp. pri druhom hode), je presne $P(A) \cdot P(B)$.

Úloha 2

Nech $P(A)$ je pravdepodobnosť udalosti A. Pokus opakujeme presne k-krát. Aká je pravdepodobnosť toho, že udalosť A

- nastane pri všetkých k opakovaniach,
- nenastane ani raz pri k opakovaniach,
- nastane aspoň raz pri k opakovaniach?

Úloha 3

Ak máme vo vreci 1000 kartičiek s číslami, pričom na 168 z nich sú prvočísla, aká je pravdepodobnosť, že ani na 20 pokusov (po každom pokuse vytiahnutú kartičku vrátime do vreca) nevytiahneme žiadne prvočíslo? Koľko pokusov musíme vykonať, aby pravdepodobnosť toho, že nevytiahneme žiadne prvočíslo, bola menšia ako 10^{-3}? Ak by kartičky boli na stole usporiadané za sebou v nejakom poradí a postupne ich otáčame zľava doprava, koľko otočení budeme potrebovať v najhoršom prípade na nájdenie kartičky s prvočíslom?

Riešenie Úlohy 2 nie je zložité, ak si uvedomíme isté súvislosti. Pravdepodobnosť toho, že udalosť A nastane pri všetkých k opakovaniach, získame zovšeobecnením vzťahu pre dve opakovania. Je to $P(A)^k$. Ak nás zaujíma pravdepodobnosť toho, že udalosť A nenastane ani raz pri k opakovaniach, uvedomme si, že je to presne to

isté, ako keby sme sa pýtali na pravdepodobnosť toho, že pri všetkých k opakovaniach nastane doplnková udalosť A^c k udalosti A. Keďže $P(A^c) = 1 - P(A)$, tak pravdepodobnosť toho, že A nenastane ani raz je $(1 - P(A))^k$. No a na záver, všimnime si, že to, že udalosť A nastane aspoň raz pri k opakovaniach je doplnkovou udalosťou k tomu, že A nenastane pri k opakovaniach ani raz. Teda pravdepodobnosť toho, že A nastane aspoň raz bude $1 - (1 - P(A))^k$.

2. Výpočty s využitím pravdepodobnosti

Prestavme si situáciu, že nepoznáme vzorec na výpočet obsahu kruhu. Ako by sme mohli experimentálne vypočítať obsah kruhu s polomerom r? Spravme „myšlienkový“ experiment. Kruh s neznámym obsahom vpíšme do štvorca s dĺžkou strany $2 \cdot r$. Ak by teraz začal padať dážď (kvapky padajú rovnomerne a náhodne), to, čo prirodzene očakávame, je skutočnosť, že čím väčšia plocha, tým na ňu spadne viac kvapiek. Dokonca očakávame, že počet kvapiek, ktorý spadne na nejakú plochu, resp. do nejakej oblasti, je priamo úmerný jej obsahu.

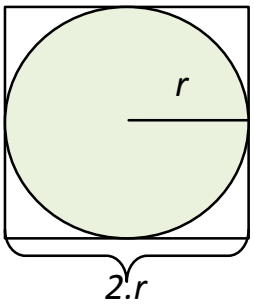
Poďme to skúsiť presne spočítať. Označme si obsah štvorca $S_{\blacksquare} = (2 \cdot r)^2 = 4 \cdot r^2$, neznámy obsah kruhu $S_{\odot}$, m nech je počet kvapiek, ktoré padli do kruhu, a nech n je celkový počet kvapiek, ktoré padli do štvorca. Podľa intuitívneho očakávania by malo platiť, že $S_{\odot}/S_{\blacksquare} = m/n$. Samozrejme, tento vzťah nebude platiť vždy, keďže kvapky padajú náhodne. No očakávame, že ak kvapiek spadne poriadne veľa, tak sa pomer m/n bude stále viac a viac blížiť k pomeru $S_{\odot}/S_{\blacksquare}$, podobne, ako to bolo pri simulovaní hodu kockou. Platí, že $S_{\odot}/S_{\blacksquare} = P(A)$, kde $P(A)$ je pravdepodobnosť udalosti, že kvapka padne do kruhu (tento vzťah súvisí s takzvanou geometrickou definíciou pravdepodobnosti). Keďže obsah štvorca poznáme, ako dôsledok očakávaného vzťahu $S_{\odot}/S_{\blacksquare} = m/n$ dostávame vzorec pre očakávaný obsah kruhu $S_{\odot} = \frac{m}{n} \cdot S_{\blacksquare} = \frac{m}{n} \cdot 4 \cdot r^2$. Program na pravdepodobnostný výpočet obsahu kruhu by mohol vyzerať takto:

```
M := 0;
for I := 1 to N do
begin
  X := 2 * R * Random;
  Y := 2 * R * Random;
  JeVKruhu := (Sqrt((X-R)*(X-R) + (Y-R)*(Y-R)) <= R);
  if JeVKruhu then
    Inc(M);
end;
ObsahKruhu := 4*R*R*M/N;
```

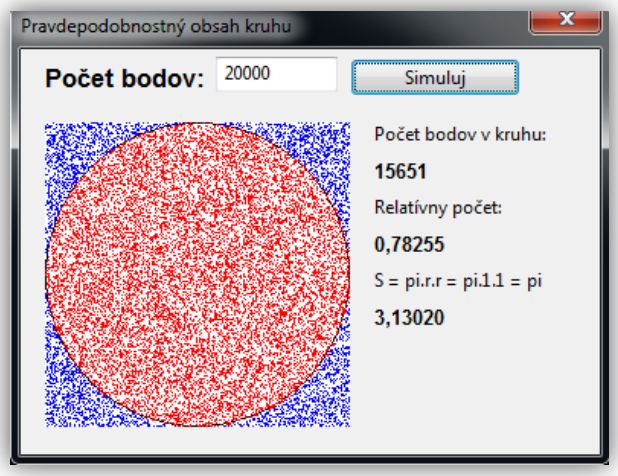
`Random` je funkcia, ktorá vráti náhodné reálne číslo z intervalu $\langle 0, 1)$. Vynásobením vrátenej hodnoty priemerom kruhu $2 \cdot r$ (čo je to isté ako dĺžka strany štvorca) dostávame náhodné reálne číslo z intervalu $\langle 0, 2 \cdot r)$.

Túto metódu výpočtu obsahu zadaného útvaru vieme použiť nielen na kruhy, ale aj na ľubovoľné iné geometrické útvary, v ktorých dokážeme rýchlo určiť, či zadaný bod je vo vnútri, alebo mimo zadaného útvaru.

Pre obsah kruhu so zadaným polomerom r platí $S_{\odot} = \pi \cdot r^2$. Tento vzťah môžeme využiť na približný výpočet čísla π tak, že uvážime kruh s polomerom 1: $S_{\odot} = \pi \cdot r^2$, pre kruh s polomerom 1 dostávame $S_{\odot} = \pi \cdot 1^2 = \pi$. Vidíme, že obsah kruhu s polomerom 1 je presne číslo π, a ten môžeme približne vypočítať vyššie uvedeným postupom využívajúcim pravdepodobnosť.



$$\pi = 4.\sum_{k=0}^{\infty} \frac{(-1)^k}{2 \cdot k + 1} =$$
$$= \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \cdots$$



Obrázok 2.3: Výpočet čísla π s použitím pravdepodobnosti

Aktivita 1

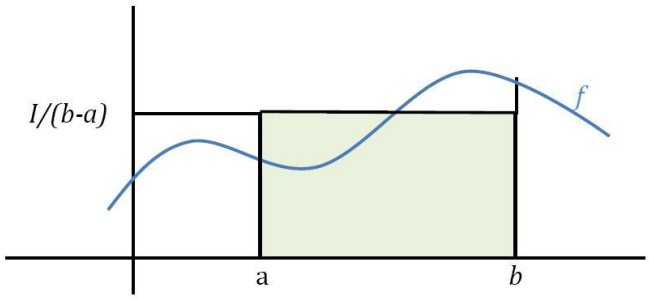
Pripravený softvér v aktivite „Výpočet obsahu kruhu“ umožňuje simuláciou náhodného generovania bodov vo štvorci vypočítať približný obsah kruhu s polomerom 1, t. j. hodnotu čísla π.

Ako súvisí presnosť výpočtu s celkovým počtom „kvapiek“ - náhodne generovaných bodov vo vnútri štvorca? Ako sa mení presnosť výpočtu pri opakovaní simulácie s rovnakým počtom generovaných bodov?

Spolu s predchádzajúcim príkladom na výpočet obsahu kruhu je výpočet plochy určitého integrálu ďalším príkladom numerického pravdepodobnostného algoritmu. Tieto metódy sa nazývajú niekedy aj „Monte Carlo“ metódy.

Názov Monte Carlo pochádza z článku, ktorý napísali Metropolis a Ulam v roku 1949.

Chceme vypočítať $I = \int_a^b f(x)\,\mathrm{d}x$, čo je plocha pod funkciou $f(x)$ na intervale $\langle a, b\rangle$. Hlavná myšlienka výpočtu plochy je znázornená na obrázku 2.4. Je zrejmé, že obdĺžnik s výškou $I/(b-a)$ má plochu rovnú I. Takže na intervale $\langle a, b\rangle$ musí byť priemerná výška krivky a obdĺžnika rovnaká a rovnajúca sa $I/(b-a)$. Pravdepodobnostný algoritmus výpočtu integrálu je založený na odhade priemernej výšky krivky na intervale $\langle a, b\rangle$, ktorú vynásobíme $b-a$. Priemernú výšku vypočítame tak, že náhodne vyberieme n bodov, v ktorých vypočítame hodnotu f, tie sčítame a nakoniec vydelíme s n.



Obrázok 2.4: Zelenou je vyfarbená plocha, ktorá sa rovná veľkosťou ploche pod krivkou na intervale $\langle a, b\rangle$.


```
type
  TFunc = function(x : Real) : Real;

function IntegrovanieMonteCarlo(f : TFunc;
                                n : Integer; a, b : Real);
var
  Sucet : Real;
  i : Integer;
begin
  Sucet := 0;
  for i := 1 to n do
    Sucet := Sucet + f(Random * (b - a));
  Result := (b - a)*(Sucet / n);
end;
```

Prirodzená otázka je, aké musí byť n, aby sme vypočítali I s presnosťou na p desatinných miest? Bohužiaľ, každá presná cifra výsledku vyžaduje 100-krát väčšiu prácu počítača. Deterministické metódy integrovania sú predsa oveľa efektívnejšie! Keby sme body, v ktorých počítame hodnotu funkcie f, nevyberali náhodne, ale systematicky, napríklad $a + i \cdot \Delta$, postupne pre $i = 0,1,\ldots,n-1$ a $\Delta = (b-a)/n$, dostali by sme presnú hodnotu I pre oveľa menšie n. Nevýhodou každej deterministickej metódy ale je, že pre ňu existuje nejaký príklad „nevhodnej" funkcie, ktorá ju prekabáti a vypočítaná hodnota nebude správna. Pri pravdepodobnostnom algoritme takáto nevhodná funkcia neexistuje, ale na druhej strane so zanedbateľne malou pravdepodobnosťou môže vypočítať zlý výsledok aj pre obyčajnú funkciu.

Výhoda pravdepodobnostného algoritmu sa ukáže, ak potrebujeme vypočítať viacnásobný integrál. V prípade deterministického algoritmu používame systematický spôsob vyberania bodov, v ktorých počítame hodnotu funkcie. Keď máme napríklad 100 bodov na určenie jednoduchého integrálu, aby sme dosiahli rovnakú presnosť, na určenie dvojitého integrálu budeme potrebovať takmer isto 100×100 bodov. Pri metóde integrovania Monte Carlo nezáleží veľmi na tom, koľkorozmerný je integrál. Samozrejme, aj tu rastie počet potrebných operácií, ale dimenzia integrálu nemá taký veľký vplyv ako pri deterministickej metóde. Od rozmeru 4 sa v praxi používajú na integrovanie rôzne varianty metódy Monte Carlo, lebo nepoznáme žiadnu porovnateľne efektívnu deterministickú metódu.

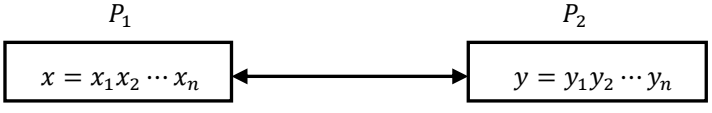
Aktivita 2

Pripravený softvér v aktivite „Výpočet určitého integrálu" realizuje integrovanie funkcie `sin` metódou Monte Carlo, t. j. na základe pravdepodobnostného výpočtu priemernej hodnoty bodov na zadanom intervale. Preskúmajte presnosť uvedenej metódy. Funkciu, ktorá je integrovaná, je možné zmeniť v zdrojovom kóde programu vo funkcii `Funkcia`. Preskúmajte aj zdrojový kód a prediskutujte jeho implementačnú náročnosť.

3. Testovanie zhody

Nemusí ísť len o databázy. Napríklad môžeme mať dva súbory umiestnené na rôznych počítačoch a potrebujeme overiť, či sú tieto súbory rovnaké.

Uvažujme nasledujúcu situáciu: majme dva od seba vzdialené počítače P_1 a P_2 (Obrázok 2.5), napríklad z bezpečnostných dôvodov. V oboch sú tie isté údaje, napríklad nejaká veľká databáza. Všetky operácie sa súčasne vykonávajú v oboch databázach. Po nejakom čase potrebujeme zistiť, či sú obe databázy naozaj identické.



Obrázok 2.5: Dva počítače s dvomi databázami x a y.

V nasledujúcom texte ukážeme, ako nám náhoda pomôže neuveriteľným spôsobom zmenšiť množstvo údajov, ktoré musíme vzájomne porovnať a súčasne môžeme dosiahnuť ľubovoľne vysokú mieru istoty, že ak údaje sú rôzne, tak správne zistíme, že sú rôzne. V prípade, že sú zhodné, dosiahneme dokonca stopercentnú istotu, že správne povieme, že sú zhodné.

Chceme navrhnúť algoritmus, na základe ktorého určíme, či P_1 a P_2 obsahujú rovnaké x a y. Takýto postup sa nazýva *komunikačný protokol* a mierou jeho efektívnosti bude, koľko informácií si musia vzájomne P_1 a P_2 vymeniť. Množstvo informácií budeme merať počtom bitov. Predpokladajme, že počet bitov n je veľký, napr. 10^{16}. Klasickým deterministickým spôsobom musíme vždy medzi P_1 a P_2 vymeniť všetkých n bitov (dá sa to aj dokázať). Nehľadiac na to, že pri takom množstve chýb môže nastať prenosová chyba, trvá to nezanedbateľne dlhú dobu, a to aj vtedy, keď použijeme prenosovú rýchlosť 1Gb/s. Už ani nehovoriac o počte prípadných DVD nosičov, keby sme údaje chceli preniesť takýmto spôsobom.

Gb je gigabit.

Označme $\text{Číslo}(x)$ prirodzené číslo, ktorého zápis v dvojkovej sústave je postupnosť bitov x, t. j. $\text{Číslo}(x) = \sum_{i=1}^{n} 2^{n-i} x_i$. Ukážeme si, že riešením určenia zhody bude nasledujúci pravdepodobnostný protokol na určenie zhody:

Počítač P_1 má postupnosť bitov x dĺžky n, $x = x_1 x_2 \cdots x_n$ a počítač P_2 má postupnosť bitov y dĺžky n, $y = y_1 y_2 \cdots y_n$.

1. P_1 si. zvolí náhodne z intervalu $\langle 2, n^2 \rangle$ prvočíslo p. Všetky prvočísla majú rovnakú šancu, aby boli vybrané.

2. P_1 vypočíta $s = \text{Číslo}(x) \bmod p$ a pošle P_2 binárnu reprezentáciu čísiel s a p.

3. P_2 prečíta s a p a vypočíta $q = \text{Číslo}(y) \bmod p$.

   Ak platí, že $q \neq s$, P_2 dá výsledok „x **sa nerovná** y“.
   Ak platí, že $q = s$, P_2 dá výsledok „x **sa rovná** y“.

Všimnime si, koľko bitov informácií si musia P_1 a P_2 vymeniť. Keď si uvedomíme, že $s < p < n^2$, je zrejmé, že celková dĺžka správ v bode 2 je najviac

$$2 \cdot \lceil \log_2 n^2 \rceil = 4 \cdot \lceil \log_2 n \rceil.$$

Pomôcka: na zápis čísla n potrebujeme $\lfloor \log_2 n \rfloor + 1$ bitov, teda $\lceil \log_2 n \rceil$ bitov bude na zápis istotne stačiť.

Teda pre našu databázu údajov veľkosti $n = 10^{16}$ dostaneme $4 \cdot 16 \cdot \lceil \log_2 10 \rceil = 256$ bitov (8 bajtov). To je neporovnateľne menej než 10^{16}!

Algoritmus využíva jednoduchú myšlienku. Namiesto toho, aby sme poslali všetky údaje z P_1, t. j. všetkých n bitov, pošleme len akýsi podstatne kratší odtlačok obsahu týchto údajov dĺžky $2 \cdot \lceil \log_2 n \rceil$ bitov. Tento odtlačok sa vytvára v kroku 2 použitím náhodného prvočísla, vygenerovaného v kroku 1. V kroku 2 môžeme vidieť, že to, aký odtlačok sa pre danú postupnosť bitov na P_1 vytvorí, záleží od zvoleného prvočísla. Preto v kroku 2 posielame v správe okrem odtlačku aj toto náhodne zvolené prvočíslo, čo znamená ďalších $2 \cdot \lceil \log_2 n \rceil$ bitov v správe. Celkovo má preto správa $4 \cdot \lceil \log_2 n \rceil$ bitov. Príjemca, počítač P_2, prijme správu. Vďaka tomu sa dozvie odtlačok údajov uložených na P_1 a tiež prvočíslo, použitím ktorého vznikol odtlačok na P_1. S využitím toho istého prvočísla a tým istým spôsobom ako P_1 si aj P_2 vypočíta odtlačok údajov, ktoré má u seba. Tento odtlačok potom porovná s odtlačkom od P_1. Ak sú odtlačky rovnaké, P_2 prehlási, že údaje uložené na oboch počítačoch sú rovnaké a naopak, ak nie sú, oznámi, že nie sú rovnaké ani údaje.

Je takýto algoritmus vôbec správny?

Úloha 1

Zoberme si päťbitové postupnosti napr. $x = 01011$ a $y = 11100$ ($\text{Číslo}(x) = 11$, $\text{Číslo}(y) = 28$). Vykonajte uvedený algoritmus na určenie zhody. Pomôcka: $n = 5$.

Úloha 2	Predpokladajte, že si algoritmus zvolil prvočíslo $p = 5$ alebo $p = 17$. Vykonajte algoritmus v oboch prípadoch. Čo ste zistili?
Úloha 3	V skupine si rozdeľte prvočísla v intervale od 2 po 25 a pre každé z nich zistite, či dá algoritmus správny, alebo nesprávny výsledok.

Takže uvedený algoritmus sa môže pomýliť. Mýli sa často?

V Úlohe 3 sme videli, že pri niektorých voľbách prvočísiel dal algoritmus správne výsledky a pri iných odpovedal nesprávne. V ďalšom sa budeme usilovať určiť pravdepodobnosť, že algoritmus dá chybný výsledok. Pre každý vstup x a y rozdelíme množinu prvočísiel na dve podmnožiny, ktoré označíme *prvočísla dobré pre* (x, y), kde algoritmus dá správny výsledok, a *prvočísla zlé pre* (x, y), kde dá chybný výsledok. Vzhľadom na to, že je rovnako pravdepodobné, aby si algoritmus zvolil ľubovoľné spomedzi prvočísiel, je pravdepodobnosť chybnej odpovede

$$\frac{\text{počet zlých prvočísiel pre } (x, y)}{\text{počet všetkých prvočísiel } \leq n^2}.$$

Skúsme odhadnúť čitateľ aj menovateľ tohto zlomku. Odhadnúť menovateľ je ľahšie, lebo známa veta o prvočíslach hovorí, že

$$\text{počet prvočísiel menších alebo rovných než } m \text{ je väčší než } \frac{m}{\ln m} \text{ pre } m > 67.$$

V našom prípade $m = n^2$, takže počet prvočísiel $\leq n^2$ je väčší než $n^2/(2 \ln n)$ pre $n \geq 9$. A čo čitateľ? Chceme ukázať, že pre každý vstup (x, y) je počet zlých prvočísiel najviac $n - 1$. V takom prípade vieme odhadnúť zlomok vyjadrujúci pravdepodobnosť pomýlenia sa:

$$\frac{\text{počet zlých prvočísiel pre } (x, y)}{\text{počet všetkých prvočísiel } \leq n^2} \leq \frac{n-1}{n^2/2\ln n} \leq \frac{2 \ln n}{n}.$$

Takže pravdepodobnosť chybného výsledku pre vstup (x, y) je najviac

$$\frac{2 \ln n}{n}.$$

Pre $n = 10^{16}$ to je menej než 10^{-14}. V prípade, že sa nám takáto pravdepodobnosť zdá ešte stále privysoká, môžeme ju ešte znížiť opakovaným spustením rovnakého algoritmu. Každý beh algoritmu je nezávislý od predchádzajúceho. Všimnime si, že keď dostaneme výsledok: „x sa nerovná y“, je to určite tak a $x \neq y$. Chyba môže nastať iba v prípade odpovede: „x sa rovná y“. Pravdepodobnosť, že by sa algoritmus pomýlil povedzme k-krát za sebou, je preto

$$\left(\frac{2 \ln n}{n}\right)^k,$$

čo je pre $k = 10$ menej než 10^{-140}, a to je už prakticky zanedbateľné. Je oveľa pravdepodobnejšie, že sa vyskytne chyba výpočtu v dôsledku chyby hardvéru.

Ešte nám ostalo ukázať, že počet zlých prvočísiel pre každý vstup (x, y) je naozaj najviac $n - 1$.

Keď $s = X \bmod p = Y \bmod p$, potom $X = k_1 p + s$ a $Y = k_2 p + s$ a $X - Y = (k_1 - k_2)p$

Keď je $x = y$, je každé prvočíslo dobré – pre ľubovoľné prvočíslo p platí, že $\text{Číslo}(x) \bmod p = \text{Číslo}(y) \bmod p$.

V prípade, že je $x \neq y$, dostaneme zlú odpoveď „x sa rovná y“ len keď $\text{Číslo}(x) \bmod p = \text{Číslo}(y) \bmod p$, čo znamená, že $p \mid \big(\text{Číslo}(x) - \text{Číslo}(y)\big)$. Takže

prvočíslo p je zlé pre (x, y) práve vtedy, keď delí hodnotu $z = \text{Číslo}(x) - \text{Číslo}(y)$. Odhadneme veľkosť z. Pripomeňme si, že x aj y sú postupnosti bitov, ktorých dĺžka je n, takže aj čísla, ktoré reprezentujú sú menšie ako 2^n, teda platí aj $z < 2^n$. Číslo z môžeme napísať rozložené na prvočísla:

$$z = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k},$$

kde $p_1 < p_2 < \cdots < p_k$ a $e_1, e_2, \cdots e_k$ sú nezáporné celé čísla. Je dôležité si uvedomiť, že ak nejaké prvočíslo delí číslo z, potom sa musí nachádzať v jeho rozklade na súčin prvočísiel. Inými slovami, pri vyššie uvedenom rozklade má číslo z práve k prvočíselných deliteľov, ktoré pre nás predstavujú zlé prvočísla. Najmenšie prvočíslo je 2, t.j. $p_i \geq 2$ pre všetky i od 1 po k a preto:

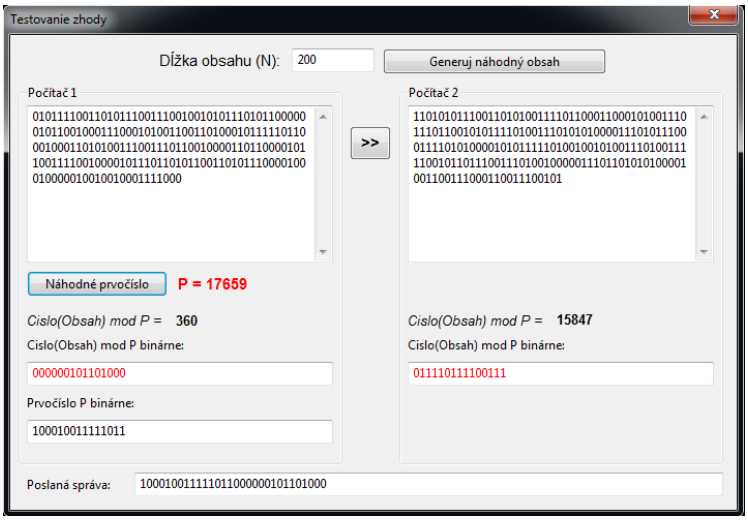
$$z = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k} \geq p_1 p_2 p_3 \ldots p_k \geq 2^k.$$

Takže $2^k \leq z$, ale už vieme, že platí aj $z < 2^n$, teda $2^k < 2^n$. To platí, keď $k \leq n - 1$, čo sme chceli ukázať.

Aktivita 1.

Pripravený softvér v aktivite „Testovanie zhody“ umožňuje prakticky si vyskúšať, ako protokol funguje. Tlačidlom „Generuj náhodný obsah“ si môžeme vytvoriť náhodnú postupnosť bitov zadanej dĺžky na „oboch počítačoch“. Ten môžeme podľa potreby neskôr upravovať. Tlačidlom „>>“ vieme binárny obsah prekopírovať „z prvého počítača na druhý“ (aby sme mohli skúšať prípad, kedy je obsah na oboch počítačoch rovnaký). Kliknutím na tlačidlo „Náhodné prvočíslo“ sa vyberie náhodné prvočíslo z intervalu 2 až n^2 a vypočítajú sa odtlačky obsahov.

Upozornenie: V algoritme potrebujeme generovať náhodné prvočíslo z intervalu 2 až n^2. Aby táto operácia netrvala dlho, v pripravenom programe sú prvočísla uložené v súbore `prvocisla.txt`. Ak takýto súbor nemáme alebo je v ňom nagenerovaných málo prvočísiel, dogenerovať si ich môžeme tlačidlom „Generuj prvočísla“ v úvodnom okne programu.



Obrázok 2.6: Simulácia algoritmu testovania zhody

Z implementačného hľadiska je v algoritme zaujímavé ešte jedno miesto, a to výpočet $\text{Číslo}(x) \bmod p$, kde $\text{Číslo}(x) = \sum_{i=1}^{n} 2^{n-i} x_i$. Ako by ste tento výpočet naprogramovali mysliac na to, že najväčšie číslo, ktoré premenná typu *Integer* dokáže uchovať je 2^{31}?

Efektívna implementácia spočíva vo využití vzťahu:

$$(2.a + b) \bmod p = (2.(a \bmod p) + b) \bmod p.$$

Namiesto toho, aby sme najprv vypočítali $\text{Číslo}(x)$ a potom $\text{Číslo}(x) \bmod p$, môžeme rovno počítať $\text{Číslo}(x) \bmod p$. V nasledujúcom zdrojovom kóde predpokladáme, že jednotlivé bity binárnej postupnosti x sú uložené v poli X.

```
CisloModP := 0;
for I := 1 to N do
  CisloModP := (2 * CisloModP + X[I]) mod P;
```

4. Simulovanie kocky mincou

V predchádzajúcej podkapitole sme si ukázali, ako s využitím pravdepodobnosti možno výrazne ušetriť na veľkosti prenášaných údajov. To, čo sme však za to museli „zaplatiť“, bolo, že tu bola istá pravdepodobnosť, že algoritmus nám dá nesprávny výsledok. Teraz si ukážeme príklad pravdepodobnostného algoritmu, ktorý nám dá správny výsledok stále. Avšak od pravdepodobnosti (náhody) bude tentoraz pre zmenu závisieť čas vykonávania algoritmu.

Aktivita 1.	Predstavme si situáciu, že potrebujeme zrealizovať hod spravodlivou mincou, no žiadnu nemáme po ruke. Našli sme len spravodlivú hraciu kocku. Ako môžeme pomocou hracej kocky odsimulovať hod mincou?

Asi ľahko prídeme na to, že stačí ako rub brať padnutie párneho čísla a ako líce padnutie nepárneho čísla. Obe udalosti majú rovnakú pravdepodobnosť, a to $1/2$. Teda v jazyku Pascal `Random(2)` vieme vypočítať ako `Random(6) mod 2`. Čo však v opačnej situácii? Chceme si zahrať „Človeče“, no nemáme po ruke žiadnu hraciu kocku. Máme však spravodlivú mincu. Je možné pomocou mince odsimulovať hraciu kocku? Problém môžeme preformulovať aj do jazyka Pascal takto: Je možné pomocou funkcie `Random(2)` naprogramovať funkciu `Random(6)`?

Aktivita 2.	Diskutujte, ako by išlo pomocou hádzania mince odsimulovať hod hracou kockou. Pozor, nezabudnite na to, že treba, aby každé z čísiel 1 až 6 malo rovnakú pravdepodobnosť výskytu.

Ľahko prídeme na to, že to, čo potrebujeme, je rovnomerne náhodne vrátiť číslo od 1 po 6 (resp. od 0 po 5). Ak by sme hádzali mincou 3-krát, môžeme dostať celkom $8 = 2^3$ rôznych výsledkov, každý s rovnakou pravdepodobnosťou $1/8$. Očíslujme si ich od 1 po 8. Kocka však dáva len 6 výsledkov. Náš pravdepodobnostný algoritmus preto bude fungovať nasledujúcim spôsobom: Ak nám po zrealizovaní trojice hodov (trojhodu) vyjde ako celkový výsledok číslo od 1 po 6, tak túto hodnotu vrátime ako výsledok. Ak celkový výsledok trojhodu bude 7 alebo 8, budeme algoritmus opakovať, t. j. uskutočníme ďalšiu trojicu hodov.

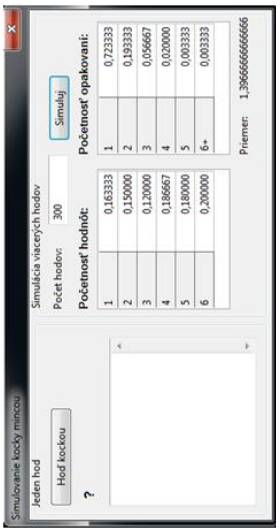
1. hod	2. hod	3. hod	Akcia
R	R	R	**Vráť 1**
R	R	L	**Vráť 2**
R	L	R	**Vráť 3**
R	L	L	**Vráť 4**
L	R	R	**Vráť 5**
L	R	L	**Vráť 6**
L	L	R	**Opakuj**
L	L	L	**Opakuj**

Obrázok 2.7: Schematické znázornenie algoritmu

Všimnime si, že tento algoritmu má naozaj požadovanú vlastnosť, a to, že každé z čísiel od 1 po 6 má rovnakú pravdepodobnosť výskytu. To preto, že každý z výsledkov „trojhodu“ je rovnako pravdepodobný. Inými slovami, tento algoritmus vždy vráti taký výsledok, ako má.

```
function SimulujHod3Mincami: Integer;
var
  Minca1, Minca2, Minca3: Integer;
begin
  repeat
    Minca1 := Random(2);
    Minca2 := Random(2);
    Minca3 := Random(2);
    // Vypočítame výsledok trojhodu – číslo od 0 po 7
    Result := Minca1 * 4 + Minca2 * 2 + Minca3;
  until Result <= 5;

  // Upravíme z 0..5 na 1..6
  Inc(Result);
end;
```



Ako sme už naznačili v úvode, od pravdepodobnosti teraz nezávisí správnosť výsledku, ale čas výpočtu. Totiž z 8 elementárnych prípadov je v 2 prípadoch akciou algoritmu ďalšie simulovanie 3 hodov mincou. Aký je však očakávaný priemerný počet opakovaní „trojhodov“, resp. aký je očakávaný počet vykonaní `repeat`-cyklu vo funkcii `SimulujHod3Mincami`?

Aktivita 3

Pripravený softvér v aktivite „Kocka pomocou mince“ umožňuje skúmať algoritmus na simulovanie hodu kockou pomocou mince. Tlačidlom „Hoď kockou“ spustíme simulovanie algoritmu. Pomocou simulácie viacerých hodov môžeme preskúmať správanie sa algoritmu pri viacerých spusteniach. V tabuľkách nájdeme relatívne početnosti hodnôt, ktoré boli výsledkom funkcie `SimulujHod3Mincami` a tiež relatívne početnosti počtu opakovaní `repeat`-cyklu pri jednom vykonaní funkcie `SimulujHod3Mincami`. Položka „Priemer“ nám hovorí, aký bol priemerný počet opakovaní `repeat`-cyklu.

Aké výsledky budete pozorovať?

Vypočítajme očakávaný počet opakovaní. Pravdepodobnosť toho, že počas jednej iterácie cyklu vygenerujeme hodnotu, ktorá bude mať za následok ukončenie výpočtu, je $6/8$. Označme si túto udalosť ako A, t.j. $P(A) = 6/8$. Doplnková udalosť A^c k udalosti A je udalosť, že výsledok je taký, že budeme musieť cyklus, a teda hádzanie mincami zopakovať minimálne ešte raz. Vieme, že platí $P(A^c) = 2/8$. Aká je pravdepodobnosť toho, že funkcia `SimulujHod3Mincami` bude potrebovať presne k opakovaní cyklu? Ak bolo potrebných k opakovaní cyklu, znamená to, že $(k-1)$-krát nastala udalosť A^c a potom raz nastala udalosť A. Pravdepodobnosť k opakovaní cyklu je teda $P(A^c)^{k-1} \cdot P(A) = \frac{3}{4} \cdot \left(\frac{1}{4}\right)^{k-1}$. Očakávaný počet opakovaní (akoby vážený priemer z nekonečného počtu čísiel) možno vypočítať takto:

$$\sum_{k=1}^{\infty} k \frac{3}{4}\left(\frac{1}{4}\right)^{k-1} = \frac{3}{4}\sum_{k=1}^{\infty} k \left(\frac{1}{4}\right)^{k-1} = \frac{3}{4}\sum_{k=1}^{\infty} 4 \cdot k \left(\frac{1}{4}\right)^{k} = 3\sum_{k=1}^{\infty} k \left(\frac{1}{4}\right)^{k} = 3 \cdot \frac{4}{9} = \frac{4}{3}.$$

Využili sme vzťah $\sum_{k=1}^{\infty} k \cdot x^k = \frac{x}{(1-x)^2}$, ktorý platí pre $|x| < 1$.

Očakávaný počet vykonaní cyklu pri jednej simulácii je $4/3$. To môžeme interpretovať tak, že čím viac simulácií spravíme, tým viac sa bude priemerný počet „opakovaní“ na jednu simuláciu blížiť k $4/3$. Z praktického hľadiska je to samozrejme veľmi dobrý výsledok, keďže nám hovorí, že môžeme očakávať, že takéto simulovanie nás nebude stáť príliš veľa práce „navyše“.

5. Las Vegas a Monte Carlo

Predchádzajúce príklady simulácie hádzania kockou pomocou mince a určovanie zhody predstavujú dva rôzne typy pravdepodobnostných algoritmov. Simulácia hádzania kockou pomocou mince vedie vždy k správnemu výsledku – číslu, ktoré padne na kocke. Počas svojej práce sa snaží využiť náhodné rozhodnutia na to, aby sa čo najrýchlejšie dostal k správnemu riešeniu. Jednotlivé výpočty algoritmu (s tými istými vstupnými údajmi) sa môžu od seba líšiť iba časom potrebným na výpočet. Takéto algoritmy sa nazývajú *Las Vegas algoritmy*.

Na druhej strane algoritmus určovania zhody, výpočtu plochy, či určovania prvočíselnosti (v nasledujúcom odseku) sa môžu pomýliť, dať nesprávny výsledok. Takýto algoritmus voláme *Monte Carlo*. Jeho užitočná vlastnosť je, že pravdepodobnosť chybnej odpovede sa za cenu času potrebného na výpočet dá ľubovoľne zmenšiť jeho opakovaným vykonávaním. Pre rozhodovacie problémy (ich výsledok je odpoveď ÁNO alebo NIE) sú dva druhy Monte Carlo algoritmov, podľa toho, kedy môžu odpovedať nesprávne. Monte Carlo s obojstrannou chybou sú také, keď je nenulová pravdepodobnosť chybnej odpovede aj v prípade ÁNO, aj NIE. Jednostranná chyba je, keď je nulová pravdepodobnosť chybnej odpovede aspoň pri jednej z odpovedí ÁNO alebo NIE.

Pri algoritme na určovanie zhody a podobne aj pri testovaní prvočíselnosti je určitá pravdepodobnosť p, že dostaneme nesprávny výsledok. Každý výpočet algoritmu s tými istými vstupnými údajmi je nezavislý. Pravdepodobnosť toho, že by nás v m po sebe idúcich nezávislých pokusoch algoritmus opakovane klamal je p^m. Keď uvážime, že $0 < p < 1$, je zrejmé, že pravdepodobnosť opakovaného chybného výsledku sa so zvyšujúcim počtom pokusov blíži k nule. V praxi stačí, keď klesne pod nami zvolenú hranicu.

Hovoríme, že algoritmus Monte Carlo, resp. Las Vegas je efektívny, keď je očakávaný čas jeho behu ohraničený polynomickou funkciou vzhľadom na veľkosť vstupu.

6. Testovanie prvočíselnosti

Zisťovanie prvočíselnosti v dnešnej dobe nie je len matematická zábavka, ale prvočísla hrajú dôležitú úlohu napríklad v kryptografii.

Millerov test prvočíselnosti

Pierre Fermat (1601 – 1665) bol francúzsky právnik a matematik – samouk.

Pripomeňme si *malú Fermatovu vetu* :

Nech p je prvočíslo a a ľubovolné číslo, potom $a^P \equiv a\ (mod\ p)$.

$b \equiv a \pmod n$ znamená, že $n\,|(b-a)$. Alebo ešte inak: b aj a majú rovnaký zvyšok po delení číslom n.

Keď chceme overiť, či je p prvočíslo, nemôžeme využiť predchádzajúcu vetu, lebo nanešťastie neplatí opačná implikácia. Netreba sa však vzdávať. Môžeme využiť, že ak nejaké n je prvočíslo, musí byť $a^{n-1} \equiv 1 \pmod n$. Teda ak $a^{n-1} \not\equiv 1 \pmod n$, s istotou vieme, že n nie je prvočíslo.

Keď začneme s nepárnym prvočíslom n, podľa Fermatovej vety musí platiť, že $a^{n-1} \equiv 1 \pmod n$. Vieme, že $n-1$ je párne a môžeme vypočítať $x = a^{\frac{n-1}{2}}$. Pretože $x^2 \equiv 1 \pmod n$, musí byť $x \equiv \pm 1 \pmod n$. V prípade, že $x = 1$ a $\frac{n-1}{2}$ je párne, môžeme pokračovať rovnakým spôsobom a vyrátať $y = a^{\frac{n-1}{4}}$. Z rovnakých dôvodov ako predtým, pretože $y^2 \equiv 1 \pmod n$, musí byť aj $y \equiv \pm 1 \pmod n$. Takto môžeme pokračovať, pokým nedosiahneme hodnotu -1 alebo nepárny exponent (môžu nastať aj oba prípady súčasne). Intuitívne je idea zrejmá, zložené číslo sa len ťažko zamaskuje za prvočíslo. Dostali sme test, ktorý navrhol Gary Lee Miller v roku 1976.

V niektorých prípadoch sa hodí viac Las Vegas algoritmus, inokedy zasa Monte Carlo. Nedá sa jednoznačne vo všeobecnosti povedať, ktorý je lepší.

Millerov test prvočíselnosti vzhľadom na a

Nech n je nepárne prvočíslo a nech $\text{nsd}(a,n) = 1$ a $1 < a < n$.

```
begin
  k := n - 1;
  r := a^k mod n;
  while (r = 1) and (k mod 2 = 0) do
  begin
    k := k div 2;
    r := a^k mod n;
  end;
  if (r = 1) or (r = n - 1) then
    Prešiel
  else
    Neprešiel
end
```

Zložené číslo, ktoré prejde testom rovnako, ako by ním prešlo aj prvočíslo, budeme nazývať *silné pseudoprvočíslo vzhľadom na* a. Je zrejmé, že keď je n prvočíslo, nemôže sa stať, že neprejde Millerovým testom (uvedomme si, že $x \equiv \pm 1 \pmod n$ znamená, že $x = 1$ alebo $x = n - 1$), čo môžeme sformulovať do nasledujúceho tvrdenia:

Predpokladáme, že $x < n$

Ak prirodzené číslo neprejde Millerovým testom, je zložené.

Pravdepodobnostné testovanie prvočíselnosti – Rabinov test

Keď sa pokúsime nájsť najmenšie n, ktoré je súčasne silným pseudoprvočíslom vzhľadom na 3, 5 a 7, získame podozrenie, že výskyt čísiel, ktoré sú silnými pseudoprvočíslami súčasne vzhľadom na väčší počet hodnôt, je veľmi zriedkavý. Ozaj je pravda, že čím väčším počtom Millerových testov číslo prejde, tým je väčšia šanca, že je prvočíslom. Platí nasledujúca veta:

Rabinov test využíva techniku zvanú „prebytok svedkov“. Na základe matematických argumentov vieme, že báz, ktoré by odhalili zložené číslo, je relatívne veľa, avšak netušíme, ktoré to sú.

Nech n je nepárne zložené číslo. Potom n spĺňa Millerov test najviac pre $(n-1)/4$ báz b, $1 \leq b \leq n-1$.

Inak povedané, n nemôže byť silným pseudoprvočíslom vzhľadom na všetky hodnoty, ktoré prichádzajú do úvahy. Na to, aby sme sa presvedčili, či je číslo n zložené, by nebolo praktické skúšať všetkých $(n-1)/4$ báz. Keď n prejde Millerovým testom vzhľadom na b, je pravdepodobnosť toho, že sme si zvolili „nevhodnú“ hodnotu, pri ktorej sa to stane, rovná $1/4$. Keď n prejde k Millerovými testami vzhľadom na rôzne hodnoty, pričom predpokladáme, že boli vybrané náhodne, tak pravdepodobnosť, že je n zložené, je $1/4^k$. Istota, že n je prvočíslo, rastie s rastúcim k. Táto metóda sa nazýva *Rabinov pravdepodobnostný test*, lebo ho vymyslel Michael Oser Rabin. Miller ukázal, že za istých špeciálnych predpokladov (že platí rozšírená Riemannova hypotéza) zložené n neprejde testom pre hodnotu menšiu než $2(\log n)^2$, čo je malé číslo aj pre veľmi veľké n.

Aktivita 1	Pripravený softvér v aktivite „Test prvočíselnosti“ umožňuje overiť, či zadané číslo prejde Millerovým testom prvočíselnosti vzhľadom na zadanú hodnotu. Vyskúšajte, ako tento test funguje. Prvočísla si môžete vygenerovať do súboru `prvocisla.txt` pomocou tlačidla „Generuj prvočísla“ v úvodnom okne programu. V aktivite „Test prvočíselnosti“ môžeme vyskúšať aj pravdepodobnostný Millerov-Rabinov test. Parametrom tohto testu je počet náhodne zvolených hodnôt, vzhľadom na ktoré sa majú otestovať.
Aktivita 2	Pozrite si zdrojový kód aktivity „Test prvočíselnosti“ a diskutujte o implementačnej náročnosti tohto testu.

Pri efektívnej implementácii testu je zaujímavým miestom výpočet $a^k \bmod n$.

Úloha 1	Napíšte funkciu, ktoré pre zadané a, k a n výpočíta hodnotu $a^k \bmod n$. Pokúste sa o to, aby procedúra potrebovala na výpočet len približne $\log_2 n$ operácií násobenia. Ako rýchlo viete vynásobiť b-bitové číslo c-bitovým?

7. Náhodné čísla v počítačoch – pseudonáhodnosť

Všetky algoritmy predstavené v tejto kapitole sú postavené na využití rovnomerného generátora náhodných čísiel - zdroja náhodnosti. Z iných modulov však dobre vieme, že počítače sú deterministické zariadenia, ktoré len presne krok za krokom vykonávajú zadanú postupnosť inštrukcií - ich fungovanie je presne určené programom. Odkiaľ sa teda náhodné čísla (vrátené trebárs funkciou `Random`) berú v počítačoch?

Aktivita 1	Diskutujte o tom, ako by sa mohli v prísne deterministických zariadeniach, ako sú počítače, generovať náhodné čísla.

Na webovej stránke www.random.org si môžeme nechať vygenerovať náhodné číslo na základe atmosférického šumu.

Prehľad hardvérových generátorov náhodných čísel založených na fyzikálnych procesoch nájdete na stránke: http://en.wikipedia.org/wiki/Comparison_of_hardware_random_number_generators.

Ukazuje sa, že efektívne generovanie skutočne náhodných čísiel je v počítačoch naozaj problematické. Namiesto náhodných čísiel sa v počítačoch využívajú takzvané *pseudonáhodné čísla*. Pseudonáhodné čísla (presnejšie postupnosť pseudonáhodných čísiel) sú generované deterministickým algoritmom, ktorý je inicializovaný nejakým inicializačným číslom (v angličtine označovaným ako *seed* - semienko, zárodok). Tento algoritmus, nazývaný *generátor pseudonáhodných čísiel*, potom pri každej požiadavke o vygenerovanie náhodného čísla na základe inicializačného čísla a už vygenerovaných čísiel vypočíta ďalšie číslo v pseudonáhodnej postupnosti a vráti ho. V najjednoduchších generátoroch sa i-te číslo x_i pseudonáhodnej postupnosti vypočíta ako $x_i = f(x_{i-1})$, kde f je nejaká vhodná funkcia a x_0 je inicializačné číslo pre generovanú postupnosť. V generátoroch pseudonáhodných čísiel je tak celá postupnosť vygenerovaných čísiel determinovaná iba hodnotou inicializačného čísla. Inými slovami, ak použijeme na dvoch počítačoch rovnaké inicializačné čísla, pseudonáhodný generátor nám bude pseudonáhodné čísla generovať v rovnakej postupnosti.

Aktivita 2	Napíšte jednoduchý program v Pascale a overte, že ak nezavoláte procedúru `Randomize`, funkcia `Random` bude generovať vždy rovnakú postupnosť náhodných čísiel. Na overenie môžete použiť aj pripravený softvér tak, že po jeho spustení nekliknete na tlačidlo „Randomize“. Bude generovaná postupnosť rovnaká aj na rôznych počítačoch?

Keďže inicializačné číslo pseudonáhodnej postupnosti určuje celú generovanú postupnosť, malo by byť nejakým spôsobom čo „najnáhodnejšie“. Zvyčajne sa preto ako inicializačné číslo používa aktuálny systémový čas alebo číslo „vyrobené“ operačným systémom na základe zaznamenávania udalostí v systéme (napr. pohyby myšou alebo iná používateľská aktivita). Ako sme sa presvedčili v predchádzajúcej aktivite, zakaždým, keď v Pascale spustíme program, je generovaná rovnaká postupnosť „náhodných“ čísiel. My však už teraz vieme, že je to preto, že ide o pseudonáhodné čísla, ktorých generátor bol inicializovaný rovnakou inicializačnou hodnotou. Ak to chceme zmeniť, musíme zavolať procedúru `Randomize`, ktorou sa v Pascale inicializuje generátor pseudonáhodných čísiel podľa aktuálneho systémového času.

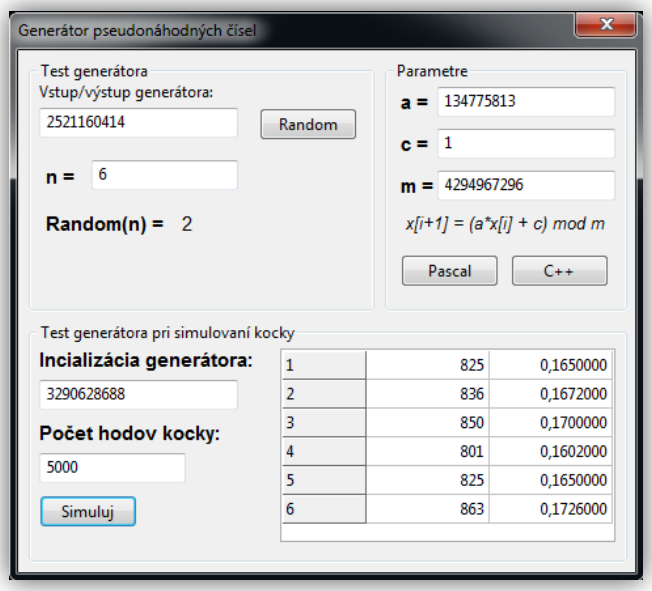
Aktivita 3	Vyskúšajte, ako sa zmení postupnosť generovaných pseudonáhodných čísiel po zavolaní procedúry `Randomize`.

V súčasnosti je známych veľa algoritmov na generovanie pseudonáhodných čísiel. Každý z nich má svoje výhody i nevýhody. Viaceré boli preskúmané štatisticko-matematickými metódami, ako „kvalitne náhodné“ sú generované čísla. Jeden z mnohých používaných algoritmov sú tzv. *lineárne kongruenčné generátory*. V tomto algoritme je i-te vygenerované číslo x_i pseudonáhodnej postupnosti vypočítané z predošlého čísla vzorcom $x_i = (a.x_{i-1} + c) \bmod m$, kde $0 < a < m$ a $0 \leq c < m$.

```
var
  Posledne: Integer = 0;

function Random(N: Integer): Integer;
begin
  Posledne := (A * Posledne + C) mod M;
  Result := Posledne mod N;
end;
```

Lineárne kongruenčné generátory sú použité napríklad v Pascale, kde sú na generovanie použité hodnoty $m = 2^{32}$, $a = 134775813$ a $c = 1$.



Obrázok 2.8: Okno aktivity „Pseudonáhodný generátor“

Aktivita 4.	Pripravený softvér v aktivite „Pseudonáhodný generátor“ umožňuje vyskúšať si, akým spôsobom lineárny kongruenčný generátor generuje pseudonáhodné čísla. Vpravo hore sa nastavujú parametre lineárneho kongruenčného generátora. V paneli umiestnenom vľavo hore je možné vyskúšať si jeden krok generátora, pričom ešte predtým musíme do editačného políčka zadať predchádzajúce číslo pseudonáhodnej postupnosti. V dolnej časti sa nachádza panel, v ktorom môžeme vyskúšať pseudonáhodný generátor pri simulovaní hádzania hracou kockou. Blížia sa relatívne početnosti jednotlivých vygenerovaných čísiel očakávanej pravdepodobnosti?

Čo sme sa naučili

Zoznámili sme sa s pravdepodobnostnými algoritmami typu Monte Carlo a Las Vegas. Ukázali sme si, ako nám náhoda môže pomáhať „proti“ zlým prípadom vstupov a dokážeme pomocou nej vytvoriť jednoduché a efektívne algoritmy a štruktúry údajov.

Literatúra a použité zdroje

[1] Motwani, R., Raghavan, P. (1995) Randomized Algorithms. Cambridge University Press. ISBN 0-521-47465-5

[2] Hromkovič, J. (2005) Design and Analysis of Randomized Algorithms. Springer, ISBN 3-540-23949-9

[3] Hromkovič, J. (2009) Algorithimc Adventures, from Knowledge to Magic, Springer, ISBN 978-3-540-85985-7

[4] Brassard, G., Bratley, P. (1996) Fundamentals of Algorithms, Prentice Hall. ISBN 0-13-335068-1

[5] Tijms, H. (2007) Understanding Probability, Chance Rules in Everyday Life. Cambridge University Press, ISBN 978-0-521-70172-3

[6] Giblin, P.: Primes and Programming, Introduction to Number theory with Computing, Cambridge University Press, 1993;

[7] Goodrich, M., Tamassia, R. Using Randomization in the Teaching of Data Structures and Algorithms, ACM SIGCSE '99, New Orleans

[8] Huraj, L.: Nebojme sa šifrovania. Metodicko-pedagogické centrum v Bratislave, Bratislava, 2002, ISBN 80-8052-160-3. http://www.fpv.umb.sk/~huraj/NebojmeSaSifrovania.pdf

[9] Hynek Bachratý, Marián Grendár, Katarína Bachratá (2010); Ako sa počíta pravdepodobnosť? Žilina, Banská Bystrica: Žilinská univerzita; Univerzita Mateja Bela, 2010. - 326 s. ISBN 978-80-554-0226-0

[10] Alfred Menezes, Paul van Oorschot and Scott Vanstone (1996) Handbook of Applied Cryptography. CRC Press, Inc.

[11] Bruce Schneier (1996) Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. Wiley.

[12] Douglas Stinson (2005) Cryptography Theory and Practice, Third Edition. CRC Press, Inc.

[13] Simon Singh (2000) The Code Book: The Secret History of Codes and Code-breaking. Fourth Estate Ltd.

[14] Zákon o elektronickom podpise. http://www.genpro.gov.sk/zakon-o-elektronickom-podpise/

Tento študijný materiál vznikol ako súčasť národného projektu Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika v rámci Aktivity „Vzdelávanie nekvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ“.

Autori © RNDr. Michal Winczer, PhD.
RNDr. František Galčík, PhD.
RNDr. Michal Forišek, PhD.

Názov Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Podnázov Kapitoly z informatiky 3

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti doc. RNDr. Stanislav Krajči, PhD.
RNDr. Jozef Jirásek, PhD.

Počet strán 36

Náklad 300 ks

Prvé vydanie, Bratislava 2011

Vydal Štátny pedagogický ústav, Pluhová 8, 830 00 Bratislava, v súčinnosti s Univerzitou Pavla Jozefa Šafárika v Košiciach, Univerzitou Komenského v Bratislave, Univerzitou Konštantína Filozofa v Nitre, Univerzitou Mateja Bela v Banskej Bystrici a Žilinskou univerzitou v Žiline

Vytlačil BRATIA SABOVCI, s r.o., Zvolen

ISBN 978-80-8118-092-7