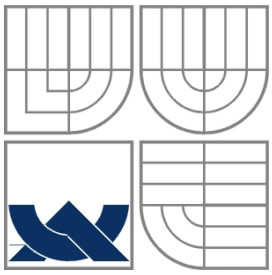
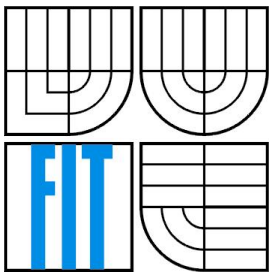


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SLOVNÍKOVÉ METODY KOMPRESE DAT

DICTIONARY METHODS OF DATA COMPRESSION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER KUBICA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. DAVID BAŘINA

BRNO 2010

Abstrakt

Zvyšující se množství ukládaných a přenášenych dat má za následek potřebu komprese. Pro tyto účely byly vytvořeny mnohé kompresní postupy. Tato práce sa zaměřuje na bezestrátovou kompresi, konkrétně na slovníkové metody. Jsou tu shrnuty poznatky o metodách LZ77 a LZ78. Tyto metody byly pro potřeby této práce implementovány podle jejich formálního popisu v jazyce C++. Výsledkem je podrobná znalost jejich principů, výhod i nevýhod a také možných vylepšení.

Abstract

Due to increasing amount of stored and transferred data, there is a need of compression. For these purposes, there were invented many compressing algorithms. This thesis focuses on lossless compression, specifically on dictionary methods. This work sums up the knowledge about methods LZ77 and LZ78.. These methods were implemented in C++ language, according to their formal description. Detailed knowledge of their principles, advantages, disadvantages and possible improvements is the product.

Klíčová slova

komprese, dekomprese, bezestrátová komprese, slovníkové metody, LZ77, LZ78

Keywords

compression, decompression, loosless compression, dictionary methods, LZ77, LZ78

Citace

Peter Kubica: Slovníkové metody komprese dat, bakalářská práce, Brno, FIT VUT v Brně, 2010

Slovníkové metody komprese dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Davida Bařiny. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Peter Kubica
15. 5. 2010

Poděkování

Chcel by som sa poďakovať pánovi Ing. Davidovi Bařinovi za vedenie tejto bakalárskej práce a cenné rady pri jej riešení.

Obsah

Obsah....1
1 Úvod....2
2 Kompresia dát....3
2.1 Univerzálna kompresia dát....4
2.2 Kompresná výkonnosť....5
3 Slovníkové metódy....6
3.1 Jednoduchá slovníková kompresia....7
3.2 LZ77....8
3.2.1 Popis....8
3.2.2 Princíp....8
3.2.3 Príklad....9
3.2.4 Súhrn a vylepšenia....10
3.2.5 LZSS....10
3.3 LZ78....11
3.3.1 Popis....11
3.3.2 Princíp....11
3.3.3 Príklad....12
3.3.4 Súhrn a vylepšenia....13
4 Implementácia....14
4.1 LZ77....14
4.1.1 Príklad použitia knižnice LZ77....14
4.1.2 Diagram tried....15
4.2 LZ78....16
4.2.1 Diagram tried....17
4.2.2 Príklad použitia knižnice LZ78....17
5 Testovanie a výsledky....18
5.1 LZ77....19
5.1.1 Test kompresie rôznych typov súborov....19
5.1.2 Testy závislostí od dĺžky bufferov....19
5.1.3 Porovnanie trvania kompresie a dekompresie....22
5.2 LZ78....22
5.2.1 Test kompresie rôznych typov súborov....22
5.2.2 Test závislostí od veľkosti slovníka....23
5.2.3 Porovnanie trvania kompresie a dekompresie....24
5.3 Porovnanie pôvodných slovníkových metód so súčasnými....24
6 Záver....26
Literatúra....27
Zoznam príloh....28

1 Úvod

Narastajúce nároky na zálohovanie údajov, zvyšujúce sa objemy dát pripadajúce na jedného používateľa, alebo potreba prenosu údajov po sieti. Tieto príklady sú len malou vzorkou z veľkého množstva faktorov, ktoré zvyšujú potrebu kompresie dát. Tá nachádza uplatnenie v širokom spektre oblastí. Je využitá napríklad v telekomunikáciách, kde je obmedzená dátová priepustnosť a tak je potrebné komprimovať hovor.

Kódovaním pomocou kompresného algoritmu sa zo súboru odstránia nadbytočné údaje. Ak sú údaje odstránené natrvalo, ide o stratovú kompresiu. Jej opakom je bezstratová kompresia, po ktorej je možné obnoviť pôvodný súbor. Tento postup musí spĺňať základnú požiadavku, ktorou je dosiahnutie požadovanej rýchlosti pri zachovaní kompresného pomeru.

Úlohou tejto práce je priblíženie tejto problematiky. Rozdelená je do niekoľkých častí, pričom prvá obsahuje vysvetlenie základných pojmov a princípov kompresie dát. O túto kapitolu sa opiera ďalšia, ktorá približuje vlastností slovníkových metód. Tie patria do kategórie bezstratovej kompresie dát. Zameriava sa na metódy LZ77 a LZ78, ktoré som si zvolil k popisu a zároveň pre potreby tejto práce implementoval. A práve popis implementácie je obsiahnutý v nasledovnej kapitole. Testovaním metód je možné zistiť ich hlavné výhody a nedostatky a následne vyvodiť závery o vhodnosti ich použitia. Takto získané poznatky sú zhrnuté v záverečnej kapitole.

2 Kompresia dát

Kompresia je proces, v ktorom sa vstupný tok prekóduje na výstupný tok, za účelom zmenšiť dátový tok. Zmenšenie dátového toku sa môže dosiahnuť redukciou dát alebo odstránením redundancie v údajoch. To je vhodné pri obmedzenej kapacite dostupnej pamäte, alebo pri obmedzenej prenosovej šírke pásma (internet). Kódovaním rozumieme proces vzájomného priraďovania symbolov z jednej množiny do druhej.

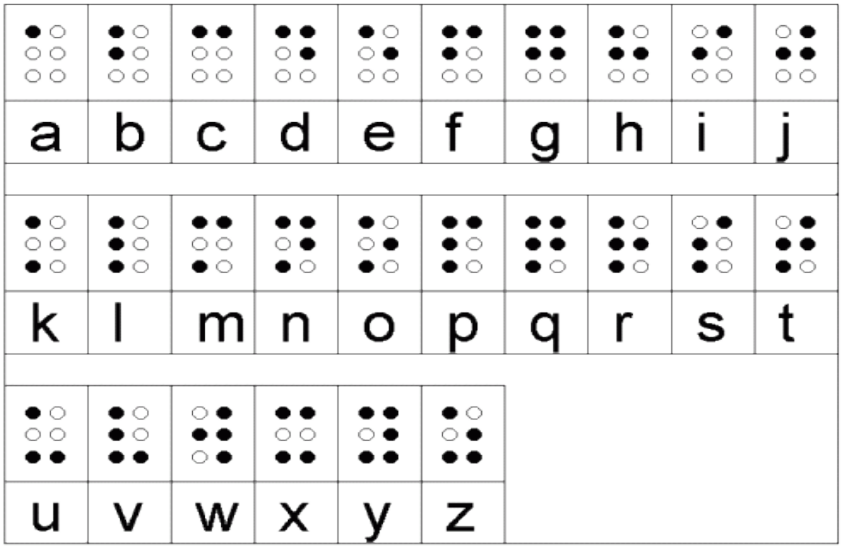
Základným princípom kompresie znižovaním redundancie je reprezentácia často vyskytujúcich sa symbolov resp. fráz kratšími kódmi a zriedkavo vyskytujúcich sa symbolov dlhými kódmi.

Rozlišujeme stratovú a bezstratovú kompresiu podľa toho, či je možne komprimované dáta dekomprimovať do pôvodnej podoby. Stratová kompresia sa využíva pri kompresii zvuku, obrazov a videa, kde sa pri kompresii dá tolerovať vypustenie určitých dát. Naopak pri kompresii textu je strata čo i len jediného bitu neakceptovateľná, preto sa komprimujú bezstratovými metódami.

O symetrickú kompresiu sa jedná, keď kompresor i dekompresor, pracujú s rovnakým algoritmom, len opačným postupom. Preto kompresia a dekompresia trvá približne rovnakú dobu. Pri asymetrických kompresných metódach pracuje kompresor iným postupom ako dekompresor a preto buď kompresia alebo dekompresia trvá dlhšie. To však nemusí byť na škodu. Pri archivácii, kde dochádza k občasnej kompresii ale častej dekompresii, môže byť kóder pomalý. Opačný prípad nastáva, keď sa súbory neustále menia a zálohujú a je malá šanca, že sa bude záloha dekomprimovať, môže byť dekompresor pomalý. V skutočnosti mnoho moderných kompresných metód je asymetrických.

Problémami uchovávania a prenosu informácii sa zaoberá teória informácií založená C. E. Shannonom v roku 1948. Dôležitým pojmom teórie informácií je entropia. Podľa [5] sa jedná o veličinu, ktorá vyjadruje mieru neurčitosti. Entropia reťazca určuje, koľko neurčitostí, náhodnosti´v reťazci je.

Najstaršími kompresnými metódami sú Braillovo písmo a Morseov kód. Hlavný rozvoj kompresie sa udial v posledných pätnástich rokoch.



Obrázok 2.1: Abeceda Braillovho písma

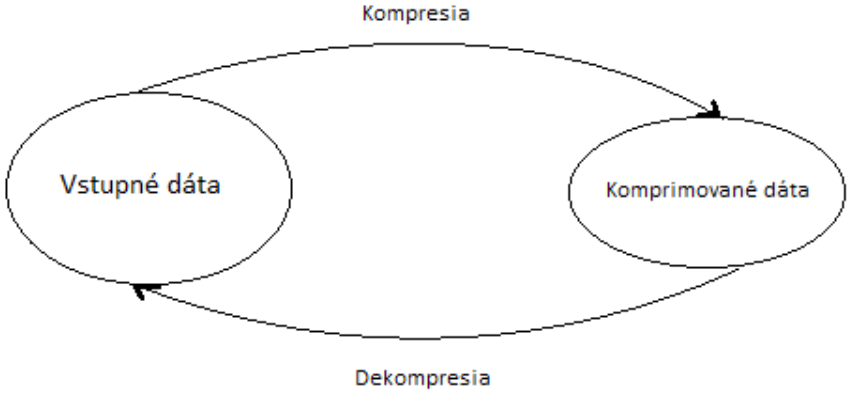
2.1 Univerzálna kompresia dát

Je taká kompresia dát, keď kompresor ani dekompresor, nevyužíva štatistiky o vstupnom toku, teda nie je možné predvídať, aký typ dát sa spracováva. Ide o bezstratovú kompresiu, teda komprimuje dáta tak, že sa dajú spätne dekomprimovať do totožnej podoby.

Existujú dva základné prístupy a postupy univerzálnej kompresie dát:

- štatistické metódy
- slovníkové metódy

Štatistické metódy sa opierajú o frekvencie výskytu jednotlivých symbolov. Využívajú kódy s premennou dĺžkou. Hlavnými dvoma zástupcami tohto typu kódovania sú Huffmanovo a aritmetické kódovanie. Táto práca je venovaná hlavne slovníkovým metódam, ktorých všeobecný popis je uvedený v ďalšej kapitole.



Obrázok 2.2: Znázornenie bezstratovej kompresie a dekompresie

A .-	N -.	0 -----
B -...	O ---	1 .----
C -.-.	P .--.	2 ..---
D -..	Q --.-	3 ...--
E .	R .-.	4 ....-
F ..-.	S ...	5 .....
G --.	T -	6 -....
H	U ..-	7 --...
I ..	V ...-	8 ---..
J .---	W .--	9 ----.
K -.-	X -..-	
L .-..	Y -.--	
M --	Z --..	

Obrázok 2.3: Morseov kód využívajúci kód s premenlivou dĺžkou.

2.2 Kompresná výkonnosť

Existuje niekoľko prístupov, ako vyjadriť úroveň kompresie. Kompresný pomer určíme ako podiel veľkostí výstupného a vstupného toku:

$$kompresný\ pomer = \frac{\text{veľkosť výstupného toku}}{\text{veľkosť vstupného toku}}$$

Ak je kompresný pomer väčší ako 1, jedná sa o expanziu.

Prevrátenou hodnotou kompresného pomeru je kompresný faktor. Táto hodnota je výstižnejšia, pretože s rastúcim kompresným faktorom rastie aj miera kompresie.

$$kompresný\ faktor = \frac{\text{veľkosť vstupného toku}}{\text{veľkosť výstupného toku}}$$

Pre porovnanie rôznych kompresných metód slúži veličina nazývaná kompresný zisk.

$$kompresný\ zisk = 100\ \ln\frac{\text{referenčná veľkosť}}{\text{komprimovaná veľkosť}}\ ,$$

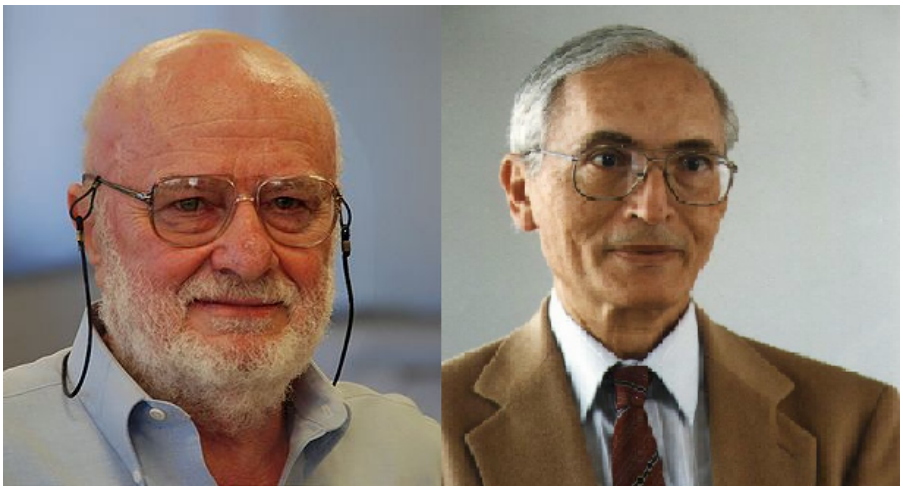
kde referenčná veľkosť je veľkosť vstupného toku, prípadne veľkosť komprimovaného toku vytvoreného inou metódou.

Rýchlosť kompresie sa môže merať v počte cyklov na byte (cpb). Jedná sa o strednú hodnotu počtu strojových cyklov na kompresiu jedného bytu. A vypočíta sa nasledovne.

$$cpb = \frac{\text{celkový počet strojových cyklov}}{\text{veľkosť vstupného toku}}$$

3 Slovníkové metódy

Pri zrode týchto metód stáli Abraham Lempel a Jacob Ziv, ktorí vyvinuli prvé slovníkové metódy. V V roku 1977 publikovali svoju prácu *A Universal Algorithm for Sequential Data Compression* [3] a spustili tým lavínu ďalšieho vývoja týchto algoritmov a vymýšľania podobných trikov v iných programoch.



Obrázok 3.1: Autori slovníkových metód (Lempel na fotografii vľavo)

Slovníkové metódy nepoužívajú modely, ani kódy s premennou dĺžkou. A preto, že kvalita kompresie nezáleží od kvality statického modelu, ktorý zavisí od typu dát, sú slovníkové metódy omnoho univerzálnejšie. Využívajú statický alebo adaptívny slovník. V ňom sa nachádzajú reťazce znakov a k nim priradené tokeny.

Reťazec n symbolov, sa môže slovníkovým procesorom komprimovať na nH bitov, kde H je entropia reťazca. Za predpokladu, že vstupný súbor je veľký, sú slovníkové metódy entropické.

Statický slovník je stály po celú dobu kompresie či dekompresie, adaptívny sa mení podľa vstupného toku dát. Využitie statického slovníka nie je pre univerzálnu kompresiu vhodné, preto sa preferujú adaptívne slovníkové metódy.

Radia sa medzi asymetrické kompresné metódy, pretože kompresor je o mnoho zložitejší a pracuje výrazne dlhšie ako dekompresor. Tieto metódy dovoľujú pridávať aj rušiť nové reťazce v slovníku počas kompresie(dekompresie). Vo všeobecnosti pracuje kóder adaptívnej slovníkovej metódy tak, že v cykle vykonáva tieto činnosti:

1. Čítanie vstupného toku.
2. Rozklad na úseky, slová (frázy).
3. Hľadanie slova v slovníku.
4. Ak sa fráza v slovníku našla: zápis značky na výstup, ak sa nenašla: pridanie frázy do slovníka a vyslanie nekomprimovaného slova na výstup.
5. Mazanie starého obsahu slovníka.

3.1 Jednoduchá slovníková kompresia

Pre uvedenie do tematiky si uvedieme v tejto podkapitole, základné fakty o tejto jednoduchej metóde.

Jedná sa o jednoduchú dvoj-priechodovú metódu založenú I. Mohamedom. V prvom priechode sa pri čítaní vstupného textu pripraví zoznam všetkých rôznych bytov, ktoré zdrojový súbor obsahuje. Ak byte má 8 bitov, zoznam má maximálnu dĺžku 256 položiek. V druhom priechode sa použije tento zoznam, ku kompresii. Teraz si podrobnejšie vysvetlíme kroky tejto metódy.

1. Číta sa vstupný súbor, pripravuje sa zoznam všetkých rôznych nájdených bytov. Zaznamená sa taktiež počet výskytov týchto bytov.
2. Zoznam bytov sa zostupne usporiada podľa počtu výskytov. Teda na začiatku sú často používané byty a na konci tie občasné.
3. Tento usporiadaný zoznam je de facto slovník. Do komprimovaného súboru sa zapíše dĺžka slovníka i slovník samotný.
4. Zdrojový súbor sa znova prechádza. Každý načítaný byte sa hľadá v slovníku a index sa zaznamená. Index môže nadobúdať hodnoty od 0 po 255, ale keďže naajćastejśie vyskytujúce sa byty sú na začiatku slovníka, väčšina indexov bude malá. Preto sa najskôr zapíše v troch bitov dĺžka indexu a až potom index samotný.

Dekompresia je pomerne jednoduchá. Prečíta sa dĺžka slovníka, slovník samotný a následne sa začnú čítať dĺžky indexov a indexy samotné. A pomocou indexu sa nájde každý byte v slovníku a pošle sa na výstup dekompresie.

Kompresia je dosiahnutá tým, že najčastejšie vyskytujúce sa byty sú na začiatku slovníka. Každý byte je tak nahradený 4 až 11 bitovou hodnotou. Najhorší prípad by nastal, keby sa v súbore nachádzalo všetkých 256 bytov a boli rovnako časté.

Nevýhodou tejto metódy je pomalá kompresia. Tento fakt spôsobujú dva prechody vstupným súborom a pomalé vyhľadávanie v slovníku keďže nie je usporiadaný podľa hodnôt bytov.

3.2 LZ77

Táto podkapitola detailne popisuje metódu LZ77.

3.2.1 Popis

Algoritmus bol publikovaný v roku 1977 v diele [3]. Prvý algoritmus na slovníkovej báze, tiež nazývaný „Sliding window“, teda pohyblivé okno. Metódy postavené na myšlienkach LZ77 sa vyvíjajú dodnes. Základnou myšlienkou tohto algoritmu je použitie načítaného toku dát ako slovníku.

Slovník má formu pohyblivého okna, ktoré sa presúva zľava doprava od začiatku kódovaného textu. Toto pohyblivé okno má dve časti: vyhľadávací buffer a predvídací buffer. Od veľkostí týchto bufferov závisia kompresné vlastnosti tohto algoritmu, ktoré budú diskutované v ďalšej kapitole.

3.2.2 Princíp

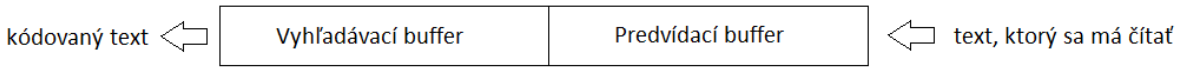
3.2.2.1 Kóder

Najskôr musí zapísať dĺžky bufferov, aby bola možná správna dekompresia. Kompresia prebieha nasledovne:

1. Prechádza sa vyhľadávací buffer sprava doľava a hľadá zhodu s prvým symbolom predvídacieho bufferu.
2. Ak nájde týchto zhôd viac, vyberie tú, v ktorej sa zhoduje čo najviac ďalších symbolov za práve porovnávanými symbolmi v bufferoch.
3. Pripraví sa značka (tag) vo formáte:
   - offset nájdenej zhody vo vyhľadávacom bufferi,
   - dĺžka zhody
   - nasledujúci znak v predvídacom bufferi (nezakódovaný znak).
4. Značka sa zapíše do výstupného toku a okno sa posunie o dĺžku zhodujúceho sa reťazca zvýšenú o 1, kvôli symbolu, ktorý je priamo v značke.
5. Ak sa žiadna zhoda nenašla, do výstupného toku sa zapíše značka s offsetom aj dĺžkou 0 a symbolom, pre ktorý sa nenašla zhoda a okno sa posunie o 1 pozíciu.

Z tohoto princípu vyplýva, že v ľavej časti okna (vo vyhľadávacom bufferi) sa nachádzajú už zakódované dáta, zatiaľ čo v pravej častí (predvídací buffer) sú dáta ešte nezakódované.

Veľkosti prvej a druhej časti značky závisia od veľkostí bufferov. Keďže offset nemôže byť väčší ako veľkosť predvídacieho bufferu, bude na jeho zápis potrebných $\log_2(s)$ bitov, kde **s** je veľkosť vyhľadávacieho bufferu. Pre dĺžku zhody platí, že nemôže byť väčší ako veľkosť predvídacieho bufferu zmenšená o jedna (kvôli znaku ktorý sa pridáva priamo do značky), teda pre zapísanie druhého poľa značky bude potrebných $\log_2(l-1)$ bitov, kde **l** je veľkosť predvídacieho bufferu. Z toho plynie poznatok, že zväčšenie veľkosti vyhľadávacieho bufferu má síce za následok častejšie nachádzanie zhody, ale značka zaberá viac miesta a kompresia sa spomalí. Veľkosť predvídacieho bufferu zasa ovplyvňuje dĺžku maximálnej zhody. V praxi sa používa vyhľadávací buffer s veľkosťou tisícov a predvídací buffer s veľkosťou stoviek bajtov.



Obrázok 3.2: Posuvné okno (Sliding window)

3.2.2.2 Dekóder

Dekodér najskôr načíta zo vstupu veľkosti bufferov, až potom prebieha samotná dekompresia. Princíp je jednoduchší ako u kódera. Používa sa buffer veľkosti okna kódera, čítajú sa vytvorené značky a pomocou nich sa pridávajú na koniec buffera zakódované reťazce. Postup je nasledovný:

1. Načíta sa značka.
2. Ak je offset i dĺžka nulová, tretia časť značky (znak) sa pridá do bufferu a pošle na výstup.
3. V opačnom prípade sa **j** znakov od offsetu **i** pridá na koniec bufferu a pošle na výstup, kde **i** je offset a **j** je dĺžka zhody. Taktiež sa do bufferu a na výstup pošle tretia časť značky (nezakódovaný znak).

3.2.3 Príklad

V tejto podkapitole si predvedieme kompresiu a dekompresiu skúšobného textu metódou LZ77. Skúšobný text: „this□is□test□aaaa□test“

3.2.3.1 Kompresia

Krok	Zakódovaný text	Zapísaná značka	Vyhľadávací buffer	Predvídací buffer
0.				this□is□test□aaaa□test
1.	„t“	(0,0,“t“)	t	his□is□test□aaaa□test
2.	„h“	(0,0,“h“)	th	is□is□test□aaaa□test
3.	„i“	(0,0,“i“)	thi	s□is□test□aaaa□test
4.	„s“	(0,0,“s“)	this	□is□test□aaaa□test
5.	„□“	(0,0,“□“)	this□	is□test□aaaa□test
6.	„is□t“	(3,3,“t“)	this□is□	test□aaaa□test
7.	„e“	(0,0,“e“)	this□is□te□	st□aaaa□test
8.	„st“	(4,1,“t“)	this□is□test	□aaaa□test
9.	„□a“	(5,1,“a“)[1]	this□is□test□a	aaa□test
10.	„aaa□“	(1,3,“□“)	this□is□test□aaaa□	test
11.	„test“	(10,3,“t“)[2]	this□is□test□aaaa□test	

Tabuľka 3.1

1 Dĺžka je väčšia ako offset, pretože prvé tri písmená „a“ z predvídacieho bufferu sú zhodné s posledným „a“ z vyhľadávacieho bufferu a prvými dvoma „a“ z predvídacieho bufferu.

2 Zhodné je aj posledné „t“, ale to sa pridá do tretieho poľa značky.

3.2.3.2 Dekompresia

Krok	Prečítaná značka	Dekódovaný text	Buffer
0.			
1.	(0,0,“t“)	„t“	t
2.	(0,0,“h“)	„h“	th
3.	(0,0,“i“)	„i“	thi
4.	(0,0,“s“)	„s“	this
5.	(0,0,“□“)	„□“	this□
6.	(3,3,“t“)	„is□t“	this□is□
7.	(0,0,“e“)	„e“	this□is□te□
8.	(4,1,“t“)	„st“	this□is□test
9.	(5,1,“a“)	„□a“	this□is□test□a
10.	(1,3,“□“)	„aaa□“	this□is□test□aaaa□
11.	(10,3,“t“)	„test“	this□is□test□aaaa□test

Tabuľka 3.2

3.2.4 Súhrn a vylepšenia

Keďže kompresia je jednoduchšia a tým pádom aj rýchlejšia ako dekompresia, metóda je užitočná v prípadoch keď potrebujeme len občasnú kompresiu, ale častú dekompresiu. Aj keď to nemusí byť na prvý pohľad zrejmé, metóda implicitne predpokladá, že rovnaké vzorky dát sa vyskytujú blízko seba, keďže práve kódujúce sa dáta nemôžu zakódovať pomocou dát ktoré už nie sú vo vyhľadávacom bufferi, teda už odtiekli. Pre takéto súbory má metóda dobré kompresné výsledky.

Od vynájdenia tejto metódy bolo navrhnutých a zrealizovaných mnoho vylepšení a modifikácii. Napríklad zrušenie tretieho poľa značky, realizácia okna kruhovou frontou, realizácia vyhľadávacieho bufferu binárnym vyhľadávacím stromom. či zavedenie premennej dĺžky pre značky „offset“ a „dĺžka“.

3.2.5 LZSS

Táto metóda je v podstate vylepšená LZ77. Vytvorili ju James Storer a Thomas Szymanski. Oproti LZ77 má tri výrazné zmeny. Predvídací buffer je implementovaný ako kruhová fronta, vyhľadávací buffer je implementovaný ako binárny vyhľadávací strom, značky majú polia offset a dĺžka nie však tretie pole pre nasledujúci symbol.

Pomocou kruhovej fronty sa výrazne zníži fyzický presun dát v pamäti. Namiesto dát sa pohybujú ukazatele začiatku a konca fronty.

Binárny vyhľadávací strom zasa podstatne urýchľuje vyhľadávanie v slovníku. Postup tvorby je nasledovný: z dát ktoré patria do vyhľadávacieho bufferu sa postupne od offsetu veľkosti dĺžky predvídacieho bufferu po offset veľkosti vyhľadávacieho bufferu zvýšenej o jedna vytvoria reťazce o veľkosti dĺžky predvídacieho bufferu. Reťazce a ich offsetty sa vložia do binárneho vyhľadávacieho stromu. Keď dôjde k zmene slovníka, strom sa aktualizuje.

Chýbajúce tretie pole značiek je nahradené tým že do výstupu kódera sa musia pridávať špeciálne bity (flagy), ktoré určujú či sa jedná o značku alebo nezakódovaný znak.

3.3 LZ78

Táto podkapitola detailne popisuje metódu LZ78.

3.3.1 Popis

Algoritmus bol publikovaný v roku 1978 v diele [4]. Bol to výsledok ďalšieho vývoja algoritmu LZ77. Základnou myšlienkou tejto metódy je použitie načítaného toku dát ako slovníku, práve tak ako u LZ77, avšak nepoužíva pohybujúce sa okno, či predvídací alebo vyhľadávací buffer. Používa slovník nájdených reťazcov, ktorý je na začiatku prázdny a ktorého veľkosť je všeobecne obmedzená len veľkosťou fyzickej pamäte. Vhodnou štruktúrou pre slovník je strom, ktorého každý uzol sa môže vetviť až do počtu všetkých symbolov. Teda pre 8-bitové symboly môže mať každý uzol maximálne 256 potomkov.

3.3.2 Princíp

3.3.2.1 Kóder

Značky, ktoré kóder vysiela na výstup majú dve položky:

1. odkaz do slovníka
2. symbol

Najskôr sa ale zapíše na výstup maximálna veľkosť slovníka, aby dekodér mohol správne prebudovávať svoj slovník a taktiež aby vedel akú veľkosť bude mať odkaz na položku slovníka. Veľkosť odkazu do slovníka vypočítame ako $\log_2(M)$ bitov, kde M je maximálna veľkosť slovníka.

Nasleduje samotná kompresia, ktorá prebieha v cykle, kým sa nezakódujú všetky znaky nasledovne:

1. Načíta sa symbol zo vstupu.
2. Symbol sa hľadá v slovníku medzi potomkami nultého uzla.
3. Ak sa nájde, načíta sa ďalší znak
4. Ak sa nenájde, pridá sa na prvú voľnú pozíciu v slovníku ako potomok uzla s predchádzajúcou zhodou, resp. ako potomok nultého uzla a na vstup sa vyšle značka vo formáte: (odkaz na uzol s predchádzajúcou zhodou, resp. odkaz na nultý uzol, znak ktorý sa v slovníku nenašiel).
5. Ak sa slovník zaplní, vykonám jednu z ďalej diskutovaných možností.

Slovník má tendenciu rýchlo rásť, čo je nevýhodou tejto metódy. Ak je slovník plný, je niekoľko možností ako postupovať:

1. Slovník sa zmrazí, stane sa statickým a už sa nepridávajú žiadne nové uzly.
2. Odstránia sa niektoré najmenej používané položky slovníka.
3. Zmaže sa celý slovník a bude sa budovať odznova. Vstup sa tak rozdelí na úseky s vlastným slovníkom.

3.3.2.2 Dekóder

Najskôr si dekóder prečíta maximálnu veľkosť slovníka, následne sa čítajú postupne všetky značky a buduje sa slovník a to týmto spôsobom:

1. Prečíta sa značka.
2. Nájde sa uzol, na ktorý ukazuje odkaz v značke.
3. Tomuto uzlu sa pridá potomok so symbolom, ktorý sa nachádza v značke.
4. Na výstup sa vyšlú všetky symboly, ktoré sa nachádzajú na ceste od nultého uzla k novo vytvorenému, vrátane novo vytvoreného.
5. Ak sa slovník naplnil, konáme obdobne ako u kódera.

3.3.3 Príklad

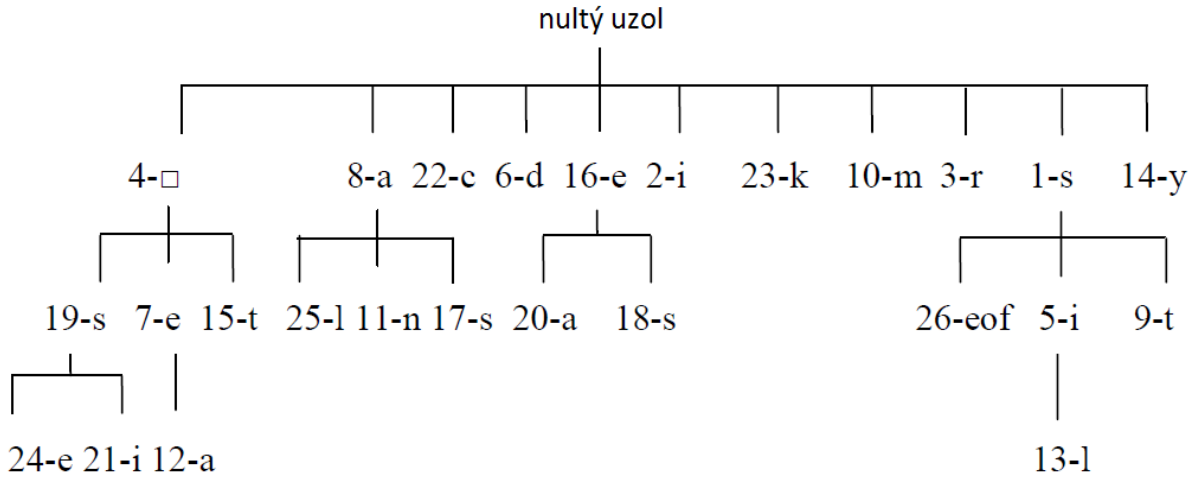
Príklad je prevzatý z: Studijní opora předmětu KKO [2].

Na skúšobnom texte si predvedieme prvých 14 krokov kompresie metódou LZ78. Skúšobným textom je reťazec „sir□sid□eastman□easily□teases□sea□sick□seals“.

Krok	Zakódovaný reťazec	Zapísaná značka
1.	„s“	(0,“s“)
2.	„i“	(0,“i“)
3.	„r“	(0,“r“)
4.	„□“	(0,“□“)
5.	„si“	(1,“i“)
6.	„d“	(0,“d“)
7.	„□e“	(4,“e“)
8.	„a“	(0,“a“)
9.	„st“	(1,“t“)
10.	„m“	(0,“m“)
11.	„an“	(8,“n“)
12.	„□ea“	(7,“a“)
13.	„sil“	(5,“l“)
14.	„y“	(0,“y“)

Tabuľka 3.3

Na nasledujúcom obrázku je znázornený slovník (slovníkový strom) v stave na konci kompresie skúšobného textu.



Obrázok 3.3: Slovníkový strom

3.3.4 Súhrn a vylepšenia

Odpadá nevýhoda metódy LZ77, ktorá má dobré kompresné výsledky za predpokladu, že rovnaké zhluky dát sa vyskytujú blízko seba. Avšak na druhej strane slovník má tendenciu veľmi rýchlo rásť, čo má za následok zmrazenie slovníka, vymazávanie starých položiek, resp. zmazanie celého slovníka. Ak sa vyberie tretia možnosť, teda zmazanie celého slovníka, budú dobré kompresné výsledky tak ako aj u LZ77 za predpokladu, že rovnaké zhluky dát sú blízko seba.

Metóda je symetrická, keďže kóder aj dekóder pracujú obdobne a trvanie kompresie a dekompresie je teda podobné. Z toho vyplýva, že je vhodná na použitie, keď sa dáta rovnako často komprimujú ako dekomprimujú.

Mojím vlastným vylepšením je pomocné pole odkazov na všetky uzly stromu pri dekompresii, Nemusí sa tak prechádzať celý strom, aby sa mohla načítaná značka nájsť a rozbaliť do pôvodného reťazca.

4 Implementácia

Táto kapitola popisuje, akým spôsobom boli implementované metódy LZ77 a LZ78, ktoré som si k implementácii zvolil.

Oba programy sú implementované v jazyku C++ ako knižnice a konzolové aplikácie, ktoré sú postavené na týchto knižniciach. Bol použitý objektový a modulárny prístup.

Keďže obe metódy vysielajú pri kompresii na výstup značky, ktoré obsahujú časti, ktorých dĺžka je určená presným počtom bitov, museli byť implementované knižnice BitWriter a BitReader. Objekty tried týchto knižníc slúžia ako medzistupeň medzi výstupom (vstupom) komprimačného (dekomprimačného) algoritmu a skutočným výstupom (vstupom).

Aplikácie, ktoré sú postavené na komprimačných knižniciach, sa starajú o prevzatie potrebných dát od užívateľa, ako je cesta k vstupnému a výstupnému súboru a iné nastavenia kompresie. Pre testovacie účely tieto programy merajú čas trvania kompresie i dekompresie.

4.1 LZ77

Moduly knižnice:

- BitReader – bitové vstupné operácie
- BitWriter – bitové výstupné operácie
- Buffer – základná trieda bufferov
- LaBuffer – trieda predvídacieho buffera
- LZ77 – základná trieda knižnice
- Sbuffer – trieda vyhľadávaceho buffera

Rozhranie knižnice LZ77 je veľmi jednoduché. Pre spustenie kompresie či dekompresie sú potrebné len 4 metódy. Metóda *setInput* slúži na nastavenie vstupného toku dát, kým metóda *setOutput* nastavuje výstupný tok. Metóda *compress* zaháji kompresiu, pri čom je potrebné zadať parametrami veľkosti vyhľadávaceho a predvídacieho buffera. Metóda *decompress* zaháji dekompresiu.

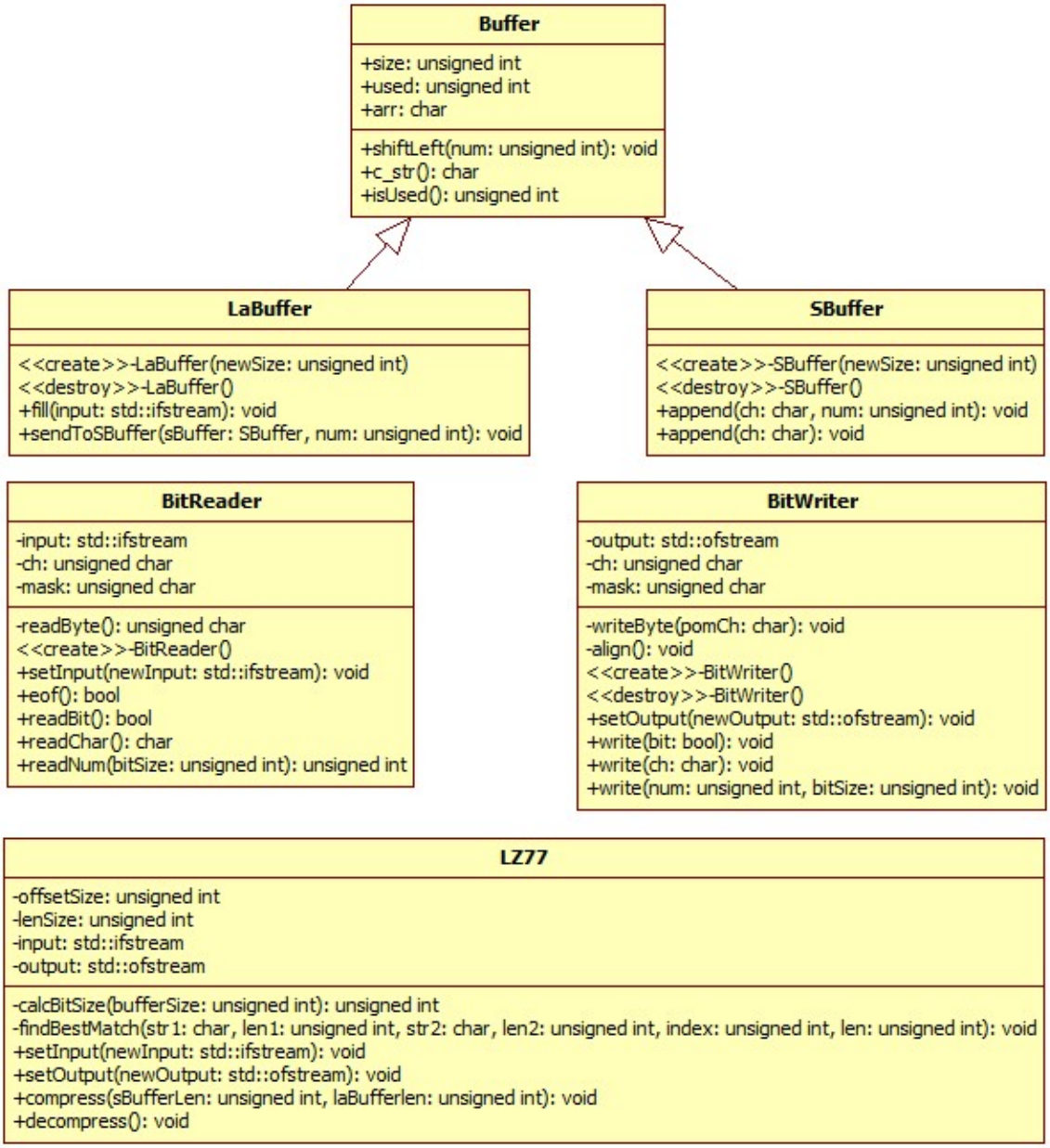
4.1.1 Príklad použitia knižnice LZ77

Nasledujúci text je kód v jazyku C++.

```
// vytvoríme objekt hlavnej triedy
LZ77 lz77;
// nastavíme vstupný tok
lz77.setInput(&myInputStream);
// nastavíme výstupný tok
lz77.setOutput(&myOutputStream));

// spustí kompresiu s vyhľadávacím bufferom veľkosti 5000 B
// a predvídacím bufferom veľkosti 100 B
lz77.compress(5000,100);
// spustí dekompresiu
lz77.compress();
```

4.1.2 Diagram tried



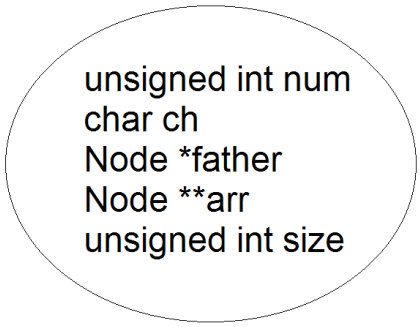
Základná trieda LZ77 využíva ostatné triedy a vytvára objekty týchto tried.

4.2 LZ78

Moduly knižnice:

- BitReader – bitové vstupné operácie
- BitWriter – bitové výstupné operácie
- LZ78 – základná trieda knižnice
- Node – trieda pre uzol stromu

Slovník je implementovaný ako obojsmerne viazaný dynamicky alokovaný strom. Spätné previazanie na predkov zrýchľuje rozkódovanie reťazcov. Trieda *Node* definuje každý uzol tohto stromu. Nasledujúci obrázok znázorňuje objekt triedy *Node*.



Obrázok 4.1: Položky uzla triedy Node

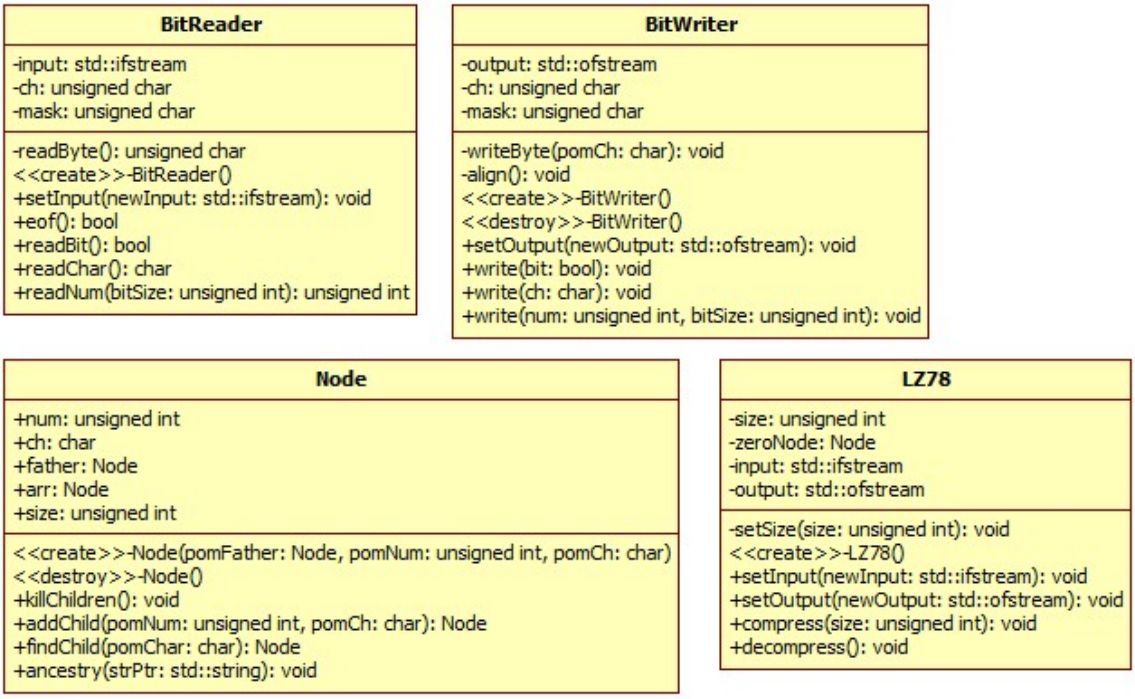
Položka *num* reprezentuje číslo uzla. Položka *father* obsahuje ukazateľ na otcovský uzol. *arr* je dynamicky alokované pole ukazateľov na dynamicky alokovaných potomkov. V položke *size* je uložený počet potomkov.

Okrem tejto stromovej štruktúry slovníka, je použité pomocné pole odkazov na každý uzol stromu. To veľmi zrýchli vyhľadanie uzla podľa jeho čísla, keďže sa nemusí prechádzať cely strom.

Z viacerých možností, ktoré sa môžu uplatniť pri zaplnení slovníka som si zvolil odstránenie všetkých uzlov stromu a následne jeho znova-vybudovanie. Vstupný tok je tak rozsekaný na bloky s vlastným slovníkom.

Rozhranie knižnice LZ78 je rovnako jednoduché. Pre spustenie kompresie či dekompresie sú potrebné rovnaké 4 metódy. Metóda *setInput* slúži na nastavenie vstupného toku dát. Metóda *setOutput* slúži na nastavenie výstupného toku. Metóda *compress* zaháji kompresiu, pri čom je potrebné zadať parametrom maximálnu veľkosť slovníka. Metóda *decompress* zaháji dekompresiu.

4.2.1 Diagram tried



Základná trieda LZ78 využíva ostatné triedy a vytvára objekty týchto tried.

4.2.2 Príklad použitia knižnice LZ78

Nasledujúci text je kód v jazyku C++.

```
// vytvoríme objekt hlavnej triedy
LZ78 lz78;

// nastavíme vstupný tok
lz78.setInput(&myInputStream);

// nastavíme výstupný tok
lz78.setOutput(&myOutputStream));

// spustí kompresiu s veľkosťou slovníka 50 kB
lz77.compress(50000);

// spustí dekompresiu
lz77.compress();
```

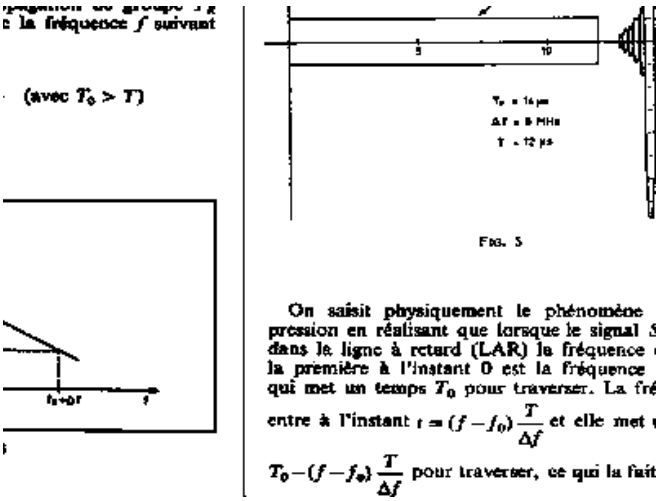
5 Testovanie a výsledky

Táto kapitola je venovaná testovaniu a porovnávaniu kompresných výsledkov metód LZ77, LZ78 a súčasne používaných slovníkových algoritmov.

Programy boli testované na korpuse Calgary. Korpus Calgary je kolekcia testových a binárnych súborov, ktoré slúžia na testovanie bezstratovej kompresie. Korpus vyvinuli Tim Bell a John Clesary v roku 1987 a doteraz slúži ako užitočná pomôcka. Obsahuje 14 súborov, uvedených v nasledujúcej tabuľke.

Názov súboru	Typ dát	Veľkosť (Byte)
bib	Bibliografia	111261
book1	Vymyslená kniha	768771
book2	Skutočná kniha	610856
geo	Geofyzikálne dáta	102400
news	Dávkový súbor USENET	377109
obj1	Objekový kód pre VAX	21504
obj2	Objektový kód pre Apple Mac	246814
paper1	Technický dokument	53161
paper2	Technický dokument	82199
pic	Čierno-biely faxový obrázok	513216
progc	Zdrojový kód v jazyku C	39611
progl	Zdrojový kód v jazyku LISP	71646
progp	Zdrojový kód v jazyku PASCAL	49379
trans	Prepis terminálovej relácie	93695

Tabuľka 5.1



Obrázok 5.1: pic (korpus Calgary)

5.1 LZ77

Táto podkapitola je zameraná na testovanie algoritmu LZ77, hlavne na závislosti kompresných vlastností od veľkosti bufferov a typov komprimovaných súborov.

5.1.1 Test kompresie rôznych typov súborov

Porovnanie kompresie na rôznych typoch súborov pri konfigurácii: predvídací buffer 200 znakov, vyhľadávací buffer 10 000 znakov

Súbor	Pôvodná veľkosť (byte)	Veľkosť po kompresii (byte)	Kompresný pomer
pic	513216	82531	0,16
trans	93695	34475	0,37
progl	71646	26690	0,37
progp	49379	18560	0,38
obj2	246814	124407	0,5
bib	111261	59761	0,54
progc	39611	22261	0,56
book2	610856	355415	0,58
paper1	53161	30950	0,58
paper2	82199	50708	0,62
news	377109	239919	0,64
book1	768771	533866	0,69
obj1	21504	16336	0,76
geo	102400	96005	0,94

Tabuľka 5.2

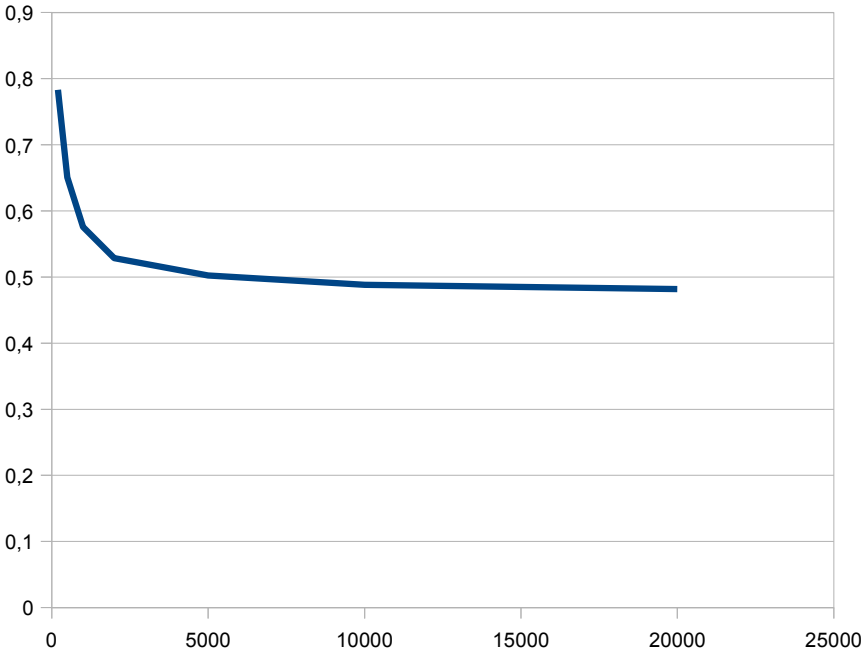
Z tabuľky vyplýva, že zatiaľ čo čierno-bielo obrázok bol komprimovaný len na 16 percent svojej pôvodnej veľkosti, geofyzikálne dáta neboli komprimované takmer vôbec. Kompresný pomer je teda silne závislý na type dát.

5.1.2 Testy závislostí od dĺžky bufferov

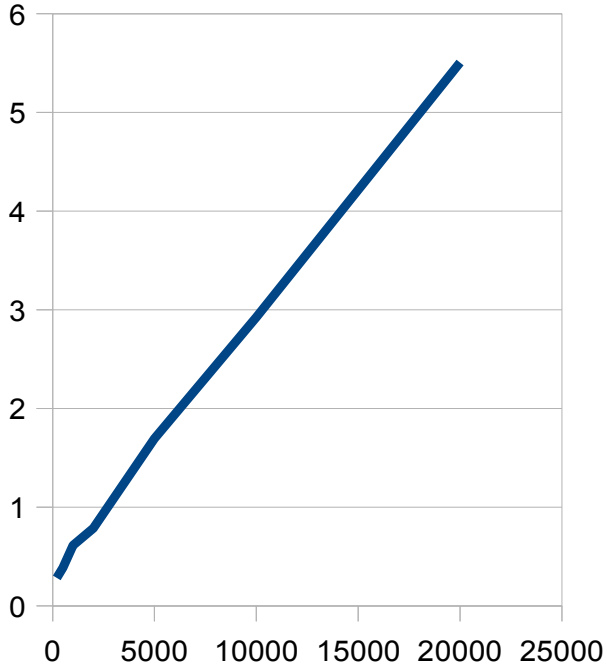
Predvídací buffer bude mať pevnú dĺžku 100 znakov, zatiaľ čo vyhľadávací buffer sa bude postupne zvyšovať. Test bude realizovaný na objektovom kóde pre Apple Mac (obj2).

Z prvého grafu (obrázok 5.2) vyplýva, že so zväčšujúcim sa vyhľadávacím bufferom sa zvyšuje aj úroveň kompresie (klesá kompresný pomer). Avšak výrazne zvyšovanie kompresie je približne do veľkosti buffera 5000 znakov.

Z druhého grafu (obrázok 5.3) vyplýva, že dĺžka trvania kompresie neustále úmerne rastie so zvyšujúcim sa bufferom. Z týchto dvoch poznaní môžeme usúdiť, že vhodnou veľkosťou vyhľadávaceho bufferu je niekoľko tisíc znakov. Ďalšie zväčšovanie buffera, zbytočne predlžuje kompresiu, bez viditeľného zlepšenia kompresie.

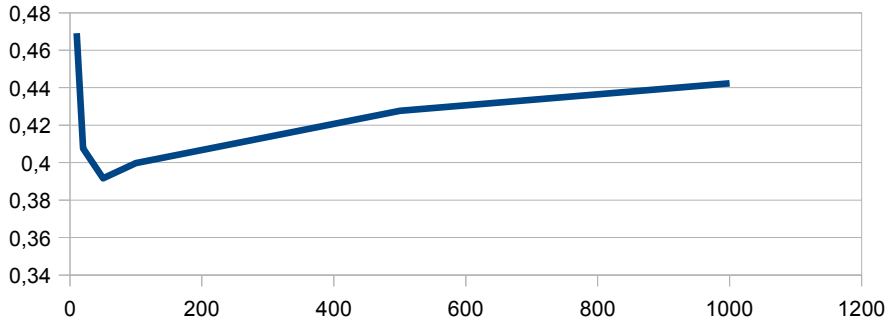


Obrázok 5.2: Závislosť kompresného pomeru od veľkosti vyhľadávacieho buffera (byte)

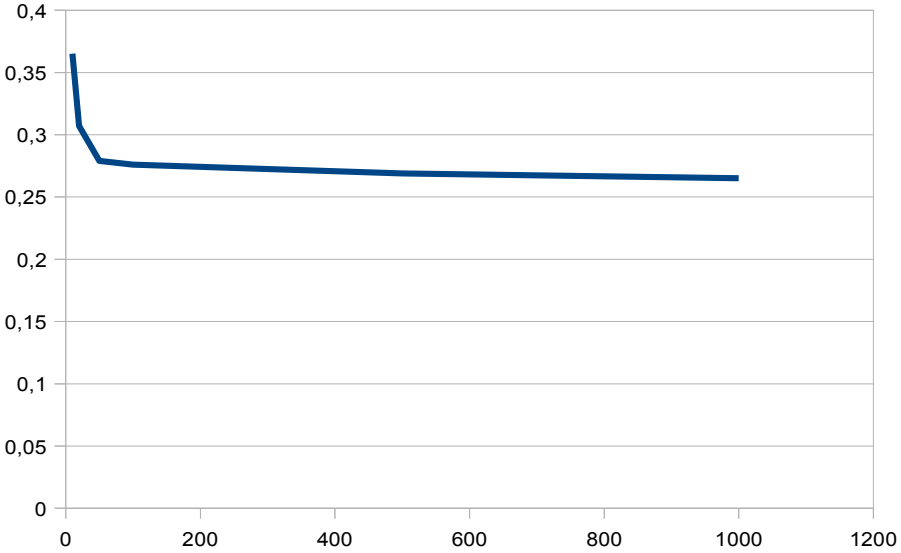


Obrázok 5.3: Závislosť trvania kompresie (s) od veľkosti vyhľadávacieho buffera (byte)

V ďalšom teste bude mať vyhľadávací buffer pevnú dĺžku 3000 znakov a veľkosť predvídacieho buffera sa bude meniť a bude sa sledovať úroveň kompresie a čas trvania kompresie. Test prebehne na zdrojovom kóde jazyku LISP (progl).



Obrázok 5.4: Závislosť kompresného pomeru od veľkosti predvídacieho buffera (byte)



Obrázok 5.5: Závislosť trvania kompresie (s) od veľkosti predvídacieho buffera (byte)

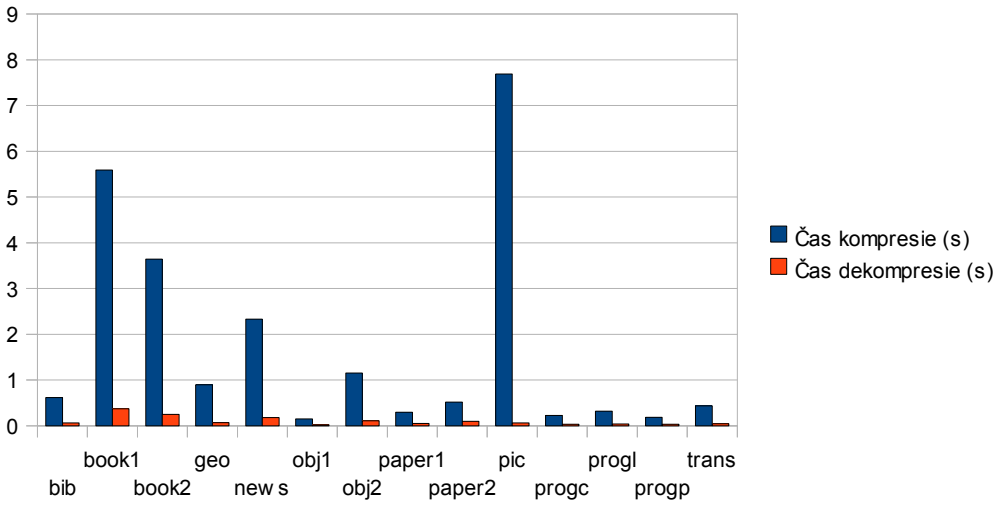
Prvý graf ukazuje, že keď je predvídací buffer veľký len niekoľko znakov, jeho zväčšovanie má za následok zlepšovanie kompresného pomeru. To však platí len do dĺžky niekoľkých desiatok znakov. Keď je buffer príliš veľký, nachádza sa len málo takýchto dlhých zhodných úsekov dát, a tým že sa musí zväčšovať veľkosť poľa *dĺžka zhody* v značke, kompresný pomer rastie (kompresia sa zhoršuje).

Z druhého grafu je jasné, že väčší predvídací buffer znamená rýchlejšiu kompresiu, pretože každá dlhá zhoda kompresiu urýchli. Avšak treba podotknúť, že významné zrýchľovanie sa deje približne do dĺžky 50 znakov.

Test ukázal, že ideálna dĺžka predvídacieho buffera je niekoľko desiatok znakov. Ďalšie zväčšovanie má za následok horší kompresný pomer.

5.1.3 Porovnanie trvania kompresie a dekompresie

Pre všetky druhy testovacích vstupných dát sa medzi sebou porovnajú dĺžky trvania kompresie a dekompresie. Veľkosť vyhľadávaceho buffera bude 3000 znakov a veľkosť predvídacieho buffera 50 znakov.



Z grafu je zrejmé, že kóder pracuje výrazne dlhšie ako dekóder, čo súhlasí s princípmi tejto metódy. Test potvrdil vedomosť, že metóda je asymetrická.

5.2 LZ78

5.2.1 Test kompresie rôznych typov súborov

Porovnanie kompresie na róznych typoch súborov při maximálnej veľkosti slovníka 30000 znakov.

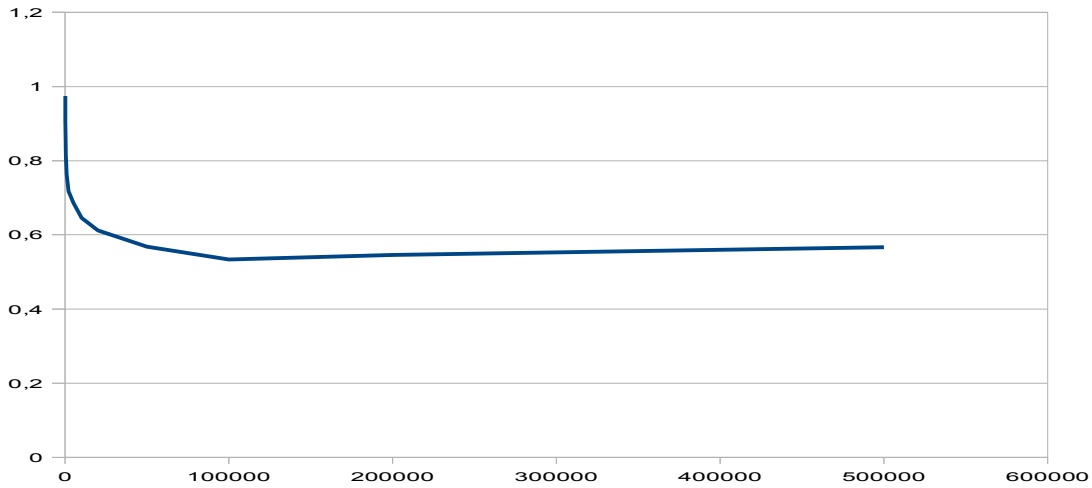
Súbor	Pôvodná veľkosť (byte)	Veľkosť po kompresii (byte)	Kompresný pomer
pic	513216	76612	0,15
progl	71646	39173	0,55
bib	111261	61699	0,55
trans	93695	52329	0,56
progp	49379	28214	0,57
book2	610856	351033	0,57
book1	768771	460907	0,6
paper2	82199	49848	0,61
obj2	246814	156580	0,63
news	377109	242439	0,64
paper1	53161	34985	0,66
progc	39611	27199	0,69
geo	102400	75697	0,74
obj1	21504	17556	0,82

Tabuľka 5.3

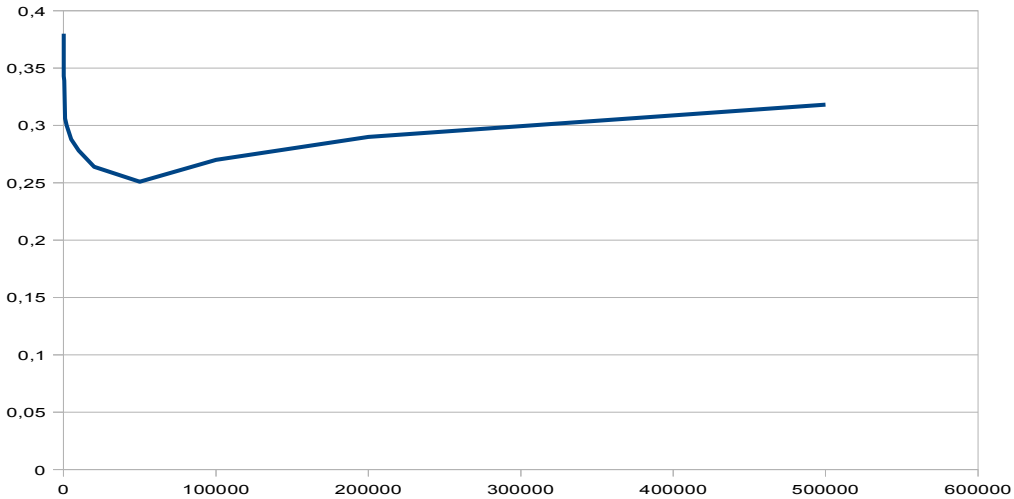
Aj metóda LZ78 vykazuje silnú závislosť kompresného pomeru od typu dát. Zle komprimujúce sa dáta obsahujú málo podobných zhlukov dát, alebo sú tieto hluky príliš rozptýlené.

5.2.2 Test závislostí od veľkosti slovníka

Tento test je zameraný na zistenie závislostí kompresného pomeru, a kompresného času od maximálnej veľkosti slovníka. Test bude vykonaný na súbore obsahujúcom skutočnú knihu (book2).



Obrázok 5.6: Závislosť kompresného pomeru od veľkosti slovníka (byte)



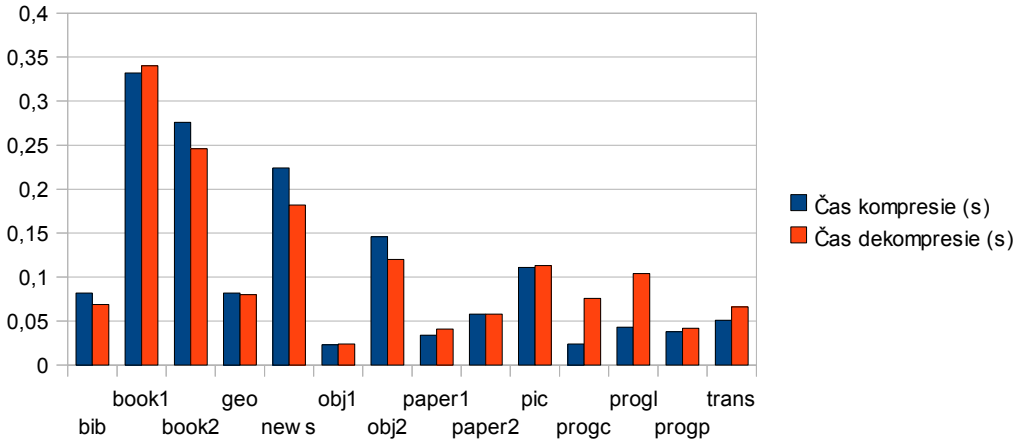
Obrázok 5.7: Závislosť trvania kompresie (s) od veľkosti slovníka (byte)

Prvý obrázok nám prezrádza, že pri veľmi malom slovníku je kompresia slabá, takmer žiadna. So zvyšujúcim sa slovníkom kompresný pomer rýchlo klesá až do veľkosti slovníka 100 kB. V tomto bude sa situácia mení a väčší slovník kompresiu zhoršuje. Dôvodom k tomuto je fakt, že súbor neobsahuje také dlhé zhody, aby sa tak veľký slovník využil, ale veľkosť značky rýchlo rastie.

Časová závislosť je podobná ako závislosť kompresného pomeru. Dôvodom k pomalej kompresii pri veľmi malom slovníku je fakt, že slovník sa neustále plní, maže a znova buduje, čo je časovo náročné. Veľmi veľký slovník kompresiu taktiež spomaľuje, pretože uzly majú veľa potomkov a hľadanie konkrétneho potomka ľubovoľného uzla je časovo náročné.

5.2.3 Porovnanie trvania kompresie a dekompresie

V tomto teste sa zameriame na porovnanie rýchlosti činnosti kódera a dekódera. Predpokladom je, že trvanie kompresie a dekompresie bude podobné, keďže aj činnosť kódera a dekóder je podobná. Test prebehne pri veľkosti slovníka 50 kB.



Z grafu je na prvý pohľad zrejmé, že trvanie kompresie a dekompresie je veľmi podobné, čím sa potvrdili predpoklady. Test potvrdil symetrickosť metódy.

5.3 Porovnanie pôvodných slovníkových metód so súčasnými

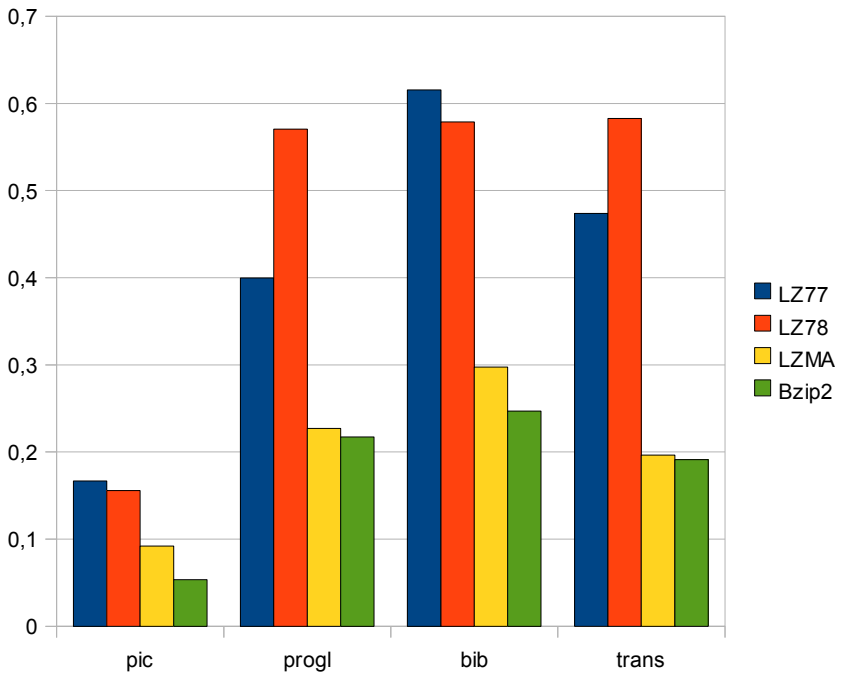
Aby sme si mohli urobiť obraz o pokroku a vývoji, ktorým slovníkové metódy prešli, porovnáme si na všetkých súboroch z korpusu Calgary metódy LZ77, LZ78, LZMA a Bzip2. LZ77 má vyhľadávací buffer veľkosti 3KB. LZ78 má veľkosť slovníka 50KB, Bzip2 má 900kB veľký slovník a LZMA má 3MB dlhý slovník.

Algoritmy LZMA a BZip2 boli vynájdené viac než 20 rokov po vynájdení LZ77 a sú stále vo vývoji, takže je viac než pravdepodobné, že ich kompresné výsledky budú výrazne lepšie.

	LZ77 (3kB,100B)	LZ78 (50kB)	LZMA (3MB)	Bzip2 (900kB)
bib	0,62	0,58	0,3	0,25
book1	0,74	0,3	0,36	0,3
book2	0,63	0,26	0,31	0,26
geo	0,94	0,56	0,52	0,56
news	0,67	0,31	0,33	0,31
obj1	0,73	0,5	0,44	0,5
obj2	0,52	0,31	0,27	0,31
paper1	0,62	0,31	0,34	0,31
paper2	0,66	0,3	0,35	0,3
pic	0,17	0,16	0,09	0,03
progc	0,6	0,32	0,33	0,32
progl	0,4	0,57	0,23	0,22
progp	0,42	0,22	0,23	0,22
trans	0,47	0,58	0,2	0,19
priemer	0,58	0,38	0,31	0,29

Tabuľka 5.4

Takmer u všetkých testovacích súboroch sa potvrdili predpoklady a súčasne používané metódy v nich dosahujú výrazne nižšie kompresné pomery ako metódy pôvodné. Roky vývoja a množstvo vylepšení, ako napríklad kódy s premennou dĺžkou, ktoré tieto metódy podstúpili ukazujú, aký obrovský je potenciál pôvodných metód Lempela a Ziva.



Obrázok 5.8: Zrovnanie kompresných pomerov algoritmov

6 Záver

Pri vypracovaní tejto práce som sa zoznámil s princípmi a základnými poznatkami z oboru kompresie dát. Konkrétne som sa zameral na slovníkové metódy, ktoré patria do kategórie bezstratovej kompresie. Z nich sa mi podarilo naštudovať a implementovať algoritmy LZ77 a LZ78. Po vykonaní testov som došiel k nasledovným záverom.

Kompresná výkonnosť testovaných slovníkových metód, silne závisí na type komprimovaných dát. Najlepšie kompresné výsledky sa dosiahli u čierno-bieleho obrázku a zdrojového kódu jazyka LISP. Naopak ako najhoršie komprimujúce sa ukázal objektový kód a geofyzikálne dáta.

Ďalšie zistenie sa týka trvania kompresia a dekompresie. Ukázalo sa že metóda LZ77 je asymetrická, jej kóder pracuje výrazne dlhšie ako dekóder, preto je dobre používateľná v prípade archivácie údajov, keď sa dáta oveľa častejšie dekomprimujú ako komprimujú. Testovaním metódy LZ78 sa zistilo, že jej kóder a dekóder pracujú podobne dlhý čas a teda metódu je vhodné používať v prípadoch, keď sa dáta približne rovnako často komprimujú a dekomprimujú.

Pre metódu LZ77 bolo testami na skúšobných dátach zistené, že najlepšie kompresné výsledky, vzhľadom na čas, dosahuje pri veľkosti vyhľadávacieho buffera niekoľko tisíc znakov a pri veľkosti predvídacieho bufferu niekoľko desiatok až stoviek znakov.

Metóda LZ78, dosahovala najlepšie kompresné výsledky pri maximálnej veľkosti slovníka niekoľko tisíc znakov.

Porovnaním pôvodných metód tvorcov Lempela a Ziva a súčasne používaných metód (LZMA, Bzip2) sa zistilo, že vývoj, ktorým prešli súčasné metódy, značne zvýšil ich výkonnosť.

Táto práca sa venuje hlavne metódam LZ77 a LZ78, avšak existuje veľké množstvo ďalších slovníkových algoritmov. Takže sa otvára možnosť nadviazať na túto prácu a implementovať ďalšie metódy a porovnať výsledky testov. Ďalšou možnosťou je implementácia komplexného testovacieho programu, ktorý by zjednodušil testovanie na rôznych typoch dát s rôznymi nastaveniami rôznych algoritmov.

Literatúra

[1] SALMON, D. *Data compression: the complete reference*. Springer. 2007. ISBN 9781846286025.

[2] DRÁBEK, V. Kódováni a komprese dat: Studijní opora. URL: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/KKO-IT/texts>.

[3] ZIV, Jacob – LEMPEL, Abraham. *A Universal Algorithm for Sequential Data Compression.* IEEE Transactions on Information Theory. 1977. 23(3). pp. 337–343.

[4] ZIV, Jacob – LEMPEL, Abraham. *Compression of Individual Sequences via Variable-Rate Coding*. IEEE Transactions on Information Theory. September. 1978. vol. 24. no. 5.

[5] SHANNON – C. E., WEAVER,Warren. *The Mathematical Theory of Communication.* University of Illinois Press. 1949. ISBN 0-252-72548-4 .

[6] *The Calgary Corpus*. URL:<http://corpus.canterbury.ac.nz/descriptions/#calgary>.

Zoznam príloh

Príloha 1: Manuál k programu lz77
Príloha 2:.Manuál k programu lz78
Príloha 3: Manuál ku knižnici LZ77
Príloha 4:.Manuál ku knižnici LZ78
Príloha 5:.Obsah CD
Príloha 6: CD

Príloha 1:.Manuál k programu lz77

Preklad programu: make

Spustenie programu:

lz77 -h
pre nápovedu

Kompresia

lz77 -c "cesta k vstupnému súboru" "cesta k výstupnému súboru"
pre kompresiu súboru zadaného cestou vstupného súboru.
Ak nebude zadaná cesta výstupného súboru, program pridá k ceste vstupného súboru príponu ".lz77"

Pri kompresii sú voliteľné ďalšie dva parametre, ktoré nastavujú veľkosti vyhľadávacieho a predvídacieho bufferu. A to tak, že:
lz77 -c -s "veľkosť vyhľadávacieho bufferu" -l "veľkosť predvídacieho bufferu" "cesta k vstupnému súboru" "cesta k výstupnému súboru"
Ak tieto parametre nebudú nastavené, použijú sa predvolené hodnoty.

Dekompresia

lz77 -d "cesta k vstupnému súboru" "cesta k výstupnému súboru"
pre dekompresiu súboru zadaného cestou vstupného súboru
Ak nebude zadaná "cesta výstupného súboru" a vstupný súbor má príponu ".lz77", tak sa bude za cestu výstupného súboru považovať "cesta vstupného súboru bez tejto prípony.

Príloha 2:.Manuál k programu lz78

Preklad programu: make

Zložka bin obsahuje program preložený na merlin.fit.vutbr.cz

Spustenie programu:

lz78 -h
pre nápovedu

Kompresia

lz78 -c "cesta k vstupnému súboru" "cesta k výstupnému súboru"
pre kompresiu súboru zadaného cestou vstupného súboru.
Ak nebude zadaná cesta výstupného súboru, program pridá k ceste vstupného súboru príponu ".lz78"

Pri kompresii je voliteľný ďalší parameter, ktorý nastavuje veľkosť slovníka. A to tak, že:
lz78 -c -s "veľkosť slovníka" "cesta k vstupnému súboru" "cesta k výstupnému súboru"
Ak hodnota veľkosti slovníka nebude nastavená, použije sa predvolená hodnota.

Dekompresia

lz78 -d "cesta k vstupnému súboru" "cesta k výstupnému súboru"
pre dekompresiu súboru zadaného cestou vstupného súboru
Ak nebude zadaná "cesta výstupného súboru" a vstupný súbor má príponu ".lz78", tak sa bude za cestu výstupného súboru považovať "cesta vstupného súboru bez tejto prípony.

Príloha 3: Manuál ku knižnici LZ77

```
Názov súboru knižnice: LZ77.h

Hlavná trieda : LZ77

Rozhranie:

// pomocou tejto metódy sa nastaví vstupný tok, ktorý je daný
parametrom
void setInput(std::ifstream *newInput);

// pomocou tejto metódy sa nastaví výstupný tok, ktorý je daný
parametrom
void setOutput(std::ofstream *newOutput);

// parameter sBufferLen nastaví veľkosť vyhľadávacieho buffera
// parameter laBufferlen nastaví veľkosť predvídacieho buffera
void compress(unsigned int sBufferLen, unsigned int laBufferlen);//
metóda, ktorá zaháji kompresiu

void decompress();// metóda, ktorá zaháji dekompresiu
```

Príloha 4: Manuál ku knižnici LZ78

Názov súboru knižnice: LZ78.h

Hlavná trieda : LZ78

Rozhranie:

```
// pomocou tejto metódy sa nastaví vstupný tok, ktorý je daný parametrom
void setInput(std::ifstream *newInput);

// pomocou tejto metódy sa nastaví výstupný tok, ktorý je daný parametrom
void setOutput(std::ofstream *newOutput);

// parameter size určuje maximálnu veľkosť slovníka
void compress(unsigned int size);// metóda, ktorá zaháji kompresiu

void decompress();// metóda, ktorá zaháji dekompresiu
```

Príloha 4: Obsah CD

```
lz77/app/main.cpp – zdrojový kód aplikácie lz77
lz77/app/Makefile

lz77/bin/lz77 – preložená aplikácia lz77

lz77/lib – knižnica LZ77
lz77/lib/BitReader.cpp
lz77/lib/BitReader.h
lz77/lib/BitWriter.cpp
lz77/lib/BitWriter.h
lz77/lib/Buffer.cpp
lz77/lib/Buffer.h
lz77/lib/LaBuffer.cpp
lz77/lib/LaBuffer.h
lz77/lib/LZ77.cpp
lz77/lib/LZ77.h – hlavný súbor knižnice
lz77/lib/SBuffer.cpp
lz77/lib/SBuffer.h

lz77/manual_app – manuál k programu lz77
lz77/manual_lib – manuál ku knižnici  LZ77


lz78/app/main.cpp – zdrojový kód aplikácie lz78
lz78/app/Makefile

lz78/bin/lz78 – preložená aplikácia lz78

lz78/lib – knižnica LZ77
lz78/lib/BitReader.cpp
lz78/lib/BitReader.h
lz78/lib/BitWriter.cpp
lz78/lib/BitWriter.h
lz78/lib/LZ78.cpp
lz78/lib/LZ78.h – hlavný súbor knižnice
lz78/lib/Node.cpp
lz78/lib/Node.h

lz78/manual_app – manuál k programu lz78
lz78/manual_lib – manuál ku knižnici  LZ78
```