UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY
A INFORMATIKY

Evidenčné číslo: 74b93af3-8dd5-43d9-b3f2-05523e0ba177

REKURENTNÉ POSTUPNOSTI

2011

András Varga

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY
A INFORMATIKY

REKURENTNÉ POSTUPNOSTI

bakalárska práca

Študijný program: Názov: Informatika
Študijný odbor: Číslo a názov: 921 Informatika
Školiace pracovisko: Katedra informatiky
Školiteľ: RNDr. Martin Sleziak, PhD.

Bratislava, 2011

András Varga

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: András Varga
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Rekurentné postupnosti

Cieľ: Popísať, implementovať a odvodiť zložitosť algoritmov na výpočet n-tého člena Fibonacciho postupnosti a iných príbuzných rekurentne zadaných postupností. Porovnať efektivitu rôznych algoritmov riešiacich tento problém.

Literatúra: E. Bach and J. Shallit. Algorithmic number theory, vol. 1.: Efficient algorithms. MIT Press, Cambridge, 1996.
D. Takahashi. A fast algorithm for computing Fibonacci numbers. Information Processing Letters, 75:243–246, 2000.

Vedúci: RNDr. Martin Sleziak, PhD.
Katedra: FMFI.KAGDM - Katedra algebry, geometrie a didaktiky matematiky

Dátum zadania: 25.10.2010

Dátum schválenia: 02.11.2010

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

Varga
.....
študent

Sleziak
.....
Vedúci

Čestne prehlasujem, že bakalársku prácu som vypracoval samostatne s použitím uvedenej literatúry a pod dohľadom môjho vedúceho práce.

. . . . . .

Chcel by som poďakovať môjmu vedúcemu RNDr. Martinovi Sleziakovi, PhD. za jeho pomoc, za inšpiratívne rady a za dohľad nad mojou činnosťou.

Abstrakt

Hlavným cieľom práce je vytvorenie, implementácia a dokázanie korektnosti rýchlych algoritmov pracujúcich na rekurentných postupnostiach druhého rádu. Základným problémom je výpočet n-tého člena takýchto postupnosti, potom zistenie či dané číslo je členom danej postupnosti. Pri skúmaní týchto problémov sme narazili na problém umocňovania matíc. V práci sa teda popisuje aj algoritmus na riešenie problému umocňovania matíc, jeho uvedenie pomôže jednoduchšie pochopiť celú tematiku. Uvedené problémy sa dajú riešiť v čase $O(n)$. V tejto práci sme sa snažili vytvoriť také algoritmy, ktoré počas ich behu používajú $O(\log n)$ aritmetických operácií. Súčasťou práce je aj dôkaz korektnosti algoritmov ako aj popis a dôkaz všetkých potrebných matematických tvrdení. Implementácia týchto algoritmov slúži ako skúška ich korektnosti. Samotná implementácia algoritmov dáva možnosť aj reálne porovnávať efektívnosť jednotlivých algoritmov s efektívnosťou už existujúcich algoritmov.

KĽÚČOVÉ SLOVÁ: Fibonacciho postupnosť, Rekurentné postupnosti druhého rádu, Výpočet n-tého člena postupnosti, Efektívne algoritmy, Dynamické programovanie, Výpočet celočíselnej druhej odmocniny pomocou binárneho vyhľadávania, Umocňovanie matíc, Príslušnosť k rekurentnej postupnosti

Abstract

The main goal of this article is derivation, proof and implementation of some fast algorithms for linear recurrent sequences. The basic problem is the evaluation of n-th member of given sequence, the second problem is a checking the membership of the given number to recurrent sequence. While solving these problems, we also ran into the problem of finding the n-th power of a matrix. This thesis therefore also describes an algorithm for calculating the n-th power of a matrix, which also helps us understand the whole topic. All these problems are solvable in time $O(n)$. In this work, we are trying to create algorithms using $O(\log n)$ arithmetic operations. The work also includes proofs of correctness of algorithms and of all necessary mathematical statements. Implementation of these algorithms also serves as a test of their correctness. Implementation of algorithms also gives the possibility to experimentally compare the effectiveness of individual algorithms compared to the ones that already exist.

KEYWORDS: Fibonacci sequence, Linear recurrent sequences, Evaluation n-th number of sequence, Effective algorithms, Dynamic programming, Evaluation the integer square root of the given number with binary searching, Evaluation the n-th power of the matrix, Membership in recurrent sequences

Obsah

Úvod

Motivácia a ciele

Táto bakalárska práca sa zaoberá návrhom, realizáciou a analýzou algoritmu na nájdenie n-tého člena rekurentných postupností. Cieľom bolo navrhnúť algoritmus aplikovateľný na čo najväčšiu množinu rekurentných vzťahov, používajúc maximálne $O(\log_2 n)$ aritmetických operácií a zároveň optimalizácia nárokov na pamäť.

Podnetom na hľadanie takého algoritmu bol veľmi špeciálny algoritmus navrhnutý Daisuke Takahashim publikovaný v jeho článku [16] na nájdenie n-tého člena Fibonacciho postupnosti. Jeho algoritmus používa tiež rádovo $O(\log_2 n)$ aritmetických operácií, ale sa dá aplikovať iba na Fibonacciho postupnosti.

Algoritmus navrhnutý v tejto práci je aplikovateľný na väčšiu množinu rekurentných vzťahov (rekurentné postupnosti druhého rádu s konštantnými koeficientami) pričom nároky na počet bitových operácií a nároky na pamäť neboli porušené. Avšak práve toto rozšírenie aplikovateľnosti algoritmu už nedovolilo jeho ďalšiu optimalizáciu a domnievam sa, že pre určité špeciálne prípady rekurentných vzťahov tento algoritmus nebude optimálny.

Postup pri návrhu algoritmu

V prvom rade pravidlá známe a platné pre Fibonacciho postupnosti bolo treba zovšeobecniť, modifikovať a dokázať ich platnosť aj pre ďalšie rekurentné vzťahy (konkrétne sa to podarilo na rekurentné postupnosti druhého rádu s konštantnými koeficientami). Potom na základe týchto nových všeobecných pravidiel som zostrojil algoritmus typu Rozdeľuj a Panuj, redukujúci daný problém na viac podobných problémov s menším vstupom. Keďže metóda dynamického programovania pracuje s lepšou časovou náročnosťou ako algoritmus typu Rozdeľuj a Panuj už navrhnutý algoritmus som realizoval touto metódou. Tým sa podarilo dosiahnuť efektívnu časovú náročnosť $O(\log n)$,

ale narástli nároky na pamäť. Avšak voľbou vhodného rekurentného vzťahu sa aj pamäťovú náročnosť podarilo optimalizovať a dosiahnuť konštantnú hodnotu. Navrhnutý algoritmus potrebuje okrem rôznych pomocných indexov 6 premenných.

Kapitola 1

Rekurentné vzťahy druhého rádu

1.1 Stručne o rekurentných vzťahoch

Keďže zúžením rekurentných postupností na množinu reálnych čísiel sa platnosť a všeobecnosť navrhnutého algoritmu nemení, v ďalšom budeme používať rekurentné postupnosti definované na množine reálnych čísiel.

Definícia 1.1.1. *Rekurentná postupnosť m-tého rádu* je taká postupnosť $\{r_i\}_0^\infty$, kde $\forall i \in \mathbb{N} : r_i \in \mathbb{R}$; r_0,r_1...r_{m-1} sú konštanty a

$$\forall n \in \mathbb{N}, n \geq m : r_n = F(r_{n-1}, r_{n-2}, ..., r_{n-m-1}, n),$$

kde F je funkcia $F : \mathbb{R}^{m+1} \to \mathbb{R}$.

Ide teda o to, že každý nasledujúci člen rekurzívnej postupnosti sa dá vypočítať použitím určitej funkcie na m predchádzajúcich členov postupnosti. V ďalšom sa budeme zaoberať s rekurentnými postupnosťami kde F predstavuje sčítanie m prvkov vynásobených ľubovoľnými konštantami. Veľa známych matematických problémov sa týka rekurentných postupností. Príkladmi sú klasické problémy známe ako Hanojské veže, rozdelenie roviny na maximálny počet častí n priamkami, Jozefov problém a samozrejme Fibonacciho postupnosť.

1.2 Matematické základy algoritmu

Keďže navrhnutý algoritmus sa dá aplikovať len na podmnožinu všetkých rekurentných postupností, a to na rekurentné postupnosti druhého rádu, a je

založený na platnosti určitých matematických tvrdení, ďalšia časť práce bude venovaná vysvetleniu a zadefinovaniu týchto pojmov a dôkazu správnosti potrebných matematických tvrdení.

Z predchádzajúcej definície rekurentných postupností vyplýva, že pod pojmom rekurentné postupnosti druhého rádu rozumieme také rekurentné postupnosti, v ktorých nasledujúci prvok sa dá vypočítať aplikovaním rekurentného vzťahu na dva predchádzajúce členy postupnosti.

Definícia 1.2.1. Zadefinujme nasledovné rekurentné postupnosti:
$F_0 = 0$, $F_1 = 1$ a $F_n = F_{n-1} + F_{n-2}$
$L_0 = 2$, $L_1 = 1$ a $L_n = L_{n-1} + L_{n-2}$
$R_0 = a$, $R_1 = b$ a $R_n = R_{n-1} + R_{n-2}$
$P_0 = 0$, $P_1 = 1$ a $P_n = cP_{n-1} + dP_{n-2}$
$Q_0 = a$, $Q_1 = b$ a $Q_n = cQ_{n-1} + dQ_{n-2}$
pre $\forall n \in \mathbb{N}$: $n \geq 2$ $a, b, c, d \in \mathbb{R}$

Takto zadefinovaná postupnosť F_n sa nazýva *Fibonacciho postupnosť* a postupnosť L_n sa nazýva *Lucasova postupnosť*.

Platnosť formuly $L_n = F_{n-1} + F_{n+1}$ sa dá dokázať pomocou matematickej indukcie. Maticovú reprezentáciu Fibonacciho postupnosti používala aj Beáta Štupáková [15] a túto reprezentáciu tu zovšeobecníme na rekurentné postupnosti druhého rádu. V článku [10] autori sa snažia zovšeobecniť známe identity pre Fibonacciho postupnosť. V tomto článku je uvedená napríklad aj maticová reprezentácia postupnosti P_n.

Definícia 1.2.2.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Lema 1.2.3.

$$\mathbf{A}^n = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n = \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}$$

Dôkaz. Indukciou vzhľadom na n:
Pre $n = 1$ platí:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} F_0 & F_1 \\ F_1 & F_2 \end{pmatrix}$$

Teraz predpokladajme, že tvrdenie platí pre n a dokážme, že platí aj pre $n+1$:

$$\mathbf{A}^{n+1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{n+1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} =$$

čo sa podľa indukčného predpokladu rovná

$$= \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} F_n & F_{n+1} \\ F_{n-1}+F_n & F_n+F_{n+1} \end{pmatrix} = \begin{pmatrix} F_n & F_{n+1} \\ F_{n+1} & F_{n+2} \end{pmatrix}$$

□

Veta 1.2.4.

$$F_{m+n} = F_m F_{n+1} + F_{m-1} F_n$$

Dôkaz.

$$\mathbf{A}^{m+n} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^m \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n =$$

Potom podľa predchádzajúcej lemy 1.2.3:

$$= \begin{pmatrix} F_{m-1} & F_m \\ F_m & F_{m+1} \end{pmatrix} \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix} =$$
$$= \begin{pmatrix} F_{m-1}F_{n-1}+F_mF_n & F_{m-1}F_n+F_mF_{n+1} \\ F_mF_{n-1}+F_{m+1}F_n & F_mF_n+F_{m+1}F_{n+1} \end{pmatrix}$$

Pritom ale platí aj

$$\mathbf{A}^{m+n} = \begin{pmatrix} F_{m+n-1} & F_{m+n} \\ F_{m+n} & F_{m+n+1} \end{pmatrix}$$

Teda podľa rovnosti matíc vyplýva: $F_{m+n} = F_{m-1}F_n + F_m F_{n+1}$. □

Táto veta má jeden dôležitý dôsledok:

Dôsledok 1.2.5.

$$F_{2n} = F_n(F_{n+1} + F_{n-1})$$
$$F_{2n+1} = F_{n+1}^2 + F_n^2$$

Dôkaz. Dosadením n v prvom prípade a $n+1$ v druhom prípade do rovnice predchádzajúcej vety namiesto m. □

Lema 1.2.6.

$$\mathbf{A}^n \begin{pmatrix} a & b \\ b & a+b \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} a & b \\ b & a+b \end{pmatrix} = \begin{pmatrix} R_n & R_{n+1} \\ R_{n+1} & R_{n+2} \end{pmatrix}$$

Dôkaz. Indukciou vzhľadom na n:
Pre $n = 0$ zrejme platí rovnosť. Pre $n = 1$ platí:

$$\mathbf{A}\begin{pmatrix} a & b \\ b & a+b \end{pmatrix} = \begin{pmatrix} b & a+b \\ a+b & a+2b \end{pmatrix} = \begin{pmatrix} R_1 & R_2 \\ R_2 & R_3 \end{pmatrix}$$

Teraz predpokladajme, že tvrdenie platí pre n a dokážme, že platí aj pre $n+1$:

$$\mathbf{A}^{n+1}\begin{pmatrix} a & b \\ b & a+b \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} a & b \\ b & a+b \end{pmatrix} =$$

čo sa podľa indukčného predpokladu rovná

$$= \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} R_n & R_{n+1} \\ R_{n+1} & R_{n+2} \end{pmatrix} = \begin{pmatrix} R_{n+1} & R_{n+2} \\ R_n + R_{n+1} & R_{n+1} + R_{n+2} \end{pmatrix} =$$
$$= \begin{pmatrix} R_{n+1} & R_{n+2} \\ R_{n+2} & R_{n+3} \end{pmatrix}$$

Správnosť tejto lemy môžeme otestovať aj pomocou toho faktu, že po dosadení $a = 0$ a $b = 1$ dostaneme lemu 1.2.3. □

Veta 1.2.7.

$$R_n = aF_{n-1} + bF_n$$

Dôkaz.

$$\begin{pmatrix} R_n & R_{n+1} \\ R_{n+1} & R_{n+2} \end{pmatrix} = \mathbf{A}^n \begin{pmatrix} a & b \\ b & a+b \end{pmatrix} = \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}\begin{pmatrix} a & b \\ b & a+b \end{pmatrix} =$$
$$= \begin{pmatrix} aF_{n-1} + bF_n & bF_n + (a+b)F_{n+1} \\ aF_n + bF_{n+1} & bF_n + (a+b)F_{n+1} \end{pmatrix}$$

Teda podľa rovnosti matíc platí: $R_n = aF_{n-1} + bF_n$

□

Lema 1.2.8.

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n-1}\begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} = \begin{pmatrix} P_{n-1} & P_n \\ P_n & P_{n+1} \end{pmatrix}$$

Dôkaz. Indukciou vzhľadom na n:
Pre $n = 1$ zrejme platí rovnosť. Pre $n = 2$ platí tiež.
Teraz predpokladajme, že tvrdenie platí pre n a dokážme, že platí aj pre $n+1$:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n-1}\begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} =$$

čo sa podľa indukčného predpokladu rovná

$$= \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix} \begin{pmatrix} P_{n-1} & P_n \\ P_n & P_{n+1} \end{pmatrix} = \begin{pmatrix} P_n & P_{n+1} \\ dP_{n-1} + cP_n & dP_n + cP_{n+1} \end{pmatrix} =$$

$$= \begin{pmatrix} P_n & P_{n+1} \\ P_{n+1} & P_{n+2} \end{pmatrix}$$

Správnosť tejto lemy môžeme otestovať aj pomocou toho faktu, že po dosadení $c = 1$ a $d = 1$ dostaneme lemu 1.2.3. □

Veta 1.2.9.

$$P_{m+n} = dP_{n-1}P_m + P_nP_{m+1}$$

Dôkaz. Inverzná matica k $\begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix}$ je $\begin{pmatrix} -c & 1 \\ 1 & 0 \end{pmatrix}$

Potom platí:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{m+n-1} \begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} = \begin{pmatrix} P_{m+n-1} & P_{m+n} \\ P_{m+n} & P_{m+n+1} \end{pmatrix}$$

Pritom:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{m+n-1} \begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n} \mathbf{I} \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{m-1} \begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} =$$

$$= \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n} \begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} \begin{pmatrix} -c & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{m-1} \begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} =$$

Podľa predchádzajúcej lemy 1.2.8:

$$= \begin{pmatrix} P_n & P_{n+1} \\ P_{n+1} & P_{n+2} \end{pmatrix} \begin{pmatrix} -c & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} P_{m-1} & P_m \\ P_m & P_{m+1} \end{pmatrix} =$$

$$= \begin{pmatrix} -cP_n + P_{n+1} & P_n \\ -cP_{n+1} + P_{n+2} & P_{n+1} \end{pmatrix} \begin{pmatrix} P_{m-1} & P_m \\ P_m & P_{m+1} \end{pmatrix} =$$

$$= \begin{pmatrix} (-cP_n + P_{n+1})P_{m-1} + P_nP_m & (-cP_n + P_{n+1})P_m + P_nP_{m+1} \\ (-cP_{n+1} + P_{n+2})P_{m-1} + P_{n+1}P_m & (-cP_{n+1} + P_{n+2})P_m + P_{n+1}P_{m+1} \end{pmatrix}$$

Teda z rovnosti matíc vyplýva: $P_{m+n} = (-cP_n + P_{n+1})P_m + P_nP_{m+1}$

□

Táto veta má jeden dôležitý dôsledok:

Dôsledok 1.2.10.

$$P_{2n} = -cP_n^2 + 2P_nP_{n+1}$$
$$P_{2n+1} = -cP_nP_{n+1} + P_{n+1}^2 + P_nP_{n+2}$$

Dôkaz. Dosadením n v prvom prípade a $n+1$ v druhom prípade do rovnice predchádzajúcej vety namiesto m. □

Lema 1.2.11.

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n-1} \begin{pmatrix} a & b \\ b & bc+ad \end{pmatrix} = \begin{pmatrix} Q_{n-1} & Q_n \\ Q_n & Q_{n+1} \end{pmatrix}$$

Dôkaz. Indukciou vzhľadom na n:
Pre $n = 1$ zrejme platí rovnosť. Pre $n = 2$ platí tiež.
Teraz predpokladajme, že tvrdenie platí pre n a dokážme, že platí aj pre $n+1$:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix} \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n-1} \begin{pmatrix} a & b \\ b & bc+ad \end{pmatrix} =$$

čo sa podľa indukčného predpokladu rovná

$$= \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix} \begin{pmatrix} Q_{n-1} & Q_n \\ Q_n & Q_{n+1} \end{pmatrix} = \begin{pmatrix} Q_n & Q_{n+1} \\ dQ_{n-1}+cQ_n & dQ_n+cQ_{n+1} \end{pmatrix} =$$
$$= \begin{pmatrix} Q_n & Q_{n+1} \\ Q_{n+1} & Q_{n+2} \end{pmatrix}$$

□

Veta 1.2.12.

$$Q_n = bP_n + adP_{n-1}$$

Dôkaz.

$$\begin{pmatrix} Q_n & Q_{n+1} \\ Q_{n+1} & Q_{n+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^n \begin{pmatrix} a & b \\ b & bc+ad \end{pmatrix} =$$
$$= \begin{pmatrix} P_n & P_{n+1} \\ P_{n+1} & P_{n+2} \end{pmatrix} \begin{pmatrix} -c & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a & b \\ b & bc+ad \end{pmatrix} =$$
$$= \begin{pmatrix} P_n & P_{n+1} \\ P_{n+1} & P_{n+2} \end{pmatrix} \begin{pmatrix} -ac+b & -bc+bc+ad \\ a & b \end{pmatrix} =$$
$$= \begin{pmatrix} (-ac+b)P_n + aP_{n+1} & adP_n + bP_{n+1} \\ (-ac+b)P_{n+1} + aP_{n+2} & adP_{n+1} + bP_{n+2} \end{pmatrix}$$

Teda podľa rovnosti matíc platí:

$$Q_n = (-ac+b)P_n + aP_{n+1} = -acP_n + bP_n + acP_n + adP_{n-1} = bP_n + adP_{n-1}$$

□

1.3 Výpočet n-tého členu Fibonnaciho postupnosti

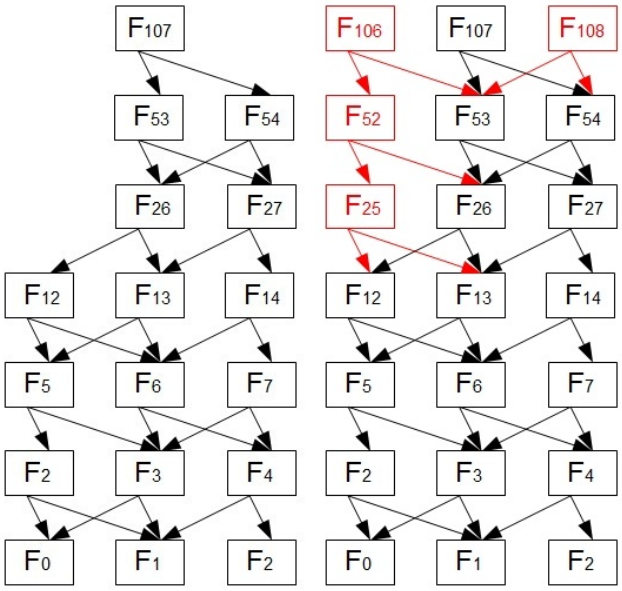
Dôsledok horeuvedenej vety o Fibonacciho postupnosti 1.2.5 nám dáva návod na výpočet n-tého členu. Túto metódu vieme zefektívniť nasledujúcim spôsobom. Keďže F_{2n} závisí od F_n, F_{n+1} a F_{n-1}, a je známe, že F_{n+1} je tiež závisí od F_n a F_{n-1} rekurentný vzťah pre F_{2n} sa teda dá skrátiť nasledovne:

$$F_{2n} = F_n(F_{n+1} + F_{n-1}) = F_n(F_n + F_{n-1} + F_{n-1}) = 2F_{n-1}F_n + F_n^2$$

Teda platí:

$$F_{2n} = 2F_{n-1}F_n + F_n^2$$
$$F_{2n+1} = F_{n+1}^2 + F_n^2$$

Nasledujúca schéma znázorní rekurzívne volania pre F_{107}:



V prvom rade si môžeme uvedomiť, že s malou úpravou sa dá popri F_{107} vypočítať aj F_{106} a F_{108} (druhý obraz). To sa nám zíde hlavne preto, lebo na výpočet R_n podľa vety 1.2.7 je potrebné naraz poznať F_n a F_{n-1}. Po druhé si môžeme uvedomiť, že ani jedna úroveň neobsahuje viac ako tri prvky. Úroveň so vstupným indexom budeme nazývať najvyššou. Potom najnižšia úroveň bude úroveň, kde sa rekurzívne volania môžu skončiť, teda prvky na najnižšej úrovni podkladáme za známe. (Predpokladajme, že poznáme aj tretí a štvrtý člen postupnosti.)
Postupne dokážme vyššie spomínané slovne popísané tvrdenia.

Tvrdenie 1.3.1. *Každá úroveň rekurzívnych volaní obsahuje práve tri členy postupnosti, ktoré sú zároveň susednými členmi danej postupnosti.*

Dôkaz. Tvrdenie dokážme pomocou matematickej indukcie. Najvyššia úroveň obsahuje tri nami zvolené členy. Základný predpoklad matematickej indukcie teda platí. Zoberme vstupný index, ako stredný prvok. Ostatné prvky si zvoľme, aby tvrdenie platilo, tj. jeho susedné členy v postupnosti.
Nech teda platí indukčný predpoklad, pre nejakú úroveň, dokážme, že tvrdenie platí aj pre o jednu nižšiu úroveň. Nech úroveň, pre ktorú platí indukčný krok obsahuje prvky s indexmi $n-1$, n a $n+1$. Podľa parity n sú dve možnosti.
Nech n je párne. Potom $n = 2k$, teda indexy sú $2k-1$, $2k$ a $2k+1$. Teda podľa upraveného rekurentného na výpočet týchto indexov potrebujeme tieto členy postupnosti:

$$F_{2k-1} = F_{2(k-1)+1} = F^2_{(k-1)+1} + F^2_{k-1} = F^2_k + F^2_{k-1}$$

$$F_{2k} = 2F_{k-1}F_k + F^2_k$$

$$F_{2k+1} = F^2_{k+1} + F^2_k$$

Z toho vyplýva, že o jedno nižšia úroveň musí obsahovať členy z postupnosti s indexmi $k-1$, k, $k+1$ a žiadny iný prvok nepotrebujeme na výpočet prvkov východiskovej úrovne. Prípade, keď n je nepárne číslo dôkaz je podobný. Tým je dokázaná správnosť indukčného kroku a teda správnosť celého tvrdenia. □

Tvrdenie 1.3.2. *Počet úrovní rekurzívnych volaní je $O(\log_2 n)$, kde n je vstupný index.*

Dôkaz. Pozrime sa na stredné prvky na rekurzívnych úrovniach, teda na stredný stĺpec na obrázku. Na najvyššej úrovni stredným prvkom je vstupný index, na najnižšej úrovni stredným členom je prvok, ktorého hodnotu už poznáme, teda jeho index je 1 alebo 2. (Môžeme určiť, za vstupnú podmienku riadok F_0, F_1, F_2, alebo F_1, F_2, F_3, keďže hodnoty týchto členov poznáme.) Potom z rekurzívneho vzťahu vyplýva, že v každom kroku stredný člen je rádovo dvakrát väčší, ako stredný člen na predchádzajúcej úrovni. Potom ale platí:

$$z * 2^k \approx n,$$

kde z je 1 alebo 2. k je počet úrovní a n je vstupný index.

$$2^k \approx n$$

$$k \approx \log_2 n$$

□

Tvrdenie 1.3.3. *Pri výpočte n-tého člena postupnosti sa dá vypočítať aj $(n-1)$-vý a $(n+1)$-vý člen postupnosti, pomocou maximálne $O(\log n)$ ďalších rekurzívnych volaní.*

Dôkaz. Toto tvrdenie je vlastne dôsledkom predchádzajúceho tvrdenia. Počet úrovní je $O(\log_2 n)$. Z rekurentného vzťahu vyplýva, že od druhej najvyššej úrovne všetky úrovne obsahujú aspoň dva členy. Keďže na každej úrovni vieme jednoducho získať tri prvky na každej úrovni potrebujeme pridať len jeden prvok navyše. (Okrem najvyššej úrovne, kde potrebujeme pridať dva prvky.) Teda musíme pridať maximálne toľko prvkov, koľko je počet úrovní. □

V tejto podkapitole sme vlastne uviedli triviálny rekurzívny algoritmus na výpočet n-tého členu Fibonacciho potupnosti. V nasledujúcej podkapitole navrhneme vylepšený algoritmus. Toto vylepšenie dosiahneme pomocou používania dynamického programovania, pričom si budeme musieť pamätať len konštantný počet členov postupnosti.

1.3.1 Iteračný algoritmus na výpočet n-tého členu Fibonacciho postupnosti

Teraz uvedieme finálnu podobu algoritmu na výpočet n-tého člena Fibonacciho postupnosti.

Algoritmus(n)
$aktindex \leftarrow getbit(n,1) * 2 + getbit(n,2)$
$P_0[0,1,2] \leftarrow [P_{aktindex-1}, P_{aktindex}, P_{aktindex+1}]$
$aktpole \leftarrow 0$
$inaktpole \leftarrow 1$
for $i = 3$ **to** $\log_2 n$ **do**
  $aktindex \leftarrow aktindex * 2 + getbit(n,i)$
  $aktpole \leftarrow (aktpole + 1) mod 2$
  $inaktpole \leftarrow (inaktpole + 1) mod 2$
  if aktindex je parne then
    $P_{aktpole}[0] \leftarrow P_{inaktpole}[0]^2 + P_{inaktpole}[1]^2$
    $P_{aktpole}[1] \leftarrow 2 * P_{inaktpole}[0] * P_{inaktpole}[1] + P_{inaktpole}[1]^2$
    $P_{aktpole}[2] \leftarrow P_{inaktpole}[1]^2 + P_{inaktpole}[2]^2$
  else
    $P_{aktpole}[0] \leftarrow 2 * P_{inaktpole}[0] * P_{inaktpole}[1] + P_{inaktpole}[1]^2$
    $P_{aktpole}[1] \leftarrow P_{inaktpole}[1]^2 + P_{inaktpole}[2]^2$
    $P_{aktpole}[2] \leftarrow 2 * P_{inaktpole}[1] * P_{inaktpole}[1] + P_{inaktpole}[2]^2$
  end if

$i \leftarrow i + 1$
end for
return $P_{aktpole}$

Funkcia $getbit(x, y)$ vráti y-ty najvyšší bit v čísle x v konštantnom čase. Keď nejaký procesor má implementovanú takúto operáciu, tak ju môžeme použiť. Inak pred spustením tohoto algoritmu číslo x treba rozbiť na bity. Táto operácia trvá $\log_2 x$ času, keďže každé číslo sa reprezentuje vo dvojkovej sústave, teda je reprezentované ako reťazec núl a jednotiek, dĺžky $\log_2 x$.

Tvrdenie 1.3.4. *Iteračný algoritmus na výpočet n-tého člena Fibonacciho postupnosti pracuje v čase $O(\log_2 n)$, teda vykoná $O(\log_2 n)$ násobení.*

Dôkaz. Algoritmus obsahuje jediný cyklus s hornou hranicou $\log_2 n$. Ďalej obsahuje len konštantný počet násobení, resp. čítanie dvoch čísiel, čo má lineárnu zložitosť, teda je zanedbateľné. Algoritmus teda vzhľadom na násobenie má zložitosť $O(\log_2 n)$. □

Tvrdenie 1.3.5. *Program využíva $O(n)$ priestoru.*

Dôkaz. Potrebujeme ukladať do pamäti konštantne veľa premenných a vstup. Vstup je číslo n, reprezentované v binárnej sústave ako $\log_2 n$ dlhý reťazec jednotiek a núl. Pritom si treba uvedomiť, že vypočítané hodnoty postupnosti môžu byť rádovo väčšie ako vstupný index, teda počas výpočtu n-tého člena postupnosti nám rastú nároky na pamäť. Posledný vyrátaný člen postupnosti je rádovo 2^n veľké číslo, reprezentované ako n dlhý reťazec jednotiek a núl. □

Tvrdenie 1.3.6. *Korektnosť: Uvedený algoritmus vypočíta hodnotu n-tého člena Fibonacciho postupnosti.*

Dôkaz. Treba dokázať že *aktindex* nadobudne hodnotu $2n$ alebo $2n + 1$ v každom kroku. $2n$ a $2n+1$ predstavujú indexy z rovníc (1.3). Teda keď sme na takej úrovni, kde stredný člen má index n, tak stredný člen na o jedno vyššej úrovni bude mať hodnotu $2n$ alebo $2n + 1$. Keď to platí, tak celý algoritmus sme previedli na už dokázané tvrdenie 1.3. To že *aktindex* nadobúda tieto požadované hodnoty vyplýva zo spôsobu určenia jeho hodnôt:

$aktindex \leftarrow aktindex * 2 + getbite(n, i)$

□

1.4 Výpočet n-tého člena v ostatných postupnostiach

Na začiatku tejto kapitoly sme definovali ďalšie rekurentné postupnosti druhého rádu R, P a Q. Viď definíciu 1.2.1. Na nájdenie n-tého člena postupnosti R bude možné použiť predchádzajúci algoritmus. Podľa vety 1.2.7 n-tý člen postupnosti R dostaneme sčítaním dvoch čísiel $aF_{n-1} + bF_n$. F_{n-1} a F_n sa dajú vypočítať pomocou predchádzajúceho algoritmu súčasne a a, b sú konštanty, teda ani časová, ani pamäťová zložitosť algoritmu sa nepokazí.

Prípade postupnosti P môžeme postupovať analogicky, ako v prípade Fibonacciho postupnosti. Pre postupnosť P platí podobný dôsledok, ako pre Fibonacciho postupnosť 1.2.10, teda drobnými zmenami sa dá horeuvedený algoritmus rozšíriť na postupnosť P. Bez dôkazu uvedieme iteračný algoritmus na výpočet n-tého člena postupnosti P, ktorý pracuje v logaritmickom čase. Všetky tvrdenia o zložitosti a korektnosti sú veľmi podobné, ako pre algoritmus pre Fibonacciho postupnosť, preto ich neuvádzame.

Algoritmus(n)
$aktindex \leftarrow getbite(n,1) * 2 + getbite(n,2)$
$P_0[0,1,2] \leftarrow [F_{aktindex-1}, F_{aktindex}, F_{aktindex+1}]$
$aktpole \leftarrow 0$
$inaktpole \leftarrow 1$
for $i = 3$ **to** $\log_2 n$ **do**
  $aktindex \leftarrow aktindex * 2 + getbite(n,i)$
  $aktpole \leftarrow (aktpole + 1) mod 2$
  $inaktpole \leftarrow (inaktpole + 1) mod 2$
  if aktindex je parne then
    $P_{aktpole}[0] \leftarrow d * P_{inaktpole}[0]^2 + P_{inaktpole}[1]^2$
    $P_{aktpole}[1] \leftarrow 2 * d * P_{inaktpole}[0] * P_{inaktpole}[1] + c * P_{inaktpole}[1]^2$
    $P_{aktpole}[2] \leftarrow d * P_{inaktpole}[1]^2 + P_{inaktpole}[2]^2$
  else
    $P_{aktpole}[0] \leftarrow 2 * d * P_{inaktpole}[0] * P_{inaktpole}[1] + c * P_{inaktpole}[1]^2$
    $P_{aktpole}[1] \leftarrow d * P_{inaktpole}[1]^2 + P_{inaktpole}[2]^2$
    $P_{aktpole}[2] \leftarrow 2 * d * P_{inaktpole}[1] * P_{inaktpole}[1] + c * P_{inaktpole}[2]^2$
  end if
  $i \leftarrow i + 1$
end for
return $P_{aktpole}$

V prípade postupnosti Q je situácia úplne podobná, ako situácia s postupnosťou R. Z vety 1.2.12 vyplýva, že n-tý člen postupnosti Q sa dá získať pomocou už známych členov postupnosti P.

1.5 Výpočet rekurentných postupnosti s absolútnym členom

Zadefinujme postupnosť $\bar{Q}_n = c\bar{Q}_{n-1} + d\bar{Q}_{n-2} + e$, $\forall n \in \mathbb{N}$: $n \geq 2$ a $\bar{Q}_0 = a$, $\bar{Q}_1 = b$, kde $a, b, c, d, e \in \mathbb{R}$. Dokážme, že pomocou horeuvedeného algoritmu sa dá riešiť aj postupnosť $\bar{Q}$. $\bar{Q}_n$ vlastne reprezentuje tie rekurzívne vzťahy, ktoré obsahujú absolútny člen.
Najprv treba dokázať dve pomocné lemy:

Lema 1.5.1.

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^n = \begin{pmatrix} -cP_n + P_{n+1} & P_n \\ -cP_{n+1} + P_{n+2} & P_{n+1} \end{pmatrix}$$

Dôkaz. Z lemy 1.2.8 platí:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^n \begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} = \begin{pmatrix} P_n & P_{n+1} \\ P_{n+1} & P_{n+2} \end{pmatrix}$$

V dôkaze vety 1.2.9 vyrátame inverznú maticu k $\begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix}$. Vynásobením sprava oboch strán rovnice touto maticou dostaneme:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^n = \begin{pmatrix} P_n & P_{n+1} \\ P_{n+1} & P_{n+2} \end{pmatrix} \begin{pmatrix} -c & 1 \\ 1 & 0 \end{pmatrix}$$

Po vynásobení pravej strany rovnice dostaneme žiadanú rovnosť. □

Poznámka 1.5.2. Predchádzajúca veta vyjadruje súvislosť medzi umocňovaním matíc istého typu a rekurentnými postupnosťami.

Lema 1.5.3. *Označme* $M_n := \sum_{i=1}^{n} \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^i$ *Potom* M_n *v uzavretom tvare je:*

$$M_n = \frac{1}{c+d-1} \begin{pmatrix} (c^2-c)P_{n+1} + (1-2c)P_{n+2} + P_{n+3} - d & (1-c)P_{n+1} + P_{n+2} - 1 \\ -cdP_{n+1} + (d-c)P_{n+2} + P_{n+3} - d & bP_{n+1} + P_{n+2} - c - d \end{pmatrix}$$

Dôkaz.

$$M_n = \sum_{i=1}^{n} \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^i \tag{1.1}$$

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix} M_n = \sum_{i=2}^{n+1} \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^i \tag{1.2}$$

Potom z (1.2)-(1.1) dostávame:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix} M_n - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} M_n = \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n+1} - \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}$$

$$\begin{pmatrix} -1 & 1 \\ d & c-1 \end{pmatrix} M_n = \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n+1} - \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}$$

Vypočítajme inverznú maticu k $\begin{pmatrix} -1 & 1 \\ d & c-1 \end{pmatrix}$ je $\begin{pmatrix} \frac{1-c}{c+d-1} & \frac{1}{c+d-1} \\ \frac{b}{c+d-1} & \frac{1}{c+d-1} \end{pmatrix}$. Potom podľa predchádzajúcej lemy platí:

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n+1} - \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix} = \begin{pmatrix} -cP_{n+1} + P_{n+2} & P_{n+1} \\ -cP_{n+2} + P_{n+3} & P_{n+2} \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix} =$$

$$= \begin{pmatrix} -cP_{n+1} + P_{n+2} & P_{n+1} - 1 \\ -cP_{n+2} + P_{n+3} - d & P_{n+2} - c \end{pmatrix}$$

$$\begin{pmatrix} -1 & 1 \\ d & c-1 \end{pmatrix} M_n = \begin{pmatrix} -cP_{n+1} + P_{n+2} & P_{n+1} - 1 \\ -cP_{n+2} + P_{n+3} - d & P_{n+2} - c \end{pmatrix}$$

Rovnicu vynásobme zľava s inverznou maticou, ktorú sme práve získali.

$$M_n = \begin{pmatrix} \frac{1-c}{c+d-1} & \frac{1}{c+d-1} \\ \frac{b}{c+d-1} & \frac{1}{c+d-1} \end{pmatrix} \begin{pmatrix} -cP_{n+1} + P_{n+2} & P_{n+1} - 1 \\ -cP_{n+2} + P_{n+3} - d & P_{n+2} - c \end{pmatrix}$$

Po úpravách dostaneme:

$$M_n = \frac{1}{c+d-1} \begin{pmatrix} (c^2-c)P_{n+1} + (1-2c)P_{n+2} + P_{n+3} - d & (1-c)P_{n+1} + P_{n+2} - 1 \\ -cdP_{n+1} + (d-c)P_{n+2} + P_{n+3} - d & bP_{n+1} + P_{n+2} - c - d \end{pmatrix}$$

Teda uvedená lema platí. □

Zadefinujme ešte jednu postupnosť: $S_n = cS_{n-1} + dS_{n-2} + e$, $\forall n \in \mathbb{N}$: $n \geq 2$ a $S_0 = 0$, $S_1 = 0$, kde $c, d, e \in \mathbb{R}$.

Veta 1.5.4.

$$\bar{Q}_n = Q_n + S_n$$

Dôkaz. Pomocou matematickej indukcie. Pre $n = 0$ a $n = 1$ tvrdenie platí. Ďalej predpokladajme, že tvrdenie platí aj pre všetky menšie indexy, ako n a dokážme, že platí aj pre n.
Teda $\bar{Q}_n = c\bar{Q}_{n-1} + d\bar{Q}_{n-2} + e$ podľa indukčného predpokladu platí aj $\bar{Q}_n = cQ_{n-1} + cS_{n-1} + dQ_{n-2} + dS_{n-2} + e$, čo po úprave je

$$\bar{Q}_n = cQ_{n-1} + dQ_{n-2} + cS_{n-1} + dS_{n-2} + e$$

$$\bar{Q}_n = Q_n + S_n$$

Tvrdenie platí. □

Keďže Q_n v logaritmickom čase počítať už vieme, stačí nám navrhnúť spôsob na výpočet S_n. Treba dokázať nasledujúce tvrdenie:

Veta 1.5.5.

$$S_{n+2} = e\frac{dP_{n+1} + P_{n+2} - 1}{c + d - 1} + eP_{n+1}$$

Dôkaz. Ľahko sa dokáže, že

$$\begin{pmatrix} S_n & S_{n+1} \\ S_{n+1} & S_{n+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^n \begin{pmatrix} 0 & 0 \\ 0 & e \end{pmatrix} + M_n \begin{pmatrix} 0 & 0 \\ e & e \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ e & e \end{pmatrix}$$

Po použití liem 1.5.1 a 1.5.3 a po úpravách dostaneme rovnosť:

$$\begin{pmatrix} S_n & S_{n+1} \\ S_{n+1} & S_{n+2} \end{pmatrix} = M_n \begin{pmatrix} 0 & 0 \\ e & e \end{pmatrix} + \begin{pmatrix} 0 & eP_n \\ 0 & eP_{n+1} + e \end{pmatrix}$$

Z čoho po dosadení za M_n vyplýva:

$$S_{n+2} = e\frac{dP_{n+1} + P_{n+2} - d - c}{c + d - 1} + eP_{n+1} + e$$

Z čoho vyplýva platnosť vety. □

Podľa tejto vety dokážeme S_n vypočítať v logaritmickom čase, keďže P_{n+1} a P_{n+2} dokážeme naraz vypočítať v $\log n$ čase. Týmto sme skonštruovali algoritmus pre rekurentné postupnosti druhého rádu s konštantnými koeficientami.

Kapitola 2

Analýza a porovnanie niektorých algoritmov, na určenie n-tého člena Fibonacciho postupnosti

2.1 Použité algoritmy

V tejto kapitole porovnáme 4 algoritmy na nájdenie n-tého člena Fibonacciho postupnosti. Tri sú z nich prevzaté od iných autorov a tie budeme porovnávať s v tejto práci navrhnutým algoritmom.
Prvý algoritmus predstavuje prirodzený postup. n-tý člen Fibonacciho postupnosti sa vypočíta presne podľa definície tejto postupnosti. Tento algoritmus sa podstatne líši od ostatných nižšie uvedených algoritmov. Tento algoritmus neobsahuje žiadnu optimalizáciu, je lineárny vzhľadom na index n, zároveň pravdepodobne bude najpomalší. Má ale tú výhodu, že jeho korektnosť priamo vyplýva z definície Fibonacciho postupnosti, keďže túto definíciu využíva na postupné vypočítanie 3 po sebe idúcich členov danej postupnosti. Tento algoritmus je implementovaný v **algoritmus1.cpp**.

Druhý algoritmus Daisuke Takahashiho, ktorý je podrobne popísaný v článku [16] je implementovaný v **algoritmus4.cpp**.
Tretí algoritmus „Alternate, an integer iteration“ je popísaný okrem iných aj v článku [8]. Tento algoritmus sa podobá na tu popísaný algoritmus, keďže využíva tie isté vzťahy, ktoré sa používajú v prvej kapitole 1.2.5. Implementovaný je v **algoritmus5.cpp**.
Štvrtý v tejto práce popísaný algoritmus sa nachádza v prvej kapitole, v **algoritmus2.cpp**.

Posledné tri algoritmy sa navzájom podobajú, keďže v rámci jedného cyklu dĺžky $O(\log_2 n)$ iterujú celé čísla. Preto sa dajú porovnať aj podľa toho, koľko operácií sa vykoná počas jedného behu cyklu. V premenných, ktoré reprezentujú členy postupnosti sa počas behu programu vyskytnú aj rádovo väčšie čísla, ako vstupný index. (i-ty člen postupnosti je rádovo 2^i - veľké číslo.)[1] Na týchto premenných sa vykonajú tri typy operácie: vynásobenie dvoch členov postupnosti, výpočet druhej mocniny člena postupnosti a sčítanie dvoch členov postupnosti. Okrem týchto operácií sa vykonajú aj operácie s malými konštantami, alebo premennými, ktoré sú rádovo menšie, ako aktuálny člen postupnosti. Tieto operácie môžeme nazvať operáciami na „malých číslach". Takto zadefinované 4 typy operácií:

- Vynásobenie dvoch členov postupnosti
- Výpočet druhej mocniny člena postupnosti
- Sčítanie dvoch členov postupnosti
- Operácie na „malých číslach"

vieme zoradiť podľa toho, koľko bitových operácií sa musia vykonať na ich realizáciu. Podľa [16] na výpočet druhej mocniny čísla potrebujeme menej bitových operácií, ako na vynásobenie dvoch čísiel rovnakej dĺžky. Tiež podľa [16] sčítanie dvoch čísiel je rýchlejšia operácia, ako výpočet druhej mocniny. Operácie na „malých číslach" trvajú zanedbateľne málo času oproti sčítania dvoch členov Fibonacciho postupnosti. Teda môžeme zoradiť jednotlivé operácie nasledovne:

násobenie $>$ druhá mocnina $>$ sčítanie $\gg$ operácie na „malých číslach"

2.2 Porovnávanie algoritmov podľa použitých operácií

V tejto podkapitole porovnáme jednotlivé algoritmy podľa počtu operácií vykonaných počas jedného cyklu. Síce operácie s „malými číslami" sú rýchlejšie, ako všetky ostatné operácie, pre úplnosť uvádzame aj tieto operácie.

[1] Pomocou Binetovho vzorca sa dá dokázať, že asymptotický odhad pre F_n je $\frac{\varphi^n}{\sqrt{5}}$ veľké číslo, kde φ tu označuje zlatý rez $\varphi = \frac{1+\sqrt{5}}{2}$. Binetov vzorec sa uvedie napr. aj v bakalárskej práci [15], ale skoro všade sa nachádza, kde sa hovorí o Fibonacciho postupnosti.

	násobenie	druhá mocnina	sčítanie	malé čísla
D. T. algoritmus	0	2	6	8
Alternate	2	0	$4 - 5^2$	0
Algoritmus z Kapitoly 1	$1 - 2^3$	$5 - 4^3$	3	$1 - 2^3$

Tabuľka 2.1: Počet jednotlivých operácií počas jedného behu cyklu

Podľa predchádzajúcej tabuľky dokážeme zoradiť algoritmy podľa efektívnosti. Podľa očakávania D. Takahashiho algoritmus bude najrýchlejší, potom Alternate, a algoritmus z prvej kapitoly. Najpomalší bude algoritmus spomínaný hneď na začiatku tejto kapitoly, algoritmus podľa definície. Aby sme sa presvedčili o správnosti nášho predpokladu vykonali sme merania behov jednotlivých algoritmov. Algoritmy boli implementované pomocou knižnice NTL v programovacom jazyku C++. Výsledky meraní a vyhodnotenie výsledkov budú popísané v ďalšej podkapitole.

2.3 Porovnávanie algoritmov podľa meraných časov

Vstupný	**Merané časy behu algoritmov na jednotlivé vstupy** (sec)			
index	D. T. algoritmus	Alternate	Algoritmus z Kapitoly 1	Prirodzený postup
1 000	0.000108957	9.20296e-05	0.00010705	0.000324965
10 000	0.000193834	0.000263929	0.000449181	0.00822306
100 000	0.00518703	0.00907183	0.019192	0.424006
1 000 000	0.125531	0.229387	0.487688	40.391
2 500 000	0.540508	0.97237	2.0968	251.345
5 000 000	1.61849	2.92191	6.27913	1006.04
10 000 000	4.84948	8.7129	18.9938	-

Tabuľka 2.2: Merané časy behu algoritmov na jednotlivé vstupy v sekundách

Merania teda potvrdili naše teoretické poradie.

[2]Podľa aktuálnej hodnoty markOdd(j)
[3]Podľa parity aktuálneho indexu aktindex

Kapitola 3

Umocňovanie matíc a súvislosť medzi umocňovaním matíc druhého rádu a rekurenciami druhého rádu

V prvej kapitole sme už ukázali jednu možnosť, ako vyrátať n-tú mocninu matice pomocou algoritmu z prvej kapitoly a pomocou matematického vzorca z lemy 1.5.1, táto skutočnosť je spomínaná aj v poznámke 1.5.2. Keďže touto metódou sme dokázali vyrátať n-tú mocninu matice použitím $O(\log_2 n)$ aritmetických operácií, oplatí sa zamyslieť sa nad tým, či sa dá vytvoriť algoritmus, ktorý umocní istú maticu na n-tú použitím $O(\log_2 n)$ maticových násobení. Existuje veľa algoritmov na násobenie dvoch matíc, ktoré vynásobia dané matice pomocou polynomiálne veľa aritmetických operácií vzhľadom na veľkosť matíc. Keď si zvolíme pevnú veľkosť matíc, tak si môžeme všimnúť, že vynásobenie dvoch matíc pevnej - tej istej - dĺžky sa počíta ako operácia zahrňujúca konštantne veľa aritmetických operácií. V ďalšom na vypočítanie n-tej mocniny matíc budeme hľadať taký algoritmus, ktorý používa minimálny počet takýchto násobení.

Najprv treba upresniť podmienky na vstupné matice, tj. pre aké matice má zmysel hľadať algoritmus na vypočítanie n-tej mocniny. Takýto algoritmus má zmysel hľadať iba pre štvorcové matice.

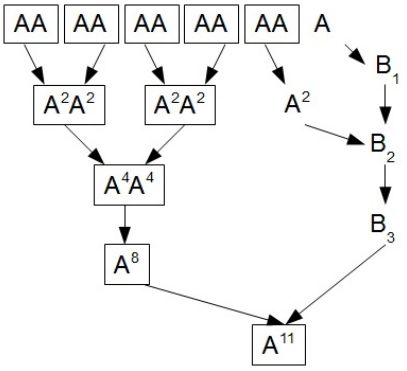
Pre štvorcové matice ale existuje efektívnejšia technika násobenia matíc Strassenov algoritmus, ktorý dokáže vynásobiť matice typu $n \times n$ v čase $O(n^{2.81})$. Najlepšia známa metóda na násobenie štvorcových matíc do roku 1990 má časovú zložitosť $O(n^{2.276})$.[3]

Naším zámerom je teda použiť čo najmenší počet takýchto maticových násobení. Počet násobení sa dá minimalizovať využitím nasledujúcej vlast-

nosti štvorcových matíc: každá dvojica matíc sa dá vynásobiť nezávisle od seba a tieto násobenia (keďže každá matica je taká istá) dajú rovnaký výsledok. Teda stačí nám spraviť jedno vynásobenie a úloha umocnenia matice A na n-tú sa zredukuje na úlohu umocnenia matice A^2 na $\lfloor \frac{n}{2} \rfloor$-ú. V prvom kroku teda treba vynásobiť A so sebou. V druhom kroku už výsledok A^2 so sebou a nie A^2 s A a ešte raz s A, teda využitím výsledku z predchádzajúceho násobenia sa počet činiteľov zredukuje vždy na polovicu. Pritom si musíme uvedomiť, že v prípade, keď n nie je párne na niektorej úrovni teda počet matíc - činiteľov - je nepárny, tak zostáva práve jedna matica bez páru (A). (Majme teda $2k + 1$ matíc na niektorej úrovni počas zredukovania činiteľov. Všetkých činiteľov môžeme zoradiť do k dvojíc a zostáva ešte jedna matica bez páru. Pre k párov dostanem výsledok jedným vynásobením a poslednú maticu si budem pamätať. Na najbližšej úrovni, kde počet činiteľov bude zase nepárny poslednú maticu z tej úrovne vynásobím s poslednou maticou z predchádzajúcej úrovne kde bol počet činiteľov nepárny a budem si pamätať tento výsledok a tento výsledok budem v ďalšom násobiť zase s nepáriteľnym členom. Na konci už len stačí vynásobiť takto získaný výsledok s maticou čo sme dosiahli postupným zredukovaním činiteľov.

Vlastne sme využili asociatívnosť násobenia.

Tento algoritmus sa ľahko reprezentuje na obrázku:



Umocnenie(A,n)
$aktexp \leftarrow n$
$B \leftarrow I$
while $aktexp > 1$ **do**
  if $aktexp$ **je nepárne then**
    $B \leftarrow A * B$
  end if
  $A \leftarrow A^2$
  $aktexp \leftarrow \lfloor \frac{aktexp}{2} \rfloor$
end while

return $A * B$

Tvrdenie 3.0.1. *Uvedený algoritmus používa* $O(\log_2 n)$ *maticových násobení, a stačí nám ukladať v pamäti dve matice veľkosti* $k \times k$*, kde* k *je veľkosť vstupnej štvorcovej matice.*

Dôkaz. Algoritmus obsahuje jediný cyklus uvedenej dĺžky. Pamäťová zložitosť je tiež zrejmá. Korektnosť algoritmu je tiež zrejmá, ľahko sa dá dokázať, že po k-tom kroku (počítané od 0) matica A reprezentuje maticu $A = A^{2^k}$ a matica B reprezentuje $B = A^l$, kde $n = a.2^{k+1} + 2^k + l$. □

V súbore **algoritmus6.cpp** je implementovaný uvedený algoritmus v C++ pomocou knižnice NTL. Na násobenie matíc som použil Strassenov algoritmus, ktorý som implementoval podľa popisu z [3].

Kapitola 4

Je dané číslo členom lineárnej rekurentnej postupnosti

Ďalšou zaujímavou otázkou je, či dané číslo patrí do Fibonacciho postupnosti, alebo nie. V článkoch [7] a [4] sa autori rozsiahle zaoberajú touto otázkou. Ich výsledky potvrdzujú, že dané číslo n je členom Fibonacciho postupnosti vtedy a len vtedy, keď aspoň jedno z dvoch čísiel $\sqrt{5n^2 \pm 4}$ je celé číslo. V tejto kapitole dokážeme podobný vzťah pre širšiu množinu rekurentných postupností. Nasledujúci postup sa opiera na horeuvedené články, prebrali sme niektoré črty dôkazov a postupne sme zovšeobecnili v nich použité vzťahy.

Chceli by sme dodať, že nasledujúce výsledky sa už nachádzajú v článku [5], pričom my sme tento článok našli len po dokázaní našich výsledkov, čo potvrdzuje aj trošku iný postup dôkazu.

4.1 Nutná a postačujúca podmienka

Vhodná postupnosť na zovšeobecnenie daného výrazu je postupnosť P_n, pričom vhodné podmienky na konštanty d a c sa určia neskôr. Ako prvú vec si môžeme ustanoviť, že sa budeme zaoberať len postupnosťami celých čísel, čiže takými postupnosťami, pre ktoré c a d sú celé čísla. Pre naše účely je nutné dodefinovať jednu novú postupnosť:

$$P_{n+1} = cP_n + dP_{n-1}; \qquad P_0 = 0, P_1 = 1$$

$$L_n = P_{n+1} + dP_{n-1}; \qquad L_0 = 2, L_1 = c$$

Postupnosť L_n je zovšeobecnením Lucasovej postupnosti. Pre naše účely nám stačí si uvedomiť len toľko, že táto postupnosť je postupnosť celých čísel.

Tvrdenie 4.1.1.

$$-(-d)^{n-1} = -P_n^2 + P_{n+1}P_{n-1}$$

Dôkaz. Na dokázanie tohto tvrdenia použijeme maticovú rovnosť

$$\begin{pmatrix} 0 & 1 \\ d & c \end{pmatrix}^{n-1} \begin{pmatrix} 0 & 1 \\ 1 & c \end{pmatrix} = \begin{pmatrix} P_{n-1} & P_n \\ P_n & P_{n+1} \end{pmatrix}$$

z lemy 1.2.8. Po prepísaní matíc na determinanty môžeme využiť fakt, že determinant umocnenej matice je determinant základnej matice umocnený na korešpondujúcu mocninu. Teda dostaneme:

$$\begin{vmatrix} 0 & 1 \\ d & c \end{vmatrix}^{n-1} \begin{vmatrix} 0 & 1 \\ 1 & c \end{vmatrix} = \begin{vmatrix} P_{n-1} & P_n \\ P_n & P_{n+1} \end{vmatrix}$$

$$\begin{vmatrix} 0 & 1 \\ d & c \end{vmatrix} = -d \qquad \begin{vmatrix} 0 & 1 \\ 1 & c \end{vmatrix} = -1$$

$$\begin{vmatrix} P_{n-1} & P_n \\ P_n & P_{n+1} \end{vmatrix} = P_{n-1}P_{n+1} - P_n^2$$

Po dosadení hodnôt determinantov do rovnice dostaneme žiadanú rovnosť. □

Lema 4.1.2. *Pre všetky členy postupnosti P_n platí $L_n^2 = P_n^2(c^2+4d)+4(-d)^n$.*

Dôkaz. Vieme, že platia nasledujúce rovnice:

$$\begin{aligned} L_n &= P_{n+1} + dP_{n-1} \\ -(-d)^{n-1} &= -P_n^2 + P_{n+1}P_{n-1} \end{aligned}$$

$$\begin{aligned} L_n &= P_{n+1} + dP_{n-1} \qquad /()^2 \\ -(-d)^{n-1} &= -P_n^2 + P_{n+1}P_{n-1} \qquad / * (-4d) \end{aligned}$$

$$\begin{aligned} L_n^2 &= P_{n+1}^2 + d^2P_{n-1}^2 + 2dP_{n+1}P_{n-1} \\ -4(-d)^n - 4dP_n^2 &= -4dP_{n+1}P_{n-1} \end{aligned}$$

Po sčítaní týchto rovníc dostaneme:

$$\begin{aligned} L_n^2 - 4(-d)^n - 4dP_n^2 &= P_{n+1}^2 + d^2P_{n-1}^2 - 2dP_{n+1}P_{n-1} \\ &= (P_{n+1} - dP_{n-1})^2 \\ &= c^2P_n^2 \end{aligned}$$

Čo sme chceli dokázať. □

Keďže ani hodnotu L_n, ani hodnotu n nepoznáme, túto všeobecnú rovnicu musíme upresniť. Keď si zvolíme vhodne hodnotu d, tak by sme mohli $4(-d)^n$ odhadnúť. Teda pribudne naša prvá podmienka:

$$\boxed{d = \pm 1}.$$

Pomocou tejto podmienky nie je pre nás potrebné, aby sme n určili.

Veta 4.1.3. *Pre všetky členy postupnosti P_n platí, že uvedený výraz je celé číslo:*

$d = 1$	$d = -1$
$\sqrt{P_n^2(c^2+4)+4(-1)^n}$	$\sqrt{P_n^2(c^2-4)+4}$.
	Tu pribudne podmienka $\boxed{c^2 > 4}$.

Dôkaz. Táto veta priamo vyplýva z predchádzajúcej lemy. Po dosadení uvedených hodnôt namiesto d postupne dostaneme:

$L_n^2 = P_n^2(c^2+4)+4(-1)^n$	$L_n^2 = P_n^2(c^2-4)+4$
$L_n = \sqrt{P_n^2(c^2+4)+4(-1)^n}$	$L_n = \sqrt{P_n^2(c^2-4)+4}$.

Keďže číslo L_n na ľavej strane rovnice je celé číslo aj na pravej musíme dostať celé číslo. □

Teraz, aby sme dokázali opačnú implikáciu najprv uvedieme takú rovnicu s dvoma neznámymi, ktorej diskriminant sa rovná vzorcom zo znenia predchádzajúcej vety.

$$y^2 - cxy - x^2 = \pm 1, \qquad (4.1)$$
$$(D = c^2x^2 + 4x^2 \pm 4 = x^2(c^2+4) \pm 4)$$

$$y^2 - cxy + x^2 = 1 \qquad (4.2)$$
$$(D = c^2x^2 - 4x^2 + 4 = x^2(c^2-4) + 4)$$

Lema 4.1.4. *Dvojica (P_n,P_{n+1}) $\forall n \in \mathbb{N}$ rieši rovnicu*

$y^2 - cxy - x^2 = \pm 1$,	$y^2 - cxy + x^2 = 1$
pre $d = 1$.	*pre* $d = -1$.

Dôkaz. Dokážme jedno všeobecnejšie tvrdenie:
Dvojica (P_n,P_{n+1}) $\forall n \in \mathbb{N}$ rieši rovnicu

$$y^2 - cxy - dx^2 = (-d)^{n-1}.$$

Potom po dosadení uvedených hodnôt namiesto d dostaneme špeciálne prípady rovnosti, uvedené v znení lemy. V tvrdení 4.1.1 sme dokázali rovnosť

$$-(-d)^{n-1} = -P_n^2 + P_{n+1}P_{n-1}.$$

Po postupných úpravách dostaneme:

$$\begin{aligned} -(-d)^{n-1} &= -P_n^2 + P_{n+1}P_{n-1} \\ &= -P_n^2 + (cP_n + dP_{n-1})P_{n-1} \\ &= -P_n^2 + cP_nP_{n-1} + dP_{n-1}P_{n-1} \\ (-d)^{n-1} &= P_n^2 - cP_nP_{n-1} - dP_{n-1}^2 \end{aligned}$$

Čo sme chceli dokázať. □

Lema 4.1.5. *Ak dvojica $(x, y) \in \mathbb{N}^2$ je riešením rovnice*

$$y^2 - cxy - x^2 = \pm 1, \qquad y^2 - cxy + x^2 = 1$$

pre $d = 1$, *pre $d = -1$ a $c^2 > 4$,*

tak $(x, y) = (P_n, P_{n+1})$ pre nejaké $n \in \mathbb{N}$.

Dôkaz. Lemu dokážeme pomocou indukcie podľa $(x+y)$. Aby takáto indukcia sa dala používať potrebujeme zabezpečiť dobre zadefinovaný indukčný krok. Najprv overme, že za podmienky $c > 0$ postupnosť $(P_n)_{n=1}^{\infty}$ bude rastúca indukciou vzhľadom na n. Báza sa ľahko overí dosadením konkrétnych hodnôt, preto tu uvedieme len indukčný krok.

$\boxed{c > 0}$

$P_{n+1} = cP_n + P_{n-1} > cP_n > P_n$

Odvodenie je samovysvetlujúce.

V tejto časti $c > 0$ a $c^2 > 4$ spoločne dajú $\boxed{c > 2}$

$P_{n+1} = cP_n - P_{n-1} > (c-1)P_n > P_n$

Tu sme využili indukčný krok, keď sme odhadli, že $P_n > P_{n-1}$, teda $cP_n - P_{n-1} = (c-1)P_n + (P_n - P_{n-1}) > (c-1)P_n$.

Musíme ešte dokázať, že v prvom prípade $y - cx$ a v druhom prípade $cx - y$ je prirodzené číslo.

$$\begin{aligned} y^2 - cxy - x^2 &= \pm 1 \\ y(y - cx) &= x^2 \pm 1 \end{aligned}$$

Na ľavej strane rovnice y je kladné, na pravej strane $x^2 \pm 1$ je nezáporné

$$\begin{aligned} y^2 - cxy + x^2 &= 1 \\ y(y - cx) &= -x^2 + 1 \\ y(cx - y) &= x^2 - 1 \end{aligned}$$

Na ľavej strane rovnice y je kladné, na pravej strane $x^2 - 1$ je nezáporné

celé číslo okrem prípadu $x = 0$, teda aj $(y - cx)$ musí byť prirodzené číslo.

celé číslo okrem prípadu $x = 0$, teda aj $(cx - y)$ musí byť prirodzené číslo.

[1] [2] Keď už máme všetko pripravené môžeme dokázať lemu pomocou úplnej indukcie vzhľadom na $(x+y)$. Bázu $x+y = 1$ môžeme overiť dosadením konkrétnych hodnôt. Máme dve možnosti $x = 0$ a $y = 1$ alebo $x = 1$ a $y = 0$. Tieto čísla sú aj členom postupnosti $(P_n)_{n=1}^{\infty}$. Najprv dokážem, že sa nám stačí v indukcií zaoberať len s prípadom $y > x$ (V ostatných prípadoch dostávame len triviálne riešenia, alebo podobné k prípadu $y > x$).
$x = y$:

$$\begin{aligned} y^2 - cy^2 - y^2 &= \pm 1 \\ -cy^2 &= \pm 1 \\ y^2 &= \pm\frac{1}{c} \end{aligned}$$

To sa môže nastať výlučne vtedy, keď $c = 1$, čo je Fibonacciho postupnosť. Vtedy $x = 1$ a $y = 1$ je riešením rovnice a aj členmi postupnosti P_1, P_2.

$$\begin{aligned} y^2 - cy^2 + y^2 &= 1 \\ y^2(2 - c) &= 1 \\ y^2 &= \frac{1}{2 - c} \end{aligned}$$

Z predpokladu $c > 2$ plynie, že $2 - c < 0$, teda aj $y^2 = \frac{1}{2-c} < 0$ čo je v rozpore s predpokladom, že $y \in \mathbb{N}$. Teda rovnica nemá riešenie.

$x > y$:

$$y^2 - cxy - x^2 < y^2 - cy^2 - y^2 = \\ = -cy^2 < -c$$

Keďže $c > 0$ takto máme $y^2 - cxy - x^2 < -c < -1$, teda ľavá strana rovnice sa nemôže rovnať pravej.

Rovnica je zrejme symetrická, teda všetky riešenia sú aj riešeniami, keď $y > x$.

$y > x$: Teraz spravme už spomínaný indukčný krok. Predpokladajme, že (x, y) rieši danú rovnicu.

Dosadením dokážme, že aj $(a, b) = (y - cx, x)$ rieši rovnicu.

$$b^2 - cab - a^2 = \\ = x^2 - c(y - cx)x - (y - cx)^2 = \\ = x^2 + cxy - y^2 = \pm 1$$

Dosadením dokážme, že aj $(a, b) = (cx - y, x)$ rieši rovnicu.

$$a^2 - cba + b^2 = \\ = (cx - y)^2 - cb(cx - y) + x^2 = \\ = x^2 - cyx + y^2 = 1$$

[1] Pre $x = 0$; $y - cx = y - c0 = y$, za predpokladu, že $y \in \mathbb{N}$ dostaneme, že aj $y - cx \in \mathbb{N}$.

[2] V oboch prípadoch si môžeme uvedomiť, že uvedené vzorce budeme potrebovať na uskutočnenie indukčného kroku. Pritom ale vieme, že keď $x = 0$, tak $y = 1$, aby dvojica (x, y) riešila rovnicu zo znení lemy. A práve $x + y = 1$, teda $x = 0$ a $y = 1$ (lebo $y > x$) bude báza indukcií, teda prípad $x = 0$ v tejto častí dôkazu môžeme zanedbať.

Potom ale podľa indukčného predpokladu $a = P_n$ a $b = P_{n+1}$ pre niektoré n. Ale potom $x = P_{n+1}$ a $y = a + cx = P_n + cP_{n+1} = P_{n+2}$.

Potom ale podľa indukčného predpokladu $a = P_n$ a $b = P_{n+1}$ pre niektoré n. Ale potom $x = P_{n+1}$ a $y = -a + cx = -P_n + cP_{n+1} = P_{n+2}$.

Aby sme mohli použiť indukčný predpoklad pre dvojicu (a, b) musime ešte dokázať, že $a + b < x + y$.

$a + b = y - cx + x$ pre $x > 0$; $y + x - cx < y + x$.

Vieme, že $a + b = cx - y + x$ a $y(cx - y) = x^2 - 1$. Teda $a + b = \frac{x^2-1}{y} + x$, preto nám stačí dokázať nerovnosť $x^2 - 1 < y^2$. Platí ale $y > x$, teda $y^2 > x^2$.

Všetky možnosti sme prebrali, teda uvedená lema je platná. □

A teraz môžeme sformulovať nutnú a postačujúcu podmienku na to, aby sme mohli určiť či je nejaké číslo členom postupnosti $(P_n)_{n=1}^{\infty}$:

Veta 4.1.6. *Číslo $x \in \mathbb{N}$ je členom postupnosti $(P_n)_{n=1}^{\infty}$ vtedy, a len vtedy keď*

$(4 + c^2)x^2 \pm 4$ *pre* $d = 1$, $c > 0$

$(c^2 - 4)x^2 + 4$ *pre* $d = -1$, $c > 2$

je druhou mocninou prirodzeného čísla.

Dôkaz. Prvá implikácia tejto vety priamo vyplýva z vety 4.1.3. Teda stačí nám dokázať, že ak čísla uvedeného tvaru sú druhou mocninou nejakého celého čísla, tak číslo x je členom danej postupnosti.

Dokážme, že ak $(4+c^2)x^2 \pm 4$ má druhú odmocninu v množine celých čísel, tak $\exists y \in \mathbb{N}$ také, že (x, y) bude riešiť rovnicu (4.1). Napíšme známy vzorec pre rovnicu (4.1):

$$y = \frac{cx \pm \sqrt{(-cx)^2 + 4x^2 \pm 4}}{2}$$

$$y = \frac{cx \pm \sqrt{(c^2 + 4)x^2 \pm 4}}{2}$$

Vieme, že $\sqrt{(c^2 + 4)x^2 \pm 4}$ je celé číslo, označme ho $n := \sqrt{(c^2 + 4)x^2 \pm 4}$. Teda $y = \frac{cx \pm n}{2}$. Teraz nám stačí dokázať, že $cx \pm n$ je v každom prípade párne, teda cx a n majú totožnú paritu. Pričom po úpravách

Tento prípad sa dokáže analogicky, pomocou rovnice (4.2).

dostaneme, že $n^2 = (c^2+4)x^2 \pm 4 = (cx)^2 + 4x^2 \pm 4$. Keď cx je párne, tak n^2 bude deliteľné 4, teda po odmocnení n bude párne. Zjavne, keď cx je nepárne, tak aj n^2 bude nepárne, teda po odmocnení ani n nebude párne.
Dokázali sme, že $\exists y \in \mathbb{N}$ také, že (x, y) rieši rovnicu (4.1), teda podľa lemy 4.1.5 vieme, že je členom postupnosti P_n.

Z čoho vyplýva platnosť tejto vety. □

Ešte by sa mali vyšetriť prípady $c = 1$ a $c = 2$ v prípade, že $d = -1$.

Veta 4.1.7. *Postupnosť $P_n = 2P_{n-1} - P_{n-2}$, $P_0 = 0$, $P_1 = 1$ je v uzavretom tvare $P_n = n$.*

Dôkaz. Matematickou indukciou. Pre $n = 0$ a $n = 1$ tvrdenie priamo plynie z definície uvedenej postupnosti. Teraz nám stačí dokázať indukčný krok:

$$P_n = 2P_{n-1} - P_{n-2} = 2(n-1) - (n-2) = 2n - 2 - n + 2 = n$$

□

Poznámka 4.1.8. Z predchádzajúcej vety vyplýva, že každé prirodzené číslo je členom uvedenej postupnosti, a žiadne iné číslo nemôže byť členom tejto postupnosti.

Veta 4.1.9. *Postupnosť $P_n = P_{n-1} - P_{n-2}$, $P_0 = 0$, $P_1 = 1$ je periodická. Perióda je $0, 1, 1, 0, -1, -1$.*

Dôkaz. Priamym výpočtom.
Po sebe idúce členy postupnosti sú: $0, 1, 1, 0, -1, -1, 0, 1$. Keďže P_n je rekurencia druhého rádu každý člen postupnosti je jednoznačne dané pomocou predchádzajúcich dvoch členov postupnosti. Keď sa niekde vyskytuje tá istá dvojica predchádzajúcich členov, tak sa rekurencia zacyklí. □

Poznámka 4.1.10. Z predchádzajúcej vety vyplýva, že každý člen uvedenej postupnosti je z množiny $\{-1, 0, 1\}$.

4.2 Algoritmus na overenie nutnej a postačujúcej podmienky

Prvá možnosť, ako overiť nutnú a postačujúcu podmienku je, že vyrátame presné hodnoty $\sqrt{(4+c^2)x^2 \pm 4}$ alebo $\sqrt{(c^2-4)x^2+4}$ a otestujeme či výsledok týchto aritmetických operácií je celé číslo. Táto metóda má svoje nedostatky, tieto nedostatky sa týkajú operácie odmocnenia. Prvým takým nedostatkom je fakt, že odmocnenie je pomerne zložitou aritmetickou operáciou. Druhým takým nedostatkom je, že odmocnina sa nedá vypočítať presne, používajú sa tam isté zaokruhlenia, ktoré by sa nemali pri testovaní zanedbať. Lepším riešením by bola implementácia algoritmu, ktorý pracuje iba s celými číslami. Táto požiadavka je oprávnená, keďže úlohou je overiť, či nejaké celé číslo má, alebo nemá celočíselnú druhú odmocninu. Vstupy musia byť celé čísla, ináč by daný problém nemal zmysel takto riešiť.

V ďalšom teda budeme riešiť tento problém: či dané celé číslo z má celočíselnú druhú odmocninu. Ako prvú vec si môžeme uvedomiť, že v prípade niektorých hodnôt z vieme "rýchlo" vylúčiť existenciu celočíselnej odmocniny. To sa dá spraviť nasledovne: postupným delením dvojkou dostaneme najväčšieho deliteľa čísla z v tvare 2^i. Potom číslo z je tvaru $z = 2^i z_1$, kde z_1 nie je už deliteľné dvojkou. Podľa vzorca $\sqrt{z} = \sqrt{2^i z_1} = \sqrt{2^i}\sqrt{z_1}$ číslo z má presne vtedy celočíselnú druhú odmocninu, keď 2^i a z_1 majú tiež. Je zrejmé, že číslo 2^i má presne vtedy celočíselnú druhú odmocninu, keď i je párne číslo. Teda môžeme vylúčiť tie hodnoty z, pri ktorých takto definovaná hodnota i nie je párna. Pre ostatné hodnoty z nám stačí overiť, či z_1 má celočíselnú druhú odmocninu. Ešte by sme chceli dodať, že tento spôsob vylúčenia je rýchly preto, lebo delenie dvoma je jedna z najrýchlejších operácií - operácia SHIFT.

Rýchle sa dajú vylúčiť aj čísla tvaru $4k+3$, $8k+3$, $8k+5$ a $8k+7$. Dôkaz uvedieme pre čísla tvaru $4k+3$ a $8k+5$, keďže čísla tvaru $8k+3$ a $8k+7$ sa dokážu analogicky, ako $4k+3$. Dôkaz spravíme sporom. Predstavme si, že existuje také číslo $a \in \mathbb{N}$, že $a^2 = 4k+3$. Vieme teda, že a^2 nie je párne číslo, preto ani a nemôže byť párne. Teda číslo a má tvar $a = 2l+1$. Teda platí

$$\begin{aligned} a^2 = (2l+1)^2 &= 4l^2 + 4l + 1 \\ 4l^2 + 4l + 1 &= 4k + 3 \\ 2l^2 + 2l &= 2k + 1 \end{aligned}$$

čím sme dostali spor, keďže číslo na ľavej strane rovnici je párne, a číslo na pravej strane rovnici je nepárne. Po podobných úvahách dostaneme pre prí-

pad $8k + 5$ nasledujúce odvodenie:

$$\begin{aligned} 4l^2 + 4l + 1 &= 8k + 5 \\ l^2 + l &= 2k + 1 \\ l(l + 1) &= 2k + 1 \end{aligned}$$

čiže na ľavej strane rovnici máme znovu párne číslo, na pravej nepárne.

V ďalšom popíšeme algoritmus na zistenie celočíselnej druhej odmocniny celého čísla; potom tento algoritmus skombinujeme s už horeuvedeným algoritmom na rýchle vylúčenie niektorých vstupov a takýmto spôsobom získame pseudokód riešiaci problém tejto kapitoly. Teda chceme zistiť, či číslo z_1 má celočíselnú druhú odmocninu pomocou iterácie celých čísel. Na tento problém sme upravili algoritmus binárneho vyhľadávania. Na začiatku si zvolíme všetky čísla od 0 po z_1, ako množinu kandidátov na druhú odmocninu. V každom kroku postupne tento interval znížime o polovicu nasledujúcim spôsobom: vyberieme stredný prvok intervalu. Umocníme ho na druhú. (Táto operácia je rýchlejšia, ako odmocnenie čísla.) Potom otestujeme, či je aktuálna druhá mocnina väčšia ako z_1. Ak je väčšia, tak si zvolíme dolnú polovicu intervelu za nových možných kandidátov. V opačnom prípade si zvolíme hornú polovicu intervalu za novú množinu kandidátov na druhú odmocninu. Takýmto spôsobom sa dá zredukovať interval na triviálnu dĺžku napríklad na menej ako 4 čísla. Potom úplným preberaním možností zistíme, či medzi kandidátmi je celočíselná odmocnina. Keď sme už našli druhú odmocninu, tak vieme, že z_1 má celočíselnú druhú odmocninu, v opačnom prípade nemá. Tento algoritmus sa opiera o fakt, že $x^2 > y^2 \Leftrightarrow x > y$.

Overenie(c,d,x)
if $d = -1$ **then**
  $z \leftarrow (c^2 - 4)x^2 + 4$
else
  $z \leftarrow (4 + c^2)x^2 \pm 4$
end if
$z_1, i \leftarrow z = 2^i z_1$
if i **je nepárne then**
  return $FALSE$
end if
if z **je v tvare** $4k + 3$**,** $8k + 3$**,** $8k + 5$ **alebo** $8k + 7$ **then**
  return $FALSE$
end if
$dolna\_hranica \leftarrow 0$
$horna\_hranica \leftarrow z_1$
while $horna\_hranica - dolna\_hranica + 1 > 3$ **do**

$stredny \leftarrow (horna\_hranica + dolna\_hranica)/2$
if $stredny^2 > z_1$ **then**
$horna\_hranica \leftarrow stredny - 1$
else
$dolna\_hranica \leftarrow stredny$
end if
end while
return $TRUE \Leftrightarrow \exists n \in \langle dolna\_hranica, horna\_hranica \rangle, n^2 = z_1$

Pravdivosť, časová zložitosť a pamäťová náročnosť algoritmu priamo vyplýva z uvedených matematických tvrdení. Časová zložitosť algoritmu je $O(\log n)$, vzhľadom na operáciu druhej mocniny. V pamäti si musíme pamätať konštantne veľa čísel.

Algoritmus je implementovaný v súbore **algoritmus7.cpp** v jazyku C++ pomocou NTL.

Záver

Ako prvú vec treba pripomenúť, že v tejto práci používané označenie na zložitosť algoritmov $O(\log_2 n)$ je totožné s $O(\log n)$. Používaním dvojky sme len chceli zvýrazniť, že uvedené algoritmy sú založené na delení, alebo násobení dvojkou, teda na tom, že v každom kroku sa vynechá polovica členov z predchádzajúceho kroku, alebo sa vylúči polovica ešte existujúcich možností.

Ďalej treba poznamenať, že algoritmy a vety z prvej kapitoly fungujú aj pre reálne čísla, teda podľa potreby uvedené algoritmy sa dajú prepísať z množiny celých čísel na množinu reálnych čísiel. Samozrejme v štvrtej kapitole sa o reálnych číslach hovoriť nemá zmysel.

Ako sme ukázali v druhej kapitole, algoritmus z prvej kapitoly nie je najlepším existujúcim algoritmom pre výpočet n-tého člena konkrétnej postupnosti. Pre konkrétnu postupnosť sa dá vymyslieť aj lepší, rýchlejší algoritmus.

Ako poslednú vec chcem ešte raz poďakovať každému, kto mi pomohol, pri písaní tejto práce.

Literatúra

[1] BACH, E. - SHALLIT, J. 1997. *Algorithmic Number Theory - Volume I: Eficient Algorithms*. London: The MIT Press, 1997. ISBN 0-262-02405-5.

[2] BERNSTEIN, D. J. 1998. Detecting perfect powers in essentially linear time. *Mathematics of Computation*. 1998, roč. 67, č. 223, s. 1253–1283.

[3] ĎURIŠ, P. 2009. *Tvorba efektívnych algoritmov*. Bratislava: Knižničné a edičné centrum FMFI UK, 2009. ISBN 978-80-89186-50-1.

[4] GESSEL, I. 1972. Fibonacci is square (problem and solution H-187). *Fibonacci Quarterly*. 1972, roč. 10, č. 4, s. 417.

[5] HOGGATT, V. E., Jr. - BICKNELL-JOHNSON, M. 1978. A primer for the Fibonacci numbers xvii. *Fibonacci Quarterly*. 1978, roč. 16, č. 2, s. 130-137. Dostupné na internete: `http://www.fq.math.ca/Scanned/16-2/hoggatt.pdf`.

[6] HONSBERGER, R. 1985. *Mathematical Gems III*. USA: The Mathematical Association of America, 1985. ISBN 0-88385-300-0

[7] JAMES, P. 2009. *When is a number Fibonacci?* [online]. Swansea: Department of Computer Science, Swansea University, January 25, 2009. Dostupné na internete: `www.cs.swan.ac.uk/~csulrich/ftp/PhilJames1008.pdf`.

[8] JOHNSON, L. F. 2010. *Golden an Alternating, fast simple O(lg n) algorithms for Fibonacci*. [online]. Waterloo: University of Waterloo, 2010. Dostupné na internete: `http://arxiv.org/abs/1011.0148`.

[9] JONES, J. P. 2003. Representation of solutions of Pell equations using Lucas sequences. *Sectio Mathematicae*. 2003, č. 30, s. 75-86.

[10] KALMAN, D. - MENA, R. 2003. The Fibonacci Numbers – Exposed. *Math. Mag.* 2003, roč. 76, č. 3, s. 167-181.

[11] KESKIN, R. 2010. Solutions of some quadratic Diophantine equations. *Computers and Mathematics with Applications*. 2010, roč. 60, č. 8, s. 2225-2230.

[12] KING, M. F. 2008. *Fibonacci Numbers and Mathematical Phyllotaxis*. Durham: Department of Mathematics, University of Durham, 2008. Bakalárska práca.

[13] KISS, P. 1979. Diophantine representation of generalized Fibonacci numbers. *Elemente der Mathematik*. 1979, roč. 34, č. 6, s. 129-132.

[14] SIMONS, C. S. - WRIGHT, M. 2006. Fibonacci imposters. *International Journal of Mathematical Education in Science and Technology*. 2006, roč. 38, č. 5, s. 677-682.

[15] ŠTUPÁKOVÁ, B. 2008. *Fibonacciho a Lucasove čísla*. Bratislava: FMFI UK, 2008. Bakalárska práca.

[16] TAKAHASHI, D. 2000. A fast algorithm for computing large Fibonacci numbers. *Information Processing Letters*. 2000, roč. 75, č. 6, s. 243-246.