UNIVERZITA KARLOVA

Filozofická fakulta

Ústav informačních studií a knihovnictví

Studijní program: Informační studia a knihovnictví

Studijní obor: Informační studia a knihovnictví

Bakalářská práce

Ondřej Malý

Porovnání vybraných metod komprese textových dat

Comparison of selected textual data compression methods

Vedoucí práce: prof. RNDr. Jiří Ivánek, CSc.

Praha 2023

PODĚKOVÁNÍ

Rád bych poděkoval panu profesoru RNDr. Jiřímu Ivánkovi, CSc. za vedení mé bakalářské práce a za cenné rady.

O.M.

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně, že jsem řádně citoval všechny použité prameny a literaturu a že práce nebyla využita v rámci jiného vysokoškolského studia či k získání jiného nebo stejného titulu.

V Praze dne 28. listopadu 2023

Ondřej Malý

ABSTRAKT

Bakalářská práce se zaměřuje na porovnání vybraných metod užívaných při kompresi dat textového charakteru. V úvodu práce je představena obecná problematika komprese dat, základní členění, terminologie a způsoby měření komprese. Teoretická část pojednává o vybraných kompresních metodách a algoritmech, na kterých jsou postaveny. Zaměřuje se zejména na algoritmy, které tvoří základ této problematiky a od kterých bylo odvozeno mnoho dalších a dodnes využívaných. V praktické části se věnujeme vybraným kompresním, resp. archivačním programům a porovnáváme algoritmy ze kterých vycházejí, jejich dokumentaci a dále na vybraných textových datech porovnáváme jejich efektivitu, a to jak v rámci programu samotného, tak mezi s sebou.

KLÍČOVÁ SLOVA

komprese, komprimace, dekomprese, dekomprimace, bezeztrátová komprese, archivace, LZ77, LZ78, LZW, Huffmanovo kódování, Shannon-Fanovo kódování, Burrows-Wheelerova transformace

***

ABSTRACT

The bachelor thesis focuses on the comparison of selected methods used in text data compression. In the introduction of the thesis, the general issues of data compression, basic breakdown, terminology and methods of compression measurement are presented. The theoretical part discusses the selected compression methods and the algorithms on which they are based. It focuses in particular on the methods that form the basis of this problem and from which many other methods still in use today were derived. In the practical part, we discuss selected compression or archiving programs and compare the methods on which they are based and their documentation and, furthermore, we compare their effectiveness on selected text data, both within the program itself and with each other.

KEYWORDS

compressions, decompression, lossless compression, archive management, LZ77, LZ78, LZW, Huffman coding, Shannon-Fano algorithm, Burrows–Wheeler transform

OBSAH

1. ÚVOD

S rozvojem informačních technologií, ke kterému došlo ve 20. století, vzrostlo množství informací, které je generováno a to automaticky (například metadata, logy) či lidskou činností. To s sebou nese jednak potřebu tyto informace uchovávat, ale také potřebu je komunikovat, přenášet z jednoho zařízení na jiné. V tomto ohledu mají kompresní algoritmy nezastupitelnou úlohu. Jiroušek jako důvod pro využívání komprese uvádí dva důvody: za prvé, „*co nejlepší využití kapacity datových nosičů*“, za druhé, „*co nejlépe využít kapacitu přenosových kanálů*“ (Jiroušek 2006, s. 158). Jak uvádí Mlýnková, data samotná ze své podstaty obsahují téměř vždy redundantní informace, způsobené například duplicitami symbolů či frází, a proto je cílem kompresních algoritmů tato data transformovat do podoby s minimální redundancí, ale se zachováním informace (Mlýnková 2008, s. 193).

Na problematiku komprese dat je možné nahlížet optikou dvou vědních oborů. Na prvním místě stojí matematika, na jejímž teoretickém základě tato disciplína stojí. Druhým a neméně důležitým oborem, který tyto teoretické poznatky implementuje do praktického využití, je oblast informatiky, konkrétně obor programovaní. S kompresí se však setkáváme i v rámci jiných vědních disciplín. Jmenujme například informační vědu, která se mimo jiné zabývá kódováním informací, nebo kryptografii, která zkoumá problematiku šifrování.

V rámci této práce se budeme věnovat pouze datům textového charakteru (myšleno alfanumerická data). Ta jsou specifická tím, že na ně prakticky není možné využít algoritmy tzv. ztrátové komprese, neboť by došlo ke ztrátě/změně informace (vyjma specifických příkladů – prázdné řádky na konci textu apod.). Proto budou představeny pouze algoritmy tzv. bezeztrátové komprese.

Smyslem této práce je představit pouze ty algoritmy, které stály u zrodu této disciplíny nebo tvoří základ dalších kompresních algoritmů, jsou implementovány do nejrozšířenějších programů, nebo dle mínění autora, jsou nějakým způsobem zajímavé. A protože je komprese dat v praxi často doplňována o tzv. transformační metody, popíšeme v této práci způsob, na kterém jsou postaveny a konkrétní mechanismus jednoho ze jejich zástupců. Tato práce naopak neusiluje o úplný výčet a popis všech algoritmů, které existují, neboť to by bylo nad její rámec.

Běžný uživatel se s problematikou komprese dat setkává převážně skrze archivační či kompresní programy, o nichž pojednáme níže v této práci. Uvědomujeme si však, že problematika komprese textových dat se týká i databází, zejména relačních, jmenujme například Oracle, PostgreSQL, či Microsoft SQL Server. Každá z databází nějakým způsobem implementuje kompresní algoritmy a umožňuje uživatelům s komprimovanými daty pracovat (vyhledávat, mazat, upravovat). Byť se z našeho pohledu jedná o mimořádně zajímavou problematiku, v této práci se jí nebudeme věnovat a z praktického hlediska se zaměříme spíše na archivační a kompresní programy.

1.1 Cíle

Cílem teoretické části této práce je pojednat o problematice komprese dat a ukázat, jakým způsobem kompresi členíme. Zaměříme se na tzv. metody bezeztrátové komprese, které nacházejí uplatnění při kompresi dat textové povahy – to jsou data, která jsou tvořena kombinací znaků a číslic a jejichž sekvenčnost je nositelem informace (může se jednat o texty, ale i tabulky a podobně). Nejprve představíme základní a jednoduché způsoby komprese dat, tedy tzv. metodu potlačení nul a algoritmus proudového kódování (často označovanou zkratkou RLE). Dále se zaměříme na představení skupiny tzv. slovníkových metod, a to konkrétně na skupinu algoritmů, která se v literatuře označuje jako Lempel - Ziv - Welsch. Následně popíšeme dva zástupce tzv. statistických metod, tedy mechanismus Huffmanova a Shannon-Fanova kódování. Protože se před samotným procesem komprese dat často využívá tzv. datová transformace, kdy jsou data upravena na „*lépe komprimovatelná*“, tak představíme jednu z těchto metod, a to Burrows-Wheelerovu transformaci.

V praktické části se pak zaměříme na implementaci těchto algoritmů do konkrétního softwarového programu. Zvolíme tři zástupce na trhu běžně dostupného a používaného softwaru, konkrétně programy 7-Zip, WinRar a WinZip. Detailně se zaměříme na to, jaké algoritmy implementují (bude-li to možné odvodit z názvu či nalézt v dokumentaci). Konkrétně tedy popíšeme uživatelské rozhraní a příkazovou řádku, efektivitu komprese a dokumentaci k jednotlivým programům. Položíme si čtyři otázky, na které budeme hledat odpovědi:

(1) Má uživatel možnost v rozhraní programu, anebo příkazové řádce programu zvolit konkrétní algoritmus, který popíšeme v teoretické části?
(2) Používají dané programy jednotnou terminologii ve značení těchto algoritmů, nebo mají vlastní názvosloví?
(3) Dosahují programy stejné efektivity komprese (kompresní poměr)?
(4) Pomůže uživateli softwarová dokumentace identifikovat, jaké algoritmy program využívá, pokud to nebude zjevné z jejich názvu?

Budeme zkoumat, jaké možnosti ovlivnit kompresi (ve vztahu k výběru algoritmu) rozhraní uživateli nabízí a zda existuje jednotná terminologie v rámci zkoumaných programů. Na vybraném souboru textových dat provedeme měření – vypočteme kompresní poměr pro všechny typy kompresních algoritmů, které program nabízí, a porovnáme je mezi s sebou. Pokusíme se zodpovědět otázku, zda je možné porovnat jednotlivé programy mezi s sebou, a to na základě algoritmu, který využívají. Zhodnotíme dokumentaci k jednotlivým programům, a pokusíme se tak zodpovědět, zda může uživatel jednoduše získat informaci o tom, jakým algoritmem program komprimuje.

1.2 Metody

Jak již bylo popsáno výše, uvědomujeme si, že na problematiku komprese dat je možné nahlížet prismatem matematiky, informatiky či jiných souvisejících oborů. Vzhledem ke studijnímu oboru autora, tj. informační vědy, která bývá považována za mezioborovou disciplínu, budeme k této práci přistupovat tímto způsobem, tj. bez výrazné inklinace k matematice či informatice, bude-li to možné.

1.3 Zdroje

Literatura věnující se problematice kompresních algoritmů je v českém prostředí značně omezená. Lze ji rozdělit na knižních publikace, skripta, a webové stránky. V případě knižní literatury se jedná o edice, které vyšly zejména v devadesátých letech a po roce 2000 (např. Morkes 1998 a Jiroušek 2006). Ve většině případů již nejsou dostupné v knihkupectvích ani nebyly digitalizovány či reeditovány. Častěji se tak s touto problematikou setkáváme v příbuzné literatuře (zejména z oboru informatiky), kde je jí věnována jedna či dvě kapitoly (např. Gála 2009). Na našem trhu jsou pak lépe dostupná a často i aktuálnější, co se obsahu týče, skripta vydávaná českými univerzitami (např. Pokorný 2005). Jedná se zejména o technické či ekonomické univerzity, které nabízejí studijní programy blízké informatice. V českém jazyce pak lze nalézt i několik webových stránek, které se této problematice poměrně fundovaně věnují (např. Foltýnek 2008). Zahraniční literatura, zejména ta knižní, je mnohem pestřejší, co do nabídky, a i její aktuálnosti (např. Salomon 2005 a Sayood 2017).

2. KOMPRESE A DEKOMPRESE

Na úvod je třeba vyjasnit základní pojem *komprese*, resp. *komprimace* a s ním související pojmy. V českém prostředí se jako synonymum k pojmu *komprese* užívá nejčastěji pojem *komprimace*. Setkáme se ale i s pojmem *zhušťování* či méně formálním *zabalení*. Význam je ve všech variantách stejný. Kompresí je myšlen proces, při kterém se ze vstupního souboru o definované velikosti vytvoří soubor nový (ideálně) s velikostí menší, než byla u původního souboru - tzv. *komprimovaný/výstupní soubor*. Reverzním procesem ke kompresi je proces zvaný *dekomprese*, či *dekomprimace*. Méně formálně také *rozbalení*.

Velikost souboru obvykle měříme jednotkou zvanou *Byte*, nebo česky *Bajt* a jeho násobky podle SI soustavy (např. „*kilo*-“ „*mega*-“ „*tera*-“). Obvykle se označuje „*B*“. Bajt je složen z osmi *bitů*[1], které označujeme „*b*“. Bit je považován za základní jednotku informace a může nabývat pouze dvou hodnot 1 a 0, což reprezentuje stavy zapnuto/vypnuto, nebo booleovské operátory pravda/nepravda (Gála 2009, s. 400). Do jednoho bajtu tak lze zakódovat 2^8 znaků, tedy 256 znaků (tamtéž, s. 447).

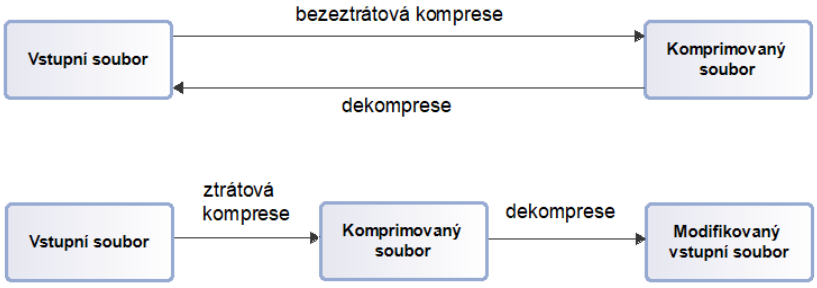
2.1 Základní pojmy

V této práci se budeme setkávat s několika základními pojmy, jejichž užívání není nikterak ustáleno a může se lišit na základě oboru či zvyklostí v daném jazyce. Nejprve tedy představíme jejich význam, tak, jak jej definuje akademický slovník cizích slov. (1) Pojem ***komprese*** je definován jako „*stlačení, stlačování, zmenšení objemu něčeho*“ a s ním související pojem (2) ***komprimace*** jako „*zhušťování, zhuštění, komprese dat pro potřeby archivace*“ (Kraus 2006). Jak je z významu patrné, je možné oba pojmy používat jako synonyma, a i v české literatuře je tak činěno (stejně jako s pojmy dekomprese a dekomprimace). Další z dvojice pojmů, které bývají mnoha českými autory využívány zástupně jsou (3) ***metoda*** a (4) ***algoritmus***. První z termínů slovník definuje jako „*způsob, jak dosáhnout něj. teoretického i praktického cíle*“ druhý pak jako „*předpis konečného počtu kroků, kterými je možno řešit stejnorodé úkoly*“ (tamtéž). Zde již z definice vyplývají rozdíly. V anglické literatuře se v souvislosti s konkrétním popisem mechanismu komprese setkáváme spíše s používáním anglického ekvivalentu slova algoritmus – viz například „*Huffman coding algorithm*“ (Sayood 2017, s. 41). Oproti tomu česká literatura používá oba pojmy jako synonyma. Například Čapek hovoří o metodě proudového kódování RLE (Čapek 2000, s. 15), kdežto Jiroušek užívá pojmu algoritmus RLE (Jiroušek 2006, s. 161). Osobně se přikláníme spíše k užití slova algoritmus, a proto je budeme v této práci převážně používat.

[1] Viz také Binary digit.

2.2 Rozdělení kompresních metod

Za základní členění kompresních metod lze považovat rozdělení na *ztrátovou* a *bezeztrátovou* kompresi[2]. Rozdíl mezi oběma metodami spočívá ve schopnosti rekonstruovat výstupní soubor (dekomprese) tak, že je 100% identický se souborem původním, pak hovoříme o bezeztrátové kompresi (Čapek 2000, s. 3).



Obrázek 1: Schéma bezeztrátové a ztrátové komprese

Jak uvádí například Jiroušek, metody ztrátové komprese nacházejí uplatnění zejména pro ty soubory, kde odstranění části informace nevede ke znehodnocení souboru jako celku – například obrázky či zvuky. U těchto souborů je možné odstranit ty části kódu, které nesou informaci, která představuje hraniční hodnotu vnímavosti lidskými smysly. Tento přístup však prakticky nelze využít při komprimaci textových souborů, neboť zde by ztráta vedla ke znehodnocení souboru a ztrátě informace. Proto u komprimací textových dat zpravidla využíváme komprimaci bezeztrátovou (Jiroušek 2006, s. 159).

Podle toho, zda je algoritmus schopen přizpůsobit se komprimovaným datům, nalezneme v literatuře rozdělení na *adaptivní* a *neadaptivní* kompresi. Autoři však tyto hranice chápou různě, a proto v literatuře panuje nejednoznačnost (Jiroušek 2006, s. 159). Například Pokorný rozlišuje ještě třetí typ členění, a to *semiadaptivní* kompresi. Tedy situaci, kdy se „*model vytvoří pro každý text zvlášť a uloží se ke komprimovaným datům*“ (Pokorný 2005, s. 158).

Z hlediska času potřebného na kompresi a dekompresi rozlišujeme algoritmy na *symetrické* a *asymetrické*. Symetrické potřebují na proces komprese srovnatelný čas, jako na proces dekomprese. V případě asymetrických algoritmů může být časově náročnější buď proces komprese, nebo dekomprese, v závislosti na použitém typu algoritmu (Jiroušek 2006, s. 159).

2.3 Měření výkonnosti/efektivity komprese

Existuje několik ukazatelů, pomocí nichž měříme výkonnost komprimačního procesu. Jedná se zejména o poměrové metody, které sledují velikost vstupního a výstupního souboru, a pak metody, které měří

[2] V anglických zdrojích nalezneme nejčastěji označení „*lossy compression*“ a „*lossless compression*“.

rychlost procesu komprese, resp. dekomprese. V literatuře se těchto ukazatelů vyskytuje více, proto uvádíme jen ty, které jsou dle našeho názoru nejrelevantnější.

Kompresní poměr[3] *(K_p)* představuje poměr velikosti výstupního souboru ku velikosti vstupního souboru (Salomon 2005, s. 62–63):

$$K_p = \frac{velikost\ výstupního\ souboru\ [b]}{velikost\ vstupního\ souboru\ [b]}$$

Zde platí, že je-li $K_p = 1$, pak nedošlo k žádné kompresi; pokud je $K_p < 1$ pak byl soubor zkomprimován; a konečně $K_p > 1$ znamená, že soubor po kompresi je větší, než byl soubor vstupní. Od kompresního poměru bývají odvozovány další metriky uveďme například *bpb* (bit per bit), *bpc* (bit per character) a dále například *bpp* (bit per pixel) a další. (Salomon 2005, s. 62–63).

Dalším ukazatelem je tzv. *Kompresní faktor*[4] *(K_f)*; i ten můžeme považovat za odvozený od kompresního poměru, neboť je jeho převrácenou hodnotou. Představuje tedy podíl velikosti vstupního souboru ku velikosti výstupního souboru (Jiroušek 2006, s. 160):

$$K_f = \frac{velikost\ vstupního\ souboru\ [b]}{velikost\ komprimovaného\ souboru\ [b]}$$

V případě, že je $K_f = 1$, pak nedošlo k žádné kompresi; pokud je $K_f > 1$ pak byl soubor zkomprimován; $K_f < 1$ znamená že soubor po kompresi je větší, než byl soubor vstupní.

Jiroušek také upozorňuje na tzv. *kompresní zisk*[5] (K_z), jehož „*výhodou je, že srovnávání kompresních zisků se provádí jejich odečítáním*“ (Jiroušek 2006, s. 160) a jenž je definován:

$$K_z = \log_e K_f$$

Ve vztahu k velikosti souboru bychom v literatuře nalezli ještě další metriky, výše uvedené však bývají nepoužívanější a pro potřeby naší práce jsou dostatečné.

Stručně ještě zmiňme další proměnnou zvažovanou při kompresi dat, a to je měření času. Rozlišujeme čas potřebný pro kompresi dat a čas potřebný pro jeho dekompresi. Při vytváření archivu založeném na komprimovaných souborech je proto nutné zohlednit, jakým způsobem bude s komprimovanými soubory pracováno. Pokud budujeme archiv pouze jako zálohu dat, tedy statický archiv, časové hledisko komprese či dekomprese nehraje tak významnou roli jako velikost souboru a místa, které bude zabírat na disku. Naopak v případě archivu, do kterého budeme často přistupovat a potřeba komprese/dekomprese bude značná, musíme zvolit takový typ komprimace, který bude z hlediska času optimální.

[3] V anglické literatuře je označován jako „*compression ratio*“.
[4] V anglické literatuře je označován jako „*compression factor*“.
[5] V anglické literatuře je označován jako „*compression gain*“.

3. VYBRANÉ KOMPRESNÍ ALGORITMY

V této kapitole představíme výběr několika algoritmů, které nacházejí uplatnění jako součást tzv. archivačních programů (viz dále), a které se využívají pro kompresi dat textového charakteru. Nejedná se o úplný výčet, neboť ten by byl nad rámec možností této práce. Detailněji popisujeme ty algoritmy, které ve své původní podobě nacházejí uplatnění dodnes, nebo se staly základem, ze kterého byly odvozovány algoritmy nové.

3.1 Metoda potlačení nul

Tato metoda patří mezi jednu z nejstarších a relativně nejjednodušších technik komprese. Je vhodná zejména pro řetězce s častým výskytem jednoho znaku. V komprimovaném textu vyhledává často se opakující znak (většinou mezera či nula). Pokud je nalezen výskyt takovéto sekvence, jsou znaky nahrazeny uspořádanou dvojicí: *indikátorem komprese* (I_C) a *počtem výskytu nahrazovaného znaku* (Čapek 2000, s. 8).

Vstupní text	F	0	0	0	0	0	J	I	L	M	N	P
Komprimovaný text	F	I_C	4	J	I	L	M	N	P			

Tabulka 1: Metoda potlačení nul

Jak je patrné, je tento algoritmus efektivní při výskytu 3 a více znaků za sebou. Při výskytu jednoho komprimovaného znaku dochází k rozšíření na znaky dva (indikátor komprese + počet výskytu). Při výskytu dvou komprimovaných znaků tak nedochází k vůbec žádné kompresi.

Omezení algoritmu spočívá ve volbě znaku, jenž reprezentuje indikátor komprese, a který se nesmí v řetězci vyskytovat. Tuto podmínku však může být v praxi obtížné splnit, a proto lze tento algoritmus kombinovat s tzv. *rozšířenou znakovou sadou*.[6] S tím je spojeno zavedení dvou přepínačů mezi sadami – přepínač tam (P_T) a přepínač zpět (P_Z) (Čapek 2000, s. 9-10).

Vstupní text	F	0	0	0	0	0	J	I	L	M	N	P
Komprimovaný text	F	P_T	I_C	P_Z	4	J	I	L	M	N	P	

Tabulka 2: Metoda potlačení nul s použitím rozšířené množiny znaků

3.2 Proudové kódování (RLE)

Algoritmus RLE[7] představuje jisté zobecnění metody potlačení nul. Rozdíl spočívá v možnosti redukovat nejen jeden (konkrétní) po sobě se opakující znak, ale všechny sekvence po sobě se opakujících znaků – nazývané *proud*. Výhodou tohoto algoritmu je možnost aplikovat jej na jakýkoliv

[6] V angličtině se používá slovní spojení „extended character set".
[7] RLE je zkratka anglického názvu algoritmu „*Run Length Encoding*".

typ dat a také rychlost komprese či dekomprese. Nevýhodou je, že kompresní poměr je silně závislý na charakteru dat a ve většině případů nemůže konkurovat, co do efektivity komprese, jiným algoritmům (Morkes 1998, s. 21).

Vstupní text	F	H	U	U	U	U	K	L	L	L	L	M
Komprimovaný text	F	H	I_C	U	4	K	I_C	L	4	M		

Tabulka 3: Proudové kódování

3.3 Skupina algoritmů Lempel – Ziv – Welsch

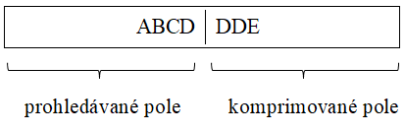
V roce 1977 publikovali Abraham Lempel a Jacob Ziv algoritmus označovaný jako LZ77[8], jehož podstata spočívá ve využití tzv. *posuvného okna*. O rok později stejná dvojice publikovala algoritmus nový, označovaný jako LZ78[9], využívající při komprimaci tzv. *dynamický slovník*. Tento algoritmus byl později modifikován Terry Welchem a používá se pro něj označení LZW (Sayood 2017, s. 134-141).

V průběhu času byly tyto algoritmy dále upravovány mnoha různými autory, a tak dnes hovoříme o třídách či „*rodinách*“ těchto algoritmů. Algoritmy LZ77, LZ78, LZW však tvoří základ, a proto o nich pojednáme detailněji.

3.3.1 Algoritmus LZ77

Z hlediska členění řadíme algoritmus LZ77 mezi slovníkovou metodu komprese, která je asymetrická a adaptivní.

Základem tohoto algoritmu je využití principu tzv. *posuvného okna*[10], které je rozděleno na dvě části. V literatuře nalezneme různá pojmenování této dvojice: *prohledávané pole* a *komprimované pole* (Jiroušek 2006, s. 164-165); *prohlížecí okno* a *aktuální okno* (Morkes 1998, s. 33); *zakódovaná* a *nezakódovaná data* (Foltýnek 2008) apod.[11]



Obrázek 2: Posuvné okno

Algoritmus skrze posuvné okno nahlíží do souboru, který má být komprimován a snaží se nalézt co nejdelší řetězec opakujících se znaků. Je-li takovýto řetězec nalezen, pak do výstupního souboru uloží

[8] Je znám také pod zkratkou LZ1.
[9] Nebo také LZ2.
[10] V anglické literatuře se používá pojem „*sliding window*“.
[11] V anglické literatuře jsou nejčastěji použity výrazy „*search buffer*“ a „*look-ahead buffer*“.

pouze informaci, kde se v prohlížecím okně řetězec nachází. Tato informace má nejčastěji podobu uspořádané trojice (*p, l, c*), kde *p* je pozice začátku nalezené sekvence v prohledávaném poli, *l* je délka této sekvence a *c* je následující symbol. (Jiroušek 2006, s. 164-166)

Celý cyklus si demonstrujeme na příkladu s řetězcem „*ABCDDDE*“. Níže uvedená tabulka představuje jednotlivé iterace komprese.

Iterace	Prohledávané pole	Komprimované pole	Výstup	Posunutí okna
1		ABCDDDE	(0, 0, A)	1
2	A	BCDDDE	(0, 0, B)	1
3	AB	CDDDE	(0, 0, C)	1
4	ABC	DDDE	(0, 0, D)	1
5	ABCD	DDE	(1, 2, E)	3
6	ABCDDDE			

Tabulka 4: Cyklus komprimace algoritmem LZ77

Komprimovaný výstup tedy vypadá takto: „*(0, 0, A) (0, 0, B) (0, 0, C) (0, 0, D) (1, 2, E)*“.

Dekomprese souboru je poměrně jednoduchá – postupně jsou procházeny zakódované trojice a zapisovány původní znaky. Ukazatel s hodnotami (0, 0, *c*) odkazuje na původní samostatný symbol, kde pro první čtyři trojice tak získáme znaky ABCD. Následuje poslední trojice (1, 2, E). Tedy první symbol v prohledávaném poli, tedy „*D*“, opakující se 2x a s následujícím symbolem „*E*“.

3.3.2 Algoritmus LZ78

Algoritmus LZ78 řadíme mezi symetrickou a adaptivní kompresi. Je založen na dynamicky vytvářeném slovníku opakujících se sekvencí řetězců. Do výstupního souboru jsou zapisovány dvojice (*i, s*), kde *i* představuje ukazatel do indexu slovníku dříve nalezených znaků či řetězců a hodnota *s* reprezentuje bezprostředně následující symbol v komprimovaném souboru. Samotný slovník je na začátku komprimace prázdný, je pouze iniciován indexem 0 a prázdným řetězcem. (Jiroušek 2006, s. 166-167)

Na řetězci „*ABCAK*“ si demonstrujeme mechanismus komprese.

Krok	Index	Řetězec	Výstup
1	0		
2	1	A	(0, A)
3	2	B	(0, B)
4	3	C	(0, C)
5	4	AK	(1, K)

Tabulka 5: Komprimace algoritmem LZ78

Výstupní soubor je tedy „*(0, A) (0, B) (0, C) (1, K)*“. Pro upřesnění: hodnota (*0, s*) tedy odkazuje na prázdný řetězec s indexem rovným 0, po němž bezprostředně následuje původní (samostatný) symbol.

Výhodou tohoto algoritmu je, že vytvořený slovník nemusí být součástí komprimovaného souboru, či dokonce, je-li při kompresi zaplněna operační paměť vyhrazená pro slovník, je možné jej smazat a stavět znovu (Jiroušek 2006, s. 166–167). Dekomprese pak probíhá postupným prohledáváním slovníku podle hodnoty indexu a nahrazením odpovídajícího symbolu či sekvence.

3.3.3 Algoritmus LZW

Tento algoritmus je odvozen z výše uvedeného algoritmu LZ78 a spadá do kategorie symetrických a semiadaptivních kompresí.

Mechanismus komprese je postaven na postupném vytváření slovníku tzv. *frází*. To jsou řetězce složené ze znaků, které se vyskytly ve zpracované části vstupního souboru. Počet znaků, z nichž se fráze skládá, udává její délku. Na počátku komprese jsou tak ve slovníku fráze o délce 1, tedy všechny unikátní znaky, které se vyskytují v komprimovaném souboru. Postupně je procházen komprimovaný soubor a slovník rozšiřován o nové fráze a současně jsou ze vstupního souboru tyto fráze odstraňovány. Výstupem komprese je soubor indexů – numerických hodnot udávajících pořadí frází ve slovníku (Pokorný 2005, s. 161).

Tento mechanismus si demonstrujme na řetězci složeném ze čtyř znaků, konkrétně „*EFHGFFFHG*“.

V iniciační fázi je slovník naplněn frázemi o délce 1, tedy unikátními znaky.

Fráze	Index
E	0
F	1
H	2
G	3

Tabulka 6: Slovník ve fázi inicializace

Následuje samotná komprese.

Krok	Vstupní řetězec	Nalezená fráze	Výstup
1	EFHGFFFHG	E	0
2	FHGFFFHG	F	1
3	HGFFFHG	H	2
4	GFFFHG	G	3
5	FFFHG	F	1
6	FFHG	FF	8
7	HG	H	2
8	G	G	3

Nová fráze	Index
EF	4
FH	5
HG	6
GF	7
FF	8
FFH	9
HG	10

Tabulka 7: Mechanismus komprese LZW

Výstup komprese je pak tvořen sekvencí kódu: „*01231823*“.

Dekompresní mechanismus vyžaduje, aby byl příjemce obeznámen s tabulkou symbolů, ze kterých se skládá komprimovaný soubor a jejich indexy (ta která vzniká ve fázi inicializace komprese). Algoritmus prochází komprimovaný řetězec a transformuje výstup zpět na původní soubor. Současně s tím v každém kroku rozšiřuje slovník o nové fráze a jím odpovídající indexy a to tak, že nová fráze se skládá z přeloženého znaku aktuálního kroku plus prvního přeloženého znaku následujícího ve zkomprimovaném výstupu. Index je novým frázím přiřazován podle nejvyšší hodnoty indexu, která se již ve slovníku nachází + 1. Průběh demonstruje následující tabulka:

Krok	Komprimovaný řetězec	Nalezená kódovaná fráze	Výstup
1	01231823	0	E
2	1231823	1	F
3	231823	2	H
4	31823	3	G
5	1823	1	F
6	823	8	FF
7	23	2	H
8	3	3	G

Nová fráze	Index
EF	4
FH	5
HG	6
GF	7
FF	8
FFH	9
HG	10

Tabulka 8: Mechanismus dekomprese LZW

Zde je důležité upozornit na kroky 5 a 6, kde pro vznik nové fráze je využito indexu 8, který se ve slovníku ještě nenachází. K tomuto jevu může dojít, jen pokud je první znak z aktuálního kroku (6) stejný, jako první znak z předcházejícího kroku (5). Na základě toho je možné chybějící frázi do slovníku doplnit (Pokorný 2005, s. 161–163).

Výstup: „*EFHGFFFHG*“

3.4 Huffmanovo kódování

Huffmanovo kódování nese název po svém objeviteli Davidu Huffmanovi a datuje se do roku 1952. Jedná se prefixový kód, který je popsán pomocí binárního stromu. Algoritmu se také říká *kód s minimální redundancí* (Pokorný 2005, s. 159). Zajímavým faktem je, že Huffman tento princip vyvinul při studiu na univerzitě, jako součást školního projektu (Sayood 2017, s. 41).

Na počátku komprese algoritmus nejprve analyzuje soubor, a vypočítá pravděpodobnosti výskytu, resp. četnosti jednotlivých znaků a dle těchto hodnot je seřadí sestupně. Jednotlivým znakům je poté přiřazen binární kód[12] (0, 1) a to tak, že znaky s nejvyšší četností získají (bitově) nejkratší kód, a naopak méně četné znaky mohou získat bitově delší kódy. Toto přiřazení kódu je *de facto* tvorba binárního stromu. Ten vytváříme od nejméně četných výskytů a postupujeme ke kořeni stromu. Nejprve vybereme

[12] Existují však i verze pro libovolně velkou (obecnou) kódovací abecedu (Pokorný 2005, s. 159–160).

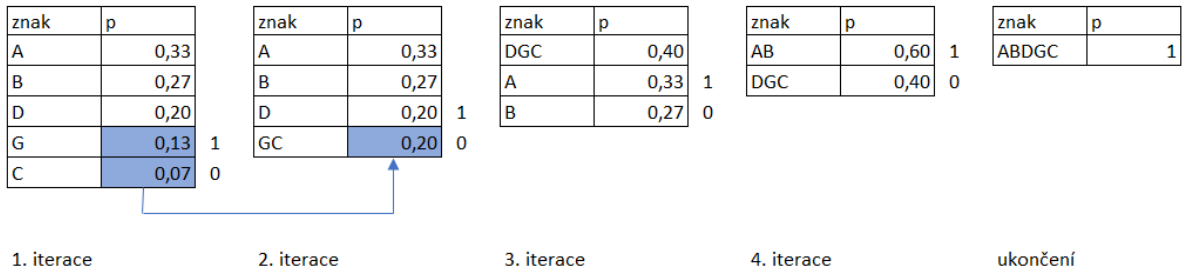
poslední dva znaky (ze seřazené tabulky), které představují koncové listy stromu. Ty nám utvoří uzel stromu, jehož hodnota odpovídá součtu četností/pravděpodobností „*listů*“, které spojuje. Postupujeme stejným způsobem, přeuspořádejme tabulku pravděpodobností a pokračujme dalším, nezpracovaným symbolem od spodu, a to do doby, než dosáhneme kořene stromu. Pro přiřazení kódu jednotlivým symbolům využijeme jejich „*cesty*“ ve stromu. Po nadefinování jednotlivých kódů začne komprese. Vstupní soubor je posupně čten a jednotlivé znaky nahrazovány výstupními sekvencemi bitů (Morkes 1998, s. 39–42).

Mějme například řetězec „*ABACDABBGDBADGA*“. Nejprve vypočteme pravděpodobnosti výskytu jednotlivých znaků[13] a vytvoříme tabulku, kterou seřadíme podle četnosti sestupně.

Znak	Frekvence znaku	Pravděpodobnost výskytu
A	5	0,33
B	4	0,27
D	3	0,20
G	2	0,13
C	1	0,07

Tabulka 9: Přehled znaků a jejich četností analyzovaného řetězce

Následně utvoříme binární strom. Začneme od posledních dvou nejméně četných znaků – tedy „*G*“ a „*C*“. Společně nám utvoří první uzel. Pokračujeme od spodu tabulky dalším nezpracovaným znakem, tedy „*D*“, které umístíme na úroveň prvnímu uzlu. Pokračujeme obdobným způsobem až ke kořeni stromu. Nakonec přiřadíme jednotlivým cestám symboly 0 a 1, jak reprezentuje níže uvedený obrázek.



Obrázek 3: Tvorba Huffmanova binárního stromu

[13] Počet výskytu daněného znaku / celkový počet znaků.

Na základě vytvořeného stromu přiřadíme jednotlivým symbolům kódy:

Znak	Kód
A	11
B	10
D	01
G	001
C	000

Tabulka 10: Přehled znaků a přiřazených kódů

Řetězec „*ABACDABBGDBADGA*“ by tedy bylo možné zakódovat do této sekvence: „*11101100001111010001011011010111*“.

Dekomprese souboru vyžaduje znalost binárního stromu. Algoritmus postupně čte soubor bit po bitu a podle hodnot se pohybuje ve struktuře stromu od kořene po jednotlivé listy (Morkes 1998, s. 41).

3.5 Shannon-Fanovo kódování

Samotný algoritmus vznikl kolem roku 1949, kdy jej nezávisle na sobě jej publikovali autoři Claude Elwood Shannon, Warren Weaver a Robert Mario Fano. Původním účelem kódu bylo zvýšení přenosové kapacity komunikačního kanálu. Algoritmus však našel uplatnění i v kompresi dat (Pokorný 2005, s. 158–159).

Z hlediska členění řadíme tento typ komprese do statistických, asymetrických a semiadaptivních metod.

V případě tohoto kódování lze taktéž využít analogie stromu, který jsme použili při demonstraci Huffmanova kódu. Rozdíl spočívá především v postupu směrem od kořene stromu k jeho listům.

Princip algoritmu je poměrně jednoduchý a lze je popsat těmito kroky:

(1) Vytvoření seznamu symbolů, které se v komprimovaném textu nacházejí.

(2) Vypočtení četnosti (nebo pravděpodobnosti výskytu) jednotlivých symbolů a jejich sestupné seřazení dle této hodnoty.

(3) Rozdělení tohoto seznamu na dvě poloviny, tak aby součet četností (pravděpodobností) byl přibližně stejný.

(4) Přiřazení jednoho bitu symbolům v obou skupinách a to tak, že první skupina bude mít hodnotu 0 a druhá skupina hodnotu 1.

(5) Rekurzivní postup od bodu 3, dokud v každém podseznamu není pouze jeden symbol.

Algoritmus si demonstrujme na následujícím řetězci: „*DBGDABGADGBCDCG*“. Vytvoříme tabulku s jednotlivými znaky, spočítáme jejich četnost a zároveň jí seřadíme sestupně:

Znak	Četnost
D	4
G	4
B	3
A	2
C	2

Tabulka 11: Přehled znaků a přiřazených kódů

Nyní postupujme podle kroku (3) a vytvoříme dvě skupiny:

- První skupina: {D, G} s četností = 8
- Druhá skupina: {B, A, C} se součtem četností = 7

Podle kroku (4) přiřadíme skupinám bit:

Znak	Bit – 1. průchod
D	0
G	0
B	1
A	1
C	1

Tabulka 12: Přiřazení bitu (1. průchod)

Rekurzivně pokračujeme od bodu (3). Skupina {D, G} již neobsahuje více členů, proto v jejím případě bude následovat krok (4). Oproti druhé skupině, kterou rozdělit musíme:

{B, A, C}

- První sub-skupina: {B, A} s četností = 2
- Druhá sub-skupina: {C} s četností = 1

Opět dle kroku (4) přiřadíme bit:

Znak	Bit – 1. průchod	Bit – 2. průchod
D	0	0
G	0	1
B	1	0
A	1	0
C	1	1

Tabulka 13: Přiřazení bitu (2. průchod)

Zbývá nám tedy sub-skupina {B, A}, proto rekurzivně pokračujeme. Krok (3) tedy můžeme přeskočit, protože nám zbývají pouze dva symboly a pokračujeme krokem (4):

Znak	Bit – 1. průchod	Bit – 2. průchod	Bit – 3. průchod
D	0	0	*nepřiřazujeme*
G	0	1	*nepřiřazujeme*
B	1	0	0
A	1	0	1
C	1	1	*nepřiřazujeme*

Tabulka 14: Přiřazení bitu (3. průchod)

Výstupem je tedy tato kódová tabulka, pomocí níž lze provést kompresi a analogicky i dekompresi:

Znak	Kód
D	00
G	01
B	100
A	101
C	11

Tabulka 15: Výstupní kódová tabulka

3.6 Kombinované algoritmy

V této části kapitoly stručně zmíníme několik kompresních mechanismů, které jsou založeny na kombinaci (použití) výše uvedených algoritmů. Důvodem je fakt, že tyto algoritmy jsou často implementovány do kompresních a archivačních programů (jak bude ukázáno níže v praktické části):

Deflate je algoritmus vyvinutý Philem Katzem, který je založený na kombinaci algoritmu LZ77 a Huffmanovým kódováním (BZip2 1996).

Bzip2 byl představen v roce 1996 Julianem Sewardem. Jedná se o kombinaci Huffmanova kódování a Burrows-Wheelerovy transformace. Algoritmus během své existence prošel několika změnami, jeho předchůdce **Bzip** využíval aritmetického kódování (BZip2 1996).

LZMA je algoritmus vytvořený Igorem Pavlovem, který je založen na slovníkové kompresi LZ77 s využitím tzv. Markovových řetězců (Salomon 2007 s. 241-242). Z tohoto algoritmu je odvozena jeho modernější verze označovaná jako **LZMA2** využívaná například programem 7-Zip (7-Zip 2019).

XZ je metoda užívaná programem WinZip, a je založena na LZMA2 kompresi (WinZip 2020).

Zstandard, nebo také **Zstd** je způsob komprimace dat vytvořený Yannem Colletem ze společnosti Facebook,[14] který navazuje na algoritmus LZ77 (a jeho pozdější modifikace), a který je kombinován

[14] Dřívější název společnosti Meta Platforms.

s tzv. entropickým kódováním (META 2018).

PPMd je jedna z variant implementace algoritmu, který navrhl Dmitry Shkarin. V literatuře bývá označována také jako PPMdH (Salomon 2007 s. 241) nebo PPMII (Salomon 2007 s. 227). Tato metoda je řazena do široké skupiny algoritmů, známých pod zkratkou **PPM**, odvozenou z anglického „*Prediction with Partial Match*“ (Sayood 2017, s. 167).

3.7 Metoda užívaná pro zvýšení efektivity komprese: Burrows-Wheelerova transformace

Jak již bylo výše naznačeno, samotná povaha dat má vliv na efektivitu komprese. Zjednodušíme-li tuto tezi, pak například pro data účetního charakteru, která obsahují povětšinou numerické hodnoty, případně symboly měny, budou vhodné jiné kompresní algoritmy než například pro kompresi literárního díla. Obecně lze říct, že často se opakující symboly v řetězci se lépe komprimují, jsou-li pozičně blíže u sebe. Proto se mnohdy před samotnou kompresí využívají techniky, které text transformují, tak aby byl samotný proces komprese efektivnější. Po dekompresi pak následuje konverze zpět na původní text.

Jednou z těchto metod je tzv. Burrows-Wheelerova transformace (BWT), o které níže pojednáváme.

Tato technika transformace byla představena v roce 1994. Její princip spočívá v reverzibilní transformaci textu na formu vhodnější pro vlastní kompresi. Obecný mechanismus je založen na cyklickém procházení textu a na přeskupení symbolů tak, že stejné symboly se přesunou blíže k sobě. Mechanismus transformace a její reverse bude popsán na následujícím příkladu.

Využijeme sousloví „*tento táta*“, které pro zjednodušení upravíme, tj. použijeme kapitálky, odstraníme diakritiku a mezeru nahradíme tečkou. Získáme tedy řetězec „*TENTO.TATA*“. Na jeho konec přiřadíme znak, který se v textu nevyskytuje, a který bude značit ukončení řetězce.[15] V tomto případě použijeme symbol „*%*“, získáme tak řetězec „*TENTO.TATA%*“. Vytvoříme tabulku, tak že každý symbol, bude představovat jeden sloupec:

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	Řetězec
T	E	N	T	O	.	T	A	T	A	%	TENTO.TATA%

Tabulka 16: BWT – první krok

Dále budeme cyklicky vytvářet další řetězce a to způsobem, že každý další řádek bude vytvořen z řádku předchozího dle následující logiky:

Nový řádek vzniká z kopie řádku předchozího, a to od pozice jeho druhého symbolu do pozice symbolu posledního. Tím nám logicky na nově vzniklém řádku zůstává neobsazený poslední symbol, a právě ten naplníme symbolem z první pozice předchozího řádku, který jsme vynechali. Počet iterací, které

[15] V anglické literatuře bývá označován jako „*EOF*“ neboli „*End-Of-File*“.

provedeme se rovná počtu symbolů v řetězci. V našem případě 11 iterací. Získáme tak následující tabulku:

Iterace	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	Nový řetězec
1	T	E	N	T	O	.	T	A	T	A	%	TENTO.TATA%
2	E	N	T	O	.	T	A	T	A	%	T	ENTO.TATA%
3	N	T	O	.	T	A	T	A	%	T	E	NTO.TATA%T
4	T	O	.	T	A	T	A	%	T	E	N	TO.TATA%TE
5	O	.	T	A	T	A	%	T	E	N	T	O.TATA%TEN
6	.	T	A	T	A	%	T	E	N	T	O	.TATA%TENT
7	T	A	T	A	%	T	E	N	T	O	.	TATA%TENTO
8	A	T	A	%	T	E	N	T	O	.	T	ATA%TENTO.
9	T	A	%	T	E	N	T	O	.	T	A	TA%TENTO.T
10	A	%	T	E	N	T	O	.	T	A	T	A%TENTO.TA
11	%	T	E	N	T	O	.	T	A	T	A	%TENTO.TAT

Tabulka 17: BWT - 1 krok

Druhým krokem bude tuto tabulku setřídit podle atributu „*Nový řetězec*“ a to lexikograficky. Tomuto kroku je třeba věnovat velkou pozornost, protože operace řazení používané například databázemi, nebo MS excelem, nemusí řadit vždy lexikograficky (problematika NULL hodnot, speciálních znaků, česká znaková sada).

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	Nový řetězec
.	T	A	T	A	%	T	E	N	T	O	.TATA%TENT
A	%	T	E	N	T	O	.	T	A	T	A%TENTO.TA
A	T	A	%	T	E	N	T	O	.	T	ATA%TENTO.
E	N	T	O	.	T	A	T	A	%	T	ENTO.TATA%
N	T	O	.	T	A	T	A	%	T	E	NTO.TATA%T
O	.	T	A	T	A	%	T	E	N	T	O.TATA%TEN
T	A	T	A	%	T	E	N	T	O	.	TATA%TENTO
T	A	%	T	E	N	T	O	.	T	A	TA%TENTO.T
T	E	N	T	O	.	T	A	T	A	%	TENTO.TATA%
T	O	.	T	A	T	A	%	T	E	N	TO.TATA%TE
%	T	E	N	T	O	.	T	A	T	A	%TENTO.TAT

Tabulka 18: BWT - 2. krok (setřídění)

Výstupem takto setříděné tabulky je sloupec s posledním symbolem řetězce. V našem případě P11. V literatuře bývá tento sloupec označován symbolem „*L*“ z anglického last. Tento sloupec představuje transformovaný řetězce vhodný pro kompresi. Z původního řetězce „*TENTO.TATA%*“ jsme tedy transformací získali řetězec „*OTTTET.A%NA*“. Všimněme si, že symbol „*T*“ se nyní v řetězci vyskytuje 3x vedle sebe.

Pro zpětnou transformaci budeme vycházet z našeho řetězce, který jsme získali jako výstup procesu transformace, tedy „*OTTTET.A%NA*“. Vytvoříme novou tabulku transpozicí tohoto řetězce. Vzniklý sloupec pojmenujeme „*Klíč*“. Kopií klíče vytvoříme druhý atribut tabulky, který vložíme na pravou stranu od klíče. Tento atribut lexikograficky seřadíme a nazveme jej „*Seřazený řetězec*“. Následně vytvoříme třetí atribut tabulky „*Nový řetězec*“, který vznikne spojením hodnoty atributů „*Klíč*“ a „*Seřazený řetězec*“ v daném řádku tabulky.

Zde jen poznamenejme k atributu „*Klíč*“, že v průběhu celé zpětné transformace se nijak neřadí a používá se vždy v nezměněné podobě (jeho řádky odpovídají vždy řetězci „*OTTTET.A%NA*“). A vždy je pozičně *před* (nalevo) od atributu „*Seřazený řetězec“* – nový řetězec narůstá o hodnotu klíče zleva.

Klíč	Seřazený řetězec	Nový řetězec
O	.	O.
T	A	TA
T	A	TA
T	E	TE
E	N	EN
T	O	TO
.	T	.T
A	T	AT
%	T	%T
N	T	NZ
A	%	A%

Tabulka 19: BWT – zpětná transformace – krok 1

Následující kroky budeme provádět cyklicky:

(1) Vytvoříme prázdnou tabulku identické struktury jako v předchozím kroku.

(2) Naplníme atribut klíč identickými hodnotami (ten se v průběhu procesu nemění a slouží jako základ pro rozšiřování řetězce).

(3) Nový řetězec z předchozího kroku vložíme do sloupce „*Seřazený řetězec*“ a opět jej lexikograficky setřídíme (pouze tento sloupec).

(4) Symboly z atributů „*Klíč*“ a „*Seřazený řetězec*“ spojíme, čím získáme hodnotu pro atribut „*Nový řetězec*“. S každou iterací narůstá atribut „*Nový řetězec*“ o jeden symbol.

(5) Cyklicky opakujeme, dokud atribut „*Nový řetězec*“ nebude mít stejný počet znaků, jaký odpovídá počtu řádků (symbolů) v atributu klíč.

Následující tabulka ukazuje, jak narůstá atribut „*Nový řetězec*“ během jednotlivých iterací:

Iterace 10	Iterace 9	Iterace 8	Iterace 7	Iterace 6	Iterace 5	Iterace 4	Iterace 3	Iterace 2	Iterace 1
O.TATA%TENT	O.TATA%TEN	O.TATA%TE	O.TATA%T	O.TATA%	O.TATA	O.TAT	O.TA	O.T	O.
TATA%TENTO.	TATA%TENTO	TATA%TENT	TATA%TEN	TATA%TE	TATA%T	TATA%	TATA	TAT	TA
TA%TENTO.TA	TA%TENTO.T	TA%TENTO.	TA%TENTO	TA%TENT	TA%TEN	TA%TE	TA%T	TA%	TA
TENTO.TATA%	TENTO.TATA	TENTO.TAT	TENTO.TA	TENTO.T	TENTO.	TENTO	TENT	TEN	TE
ENTO.TATA%T	ENTO.TATA%	ENTO.TATA	ENTO.TAT	ENTO.TA	ENTO.T	ENTO.	ENTO	ENT	EN
TO.TATA%TEN	TO.TATA%TE	TO.TATA%T	TO.TATA%	TO.TATA	TO.TAT	TO.TA	TO.T	TO.	TO
.TATA%TENTO	.TATA%TENT	.TATA%TEN	.TATA%TE	.TATA%T	.TATA%	.TATA	.TAT	.TA	.T
ATA%TENTO.T	ATA%TENTO.	ATA%TENTO	ATA%TENT	ATA%TEN	ATA%TE	ATA%T	ATA%	ATA	AT
%TENTO.TATA	%TENTO.TAT	%TENTO.TA	%TENTO.T	%TENTO.	%TENTO	%TENT	%TEN	%TE	%T
NTO.TATA%TE	NTO.TATA%T	NTO.TATA%	NTO.TATA	NTO.TAT	NTO.TA	NTO.T	NTO.	NTO	NT
A%TENTO.TAT	A%TENTO.TA	A%TENTO.T	A%TENTO.	A%TENTO	A%TENT	A%TEN	A%TE	A%T	A%

Tabulka 20: BWT – rekonstrukce původního řetězce

Poté co zkonstruujeme atribut „*Nový řetězec*“ do původní délky, zbývá identifikovat původní řetězec. Je to ten, který má na konci výstupní (EOF) znak, tedy „%“.

4. KOMPRESE V PRAKTICKÉM UŽITÍ

Předpokládejme, že uživatel disponuje teoretickými znalostmi ke kompresním algoritmům a ví přesně, který algoritmus chce využít. Cesta k jeho reálné aplikaci však nemusí být vždy snadná. Existuje několik možností, které uživatel má. První možnost, která se nabízí a která je nejnáročnější, je vlastní implementace algoritmu do zvoleného programovacího jazyka (tím může být například JAVA, C, C++, Python, R). Druhou možností je využití knihoven/tříd některého programovacího jazyka – tedy již implementovaných procedur a funkcí, které daný jazyk nabízí. Zde ovšem vyvstává riziko, že daný algoritmus je nějakým způsobem modifikovaný, proto je důležitá dokumentace či přístup ke kódu samotnému. A konečně třetí možností je využití kompresních/archivačních programů dostupných volně či za úplatu. V této části práce se proto zaměříme na tyto programy a na možnosti, které uživateli nabízejí. Vybereme a popíšeme tři zástupce nejpoužívanějších, resp. nejstahovanějších kompresních programů v českém prostředí. Pro úplnost jen dodejme, že pojmem program máme na mysli konkrétní „balíček“ softwaru, který je nabízen zdarma či prodáván konkrétním výrobcem na trhu a který podléhá licenčním ujednáním.

4.1 Cíle praktické části

V první polovině této práce jsme se zaměřili na výběr několika kompresních algoritmů a popsali jsme princip na jehož základě fungují. V další části tohoto textu se zaměříme na konkrétní software a na kompresní techniky, které využívá. Cílem této praktické části je popsat tři vybrané zástupce. Zaměříme se na to, jaké algoritmy implementují, jakým způsobem je označují a porovnáme jejich kompresní poměr (vyhodnotíme tedy efektivitu komprese).

Jsme si vědomi faktu, že na problematiku komprese lze nahlížet mnoha způsoby, zejména v závislosti na povaze a účelu dat určených ke kompresi. Tato práce si neklade za cíl podat úplný výčet dostupného software a ani jej detailně analyzovat pomocí všech metrik uvedených v kapitole 2.3, ale popsat je na základě zvolených kritérií.

4.2 Kompresní a archivační programy

Pojem kompresní a archivační program bývá mezi uživateli často vnímán jako synonymum. Přesto je však mezi nimi rozdíl. Kompresní programy implementují výše uvedené algoritmy, a umožňují tak zmenšení původního souboru, tedy kompresi. Oproti tomu archivační programy představují jakousi nadstavbu ke kompresním programům. Kromě vlastní komprese umožňují i vytváření tzv. archivů. Tedy kolekcí již komprimovaných souborů a jejich správu, mezi kterou patří například přidávání či odebírání souborů, práce se stromovou strukturou a podobně (Morkes 1998, s. 83–85). Většina programových balíčků, které uživatelé využívají pro kompresi dat, je tedy současně kompresním a archivačním programem. Tyto dvě vlastnosti nelze oddělit.

Podle serveru www.stahuj.cz (STAHUJ.CZ 2020) a www.slunecnice.cz (SLUNECNICE.CZ 2020) patří v tuzemsku mezi trojici nejstahovanějšího softwaru[16] programy WinRar, 7-Zip a WinZip, na které se zaměříme v této části.

	stahuj.cz	slunecnice.cz
WinRar	9.457.177	7.029.189
7-Zip	1.829.248	1.564.906
WinZip	971.456	1.463.785

Tabulka 21: Kompresní programy a počty stažení v ČR

4.3 Metody hodnocení a volba testovacích dat

Abychom mohli provézt praktická měření vzhledem k cílům, které jsme si stanovili, je třeba vydefinovat jednotlivé ukazatele, které budeme v této práci sledovat. Zaměříme se zejména na tyto ukazatele:

(1) V oblasti přístupu přes uživatelské rozhraní programu nebo možnosti zpracování dat příkazovou řádkou, budeme hodnotit, zda si uživatel může zvolit konkrétní algoritmus komprese, nebo zda tuto volbu učiní program za něho. Také zhodnotíme, zda se program drží jmenné konvence algoritmů popsaných výše, nebo zda zavádí vlastní terminologii.

(2) Provedeme kompresi pro všechny algoritmy daného programu a vypočteme kompresní poměr (K_P). Tím vyhodnotíme úspěšnost každého algoritmu, který program nabízí. Vyplyne-li z testu, že zkoumané programy využívají stejné či obdobné algoritmy, pak porovnáme jejich kompresní poměr mezi sebou. Všechny naměřené kompresní poměry pak vyhodnotíme navzájem mezi jednotlivými programy.

(3) Zhodnotíme, zda existuje dokumentace k danému softwaru, a to formou nápovědy uvnitř programu nebo na webových stránkách poskytovatele programu a zaměříme se na to, zda blíže identifikuje jednotlivé algoritmy používané programem (zejména tam, kde poskytovatel zavádí vlastní pojmenování algoritmů).

Jak jsme již uvedli výše, významný vliv na výsledek komprese má charakter dat a účel jejich využití. Avšak vzhledem k cíli této práce, kde kompresní poměr je pouze jedním ze sledovaných ukazatelů, nepředstavuje volba testovacích dat zásadní problém. Proto pro použité měření zvolíme pouze jeden testovací soubor, který bude obsahovat jak data numerická, tak data textová (kódy, popisky, názvy) s dostatečným počtem hodnot. Využijeme k tomu soubor z archivu tzv. „*open data*“ publikovaná Statním ústavem pro kontrolu léčiv, konkrétně z kategorie „*Seznam cen a úhrad LP/PZLÚ k 1.1.2021*“

[16] Uvedené počty stažení vztahují k 31.5.2020.

ke stažení pod názvem „*SCAU210101v14.txt*“[17]. Zvolený soubor má velikost 3,35 MB a obsahuje 9274 řádků (přibližně 3,5 milionu znaků). Pro jednodušší práci s tímto souborem jej přejmenujeme na „*s.txt*“.

4.4 Vybrané kompresní a archivační programy a jejich analýza na testovaném vzorku dat

Pro úplnost je třeba uvézt, že níže uvedené testy provádíme na běžném kancelářském notebooku s operačním systémem Windows 11. Hardwarové vybavení v tomto případě nehraje roli, neboť v této práci nehodnotíme rychlost komprese/dekomprese a dále pracujeme s relativně malými soubory. V profesionálně budovaném archivu velké instituce by se však jednalo o významné proměnné, které by bylo třeba zohlednit. Přehled použitých verzí programů demonstruje tabulka níže.

Program	Verze
WinRar	6.11 (64bitová verze)
7-Zip	21.07 alpha (x64)
WinZip	27.0

Tabulka 22: Použité verze programů

V následujících podkapitolách tyto programy detailně představíme a provedeme měření na našem testovacím vzorku dat. Jelikož je každý z programů odlišný, budeme pro kompresi dat využívat dvě základní volby – tedy volbu typu archivu a volbu metody programu. Zde pro úplnost dodejme, že většina testovaných programů pracuje s pojmem „*metoda*“ v rámci svého kontextového menu. V kapitole 2.1 jsme pojednali o rozdílu mezi metodou a algoritmem a ukázali jsme, že v literatuře bývají oba pojmy zaměňovány. Každý z programů WinRar, 7-Zip a WinZip ale používá pojem „*metoda*“ specifickým způsobem, a to konkrétně ve svém uživatelském rozhraní. Tam slouží k označení kontextové nabídky, která uživateli umožňuje zvolit to, co bychom obecně mohli nazvat typem komprese (z pohledu daného programu). Tím pak může být skutečný algoritmus (například LZMA), anebo autorem programu definovaná metoda typu „*Fast, Ultra*“ či „*Nejrychlejší*“, jak o tom bude detailněji pojednáno níže. Jednotlivé programy se tedy neshodují v terminologii. A protože s pojmem „*metoda*“ níže pracujeme, je třeba zdůraznit, že je vždy užit v kontextu konkrétního programu, o kterém právě pojednáváme.

Uvědomujeme si, že programy nabízejí další možnosti, jak proces komprese ovlivnit, například volba velikosti slovníku, velikost bloku či velikost slova. Tyto možnosti však nevyužijeme případně ponecháme na hodnotě, kterou sám program nabídl.

4.4.1 7-Zip

Tento program je vyvíjen Igorem Pavlovem a je šířen pod licencí GNU LGPL. Stejně jako například WinRar umožňuje ovládání skrze uživatelské rozhraní, nebo pomocí příkazového řádku. K dispozici je

[17] Dostupné z: https://www.sukl.cz/file/94824_1_1

detailní webová dokumentace (7-Zip 2019). Níže uvedený text pracuje s českou verzí programu, konkrétně 21.07 alpha (x64).

Uživatelské rozhraní programu 7-Zip nabízí několik rozbalovacích seznamů, jejichž obsah je aktualizován na základě programem povolených kombinací, které uživatel zvolí. Jinak řečeno formát archivu určuje typ komprimační metody. Pro potřeby této práce jsou relevantní dva z nich, konkrétně „*Formát archivu*“ a „*Komprimační metoda*“.

Formát Archivu	Komprimační metoda
7z	LZMA, LZMA2, PPMd, BZip2
bzip2	bzip2
gzip	Deflate
Tar	GNU, POSIX
wim	---
xz	LZMA2
zip	Deflate, Deflate64, BZip2, LZMA, PPMd

Tabulka 23: Povolené metody komprese v programu 7-Zip

Mimo samotného uživatelského rozhraní umožňuje program ovládání skrze příkazovou řádku, která nabízí další možnosti, pomocí nichž je možné kompresi ovlivnit. Podrobnější analýza této problematiky by však přesahovala rámec této práce, a proto využijeme pouze základní příkazy programu. Ty parametry, které neuvedeme, software nahradí tzv. „*default hodnotou*“[18]. V programu 7-Zip lze základní syntaxi kódu vyjádřit takto:

<název_příkazu> <název_přepínače> <název_a_cesta_k_novému_archivu>.<typ_archivu>
<název_a_cesta_ke_komprimovanému_souboru>

Stěžejním příkazem, který využijeme pro kompresi zvoleného souboru bude příkaz „*a*“ (add) a přepínač „*-mm*“, pomocí něhož určíme typ metody komprese. Podle tabulky 23 tedy připravíme sadu příkazů, které nám vytvoří příslušné soubory.

Začneme metodou LZMA pro typ archivu 7z a sestavíme příkaz:

```
7z a -mm=LZMA c:\Komprese\Vystup\7Z_LZMA.7z c:\Komprese\s.txt
```

Pro výstupní soubor použijeme jmennou konvenci <formát archivu>_<komprimační metoda>.

[18] Jedná se o výrobcem/programátorem určenou hodnotu parametru, která se použije v případě, že v příkazu není explicitně uvedena.

Po spuštění a provedení obdržíme informaci o správnosti provedené komprese:

```
7-Zip 19.01 alpha (x64) : Copyright (c) 1999-2019 Igor Pavlov : 2019-09-05

Scanning the drive:
1 file, 3510638 bytes (3429 KiB)

Creating archive: c:\Komprese\Vystup\7Z_LZMA.7z

Add new data to archive: 1 file, 3510638 bytes (3429 KiB)

Files read from disk: 1
Archive size: 388511 bytes (380 KiB)
Everything is Ok
```

Analogicky postupujeme s dalšími typy jednotlivých archivů a metod komprese a spustíme následující sekvenci příkazů:

```
7z a -mm=LZMA2 c:\Komprese\Vystup\LZMA2.7z c:\Komprese\s.txt
7z a -mm=PPMd c:\Komprese\Vystup\PPMD.7z c:\Komprese\s.txt
7z a -mm=BZip2 c:\Komprese\Vystup\BZIP2.7z c:\Komprese\s.txt
7z a -mm=BZip2 c:\Komprese\Vystup\BZIP2.BZIP2 c:\Komprese\s.txt
7z a -mm=Deflate c:\Komprese\Vystup\DEFLATE.gzip c:\Komprese\s.txt
7z a c:\Komprese\Vystup\TAR.tar c:\Komprese\s.txt
7z a c:\Komprese\Vystup\WIM.wim c:\Komprese\s.txt
7z a -mm=LZMA2 c:\Komprese\Vystup\LZMA2.xz c:\Komprese\s.txt
7z a -mm=Deflate c:\Komprese\Vystup\DEFLATE.zip c:\Komprese\s.txt
7z a -mm=Deflate64 c:\Komprese\Vystup\DEFLATE64.zip c:\Komprese\s.txt
7z a -mm=BZip2 c:\Komprese\Vystup\BZIP2.zip c:\Komprese\s.txt
7z a -mm=LZMA c:\Komprese\Vystup\LZMA.zip c:\Komprese\s.txt
7z a -mm=PPMd c:\Komprese\Vystup\PPMD.zip c:\Komprese\s.txt
```

Následující tabulka představuje velikost výstupních souborů v závislosti na zvolené metodě komprese a formátu archivu. Hodnoty jsou uváděny v bajtech. Tři čárky znamenají, že pro daný formát archivu není komprimační metoda podporována/povolena.

Komprimační metoda	Formát archivu						
	7z	bzip2	gzip	tar	wim	xz	zip
LZMA	388 511	---	---	---	---	---	388 556
LZMA2	388 559	---	---	---	---	388 504	---
PPMd	456 615	---	---	---	---	---	406 682
BZip2	452 479	452 365	---	---	---	---	452 509
Deflate	---	---	525 395	---	---	---	525 515
Deflate64	---	---	---	---	---	---	511 436
Nespecifikováno	---	---	---	3 512 320	3 511 904	---	---

Tabulka 24: Velikost komprimovaných souborů v závislosti na metodě (7-Zip)

Pro takto získaná data vypočteme kompresní poměr:

Komprimační metoda	Formát archivu						
	7z	bzip2	gzip	tar	wim	xz	zip
LZMA	0,11067	---	---	---	---	---	0,11068
LZMA2	0,11068	---	---	---	---	0,11066	---
PPMd	0,13007	---	---	---	---	---	0,11584
BZip2	0,12889	0,12886	---	---	---	---	0,12890
Deflate	---	---	0,14966	---	---	---	0,14969
Deflate64	---	---	---	---	---	---	0,14568
Nespecifikováno	---	---	---	1,00048	1,00036	---	---

Tabulka 25: Vypočtený kompresní poměr (7-Zip)

Z vypočteného kompresního poměru je patrné, že pro náš referenční soubor byla nejefektivnější kompresní metoda „*LZMA*2“ v archivu „*xz*“. Naopak jak ukazuje kompresní poměr pro archivy „*tar*“ a „*wim*“ s nespecifikovanou metodou (ze strany programu), je hodnota větší než 1. Došlo tedy ke zvětšení komprimovaného souboru oproti původnímu souboru. Tento typ by tedy nebyl vhodný pro kompresi.

4.4.2 WinRar

Program WinRar od společnosti RARLAB patří mezi kompresní archivační programy, ale nabízí množství funkcí navíc, jako například šifrování souboru. Jedná se o komerční software chráněný autorským právem, který je nabízen jako tzv. shareware, tedy na omezenou dobu,[19] po kterou umožnuje jeho plné využití. Poté je nutné si zakoupit licenci. Sama společnost na svých webových stránkách uvádí, že její software využívá více než 500 milionů uživatelů (WinRar 2020). V této podkapitole budeme pracovat s poslední publikovanou verzí programu, tj. verzí 6.11 (64bitová verze) (lokalizovanou pro české prostředí).

Uživatelské rozhraní programu v základním nastavení umožňuje zvolit formát archivu („*RAR*“, „*RAR4*“ a „*ZIP*“), dále pak kompresní metodu („*bez komprese*“, „*nejrychlejší*“, „*rychlá*“, „*normální*“, „*dobrá*“, „*nejlepší*“) a velikost slovníku (1 – 1024 MB). Je tedy zřejmé, že jednotlivé kompresní algoritmy jsou implicitně nastaveny do předem určených skupin a uživatel nemá možnost je ovlivnit. Z dokumentace pak není ani patrné, zda tato možnost je či není zpřístupněna alespoň skrze příkazovou řádku. Pro úplnost je třeba dodat, že v sekci „*Poděkování*“ v nápovědě k programu autoři explicitně zmiňují pouze algoritmy „*PPDM*“ a „*LZMA*“, z čehož lze dovodit jejich užití v programu.

[19] V případě WinRar 40 dní.

Obecná syntaxe kompresního příkazu v programu WinRar je:

<název_příkazu> -<název_přepínače> <název_a_cesta_k_novému_souboru>
<název_a_cesta_ke_komprimovanému_souboru>.

Obdobně jako u předchozího programu 7-Zip je základním kompresním příkazem „*a*“ (add). Přepínač, kterým volíme kompresní metodu, je označen písmenem „*m*“ a číslicí 0-5, která odpovídá danému typu komprese (0 = bez komprese; 1 = nejrychlejší; 2 = rychlá; 3 = normální; 4 = dobrá; a 5 = nejlepší).

Dle výše uvedeného tedy připravíme sadu příkazů a výstupní soubory pojmenujeme podle přepínače, který na ně byl aplikován:

```
rar a -m0 c:\Komprese\Vystup\m0.rar c:\Komprese\s.txt
rar a -m1 c:\Komprese\Vystup\m1.rar c:\Komprese\s.txt
rar a -m2 c:\Komprese\Vystup\m2.rar c:\Komprese\s.txt
rar a -m3 c:\Komprese\Vystup\m3.rar c:\Komprese\s.txt
rar a -m4 c:\Komprese\Vystup\m4.rar c:\Komprese\s.txt
rar a -m5 c:\Komprese\Vystup\m5.rar c:\Komprese\s.txt
rar a -m0 c:\Komprese\Vystup\m0.rar4 c:\Komprese\s.txt
rar a -m1 c:\Komprese\Vystup\m1.rar4 c:\Komprese\s.txt
rar a -m2 c:\Komprese\Vystup\m2.rar4 c:\Komprese\s.txt
rar a -m3 c:\Komprese\Vystup\m3.rar4 c:\Komprese\s.txt
rar a -m4 c:\Komprese\Vystup\m4.rar4 c:\Komprese\s.txt
rar a -m5 c:\Komprese\Vystup\m5.rar4 c:\Komprese\s.txt
rar a -m0 c:\Komprese\Vystup\m0.zip c:\Komprese\s.txt
rar a -m1 c:\Komprese\Vystup\m1.zip c:\Komprese\s.txt
rar a -m2 c:\Komprese\Vystup\m2.zip c:\Komprese\s.txt
rar a -m3 c:\Komprese\Vystup\m3.zip c:\Komprese\s.txt
rar a -m4 c:\Komprese\Vystup\m4.zip c:\Komprese\s.txt
rar a -m5 c:\Komprese\Vystup\m5.zip c:\Komprese\s.txt
```

Níže uvedená tabulka představuje velikost komprimovaných souborů v bajtech. Jak je patrné z výstupu, tak volba formátu archivu u programu WinRar nemá vliv na velikost souboru. Program tedy aplikuje na různé formáty vždy stejnou komprimační metodu se stejným výsledkem velikosti souboru.

Komprimační metoda	Formát archivu		
	RAR	RAR4	ZIP
Bez komprese	510 807	510 807	510 807
Nejrychlejší	531 202	531 202	531 202
Rychlá	462 730	462 730	462 730
Normální	443 720	443 720	443 720
Dobrá	438 261	438 261	438 261
Nejlepší	436 347	436 347	436 347

Tabulka 26: Velikost komprimovaných souborů v závislosti na metodě (WinRar)

Z tabulky je patrné, že nejefektivnější metodou komprese, co se velikosti souboru týče, byla volba „*Nejlepší*“, a to pro všechny typy archivů, které program nabízí.

Na základě naměřených hodnot provedeme výpočet kompresního poměru:

Komprimační metoda	Formát archivu		
	RAR	RAR4	ZIP
Bez komprese	0,14550	0,14550	0,14550
Nejrychlejší	0,15131	0,15131	0,15131
Rychlá	0,13181	0,13181	0,13181
Normální	0,12639	0,12639	0,12639
Dobrá	0,12484	0,12484	0,12484
Nejlepší	0,12429	0,12429	0,12429

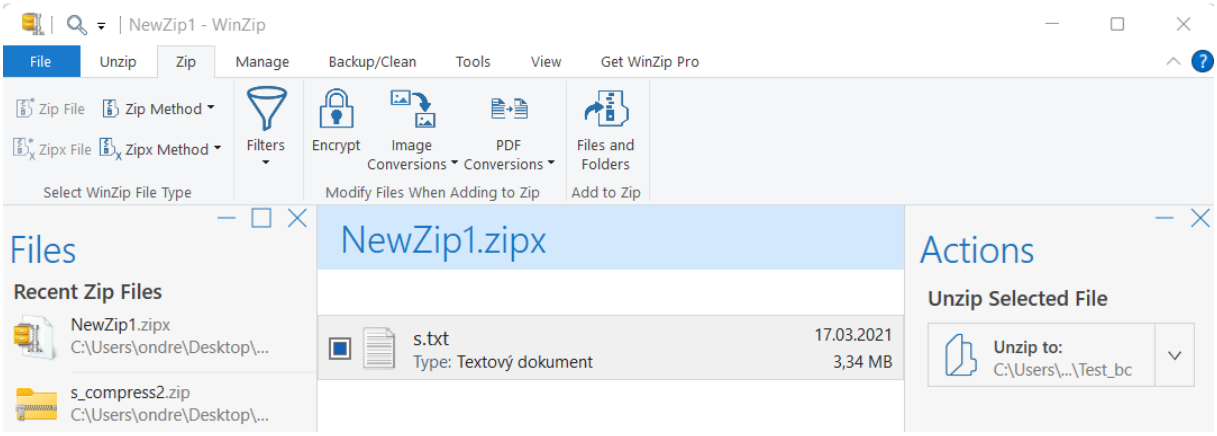
Tabulka 27: Vypočtený kompresní poměr (WinRar)

Nejnižšího kompresního poměru dosáhla metoda „*Nejlepší*“, a to s hodnotou 0,12429.

4.4.3 WinZip

Kompresní program WinZip byl poprvé představen v roce 1991 společností Nico Mak Computing. Později změnil svého majitele a aktuálně jej vlastní a vyvíjí společnost Corel Corporation. Jedná se o komerční produkt nabízený za úplatu (WinZip, 2020). V této práci využijeme verzi 27.0.

Jak vyplývá z rozhraní programu, WinZip nabízí možnost vytvořit dva typy archivů s použitím vytvoření souboru „*Zip*“, nebo „*Zipx*“. První možnost nabízí celkem 4 volby typu metody („*No compression*“, „*Super fast*“, “ *Enhanced Deflate*“ a *“Maximum“*). Tyto typy nabízí i druhá z metod, a dále ještě šest dalších („*Best method*“, „*Bzip*“, „*LZMA*“, „*PPMd*“, „*XZ*“ a „*Zstd*“). Jak je patrné, tak z metody Zipx lze alespoň částečně dovodit, s jakými algoritmy pracuje. Je třeba zdůraznit, že oproti programům WinRar a 7-Zip je rozhraní poměrně neintuitivní. Je založené na klasických záložkách známých z vyšších verzí kancelářského balíku Microsoft Office. K samotné kompresi dochází tak, že načteme či přetáhneme soubor, který má být komprimován, na záložce „*Zip*“ zvolíme metodu a poté přejdeme do záložky „*File*“, kde soubor standardní cestou uložíme a tím dojde ke kompresi souboru.



Obrázek 4: Rozhraní programu WinZip

Co se týče práce s příkazovou řádkou, je třeba podotknout, že na rozdíl od programů 7-Zip a WinRar neumožňuje WinZip její ovládání napřímo, protože příkazová řádka není součástí programu. Je nutné si

stáhnout a nainstalovat další produkt s názvem WinZip Command Line,[20] který je taktéž zpoplatněn. V tomto případě využíváme jeho trial verzi.[21]

Obecná syntaxe kompresního příkazu je v programu WinZip:

<název_příkazu> -<název_přepínače> <název_a_cesta_k_novému_souboru>
<název_a_cesta_ke_komprimovanému_souboru>.

Základním příkazem je „*-a“*, pomocí kterého vytvoříme komprimovaný soubor. Pro jeho dekompresi pak slouží příkaz „*-e“*. Dále existují ještě další příkazy, pomocí kterých lze soubory například modifikovat, aktualizovat a podobně. Výběrem vhodného přepínače pak volíme konkrétní kompresní metodu. Seznam těchto přepínačů ilustruje následující tabulka.

Přepínač	Metoda
-ep	PPMd
-el	LZMA
-eb	bzip
-ee	enhanced deflate
-ex	maximum
-es	super fast
-ez	best method
-et	ZStd
-eq	XZ
-e0	no compression

Tabulka 28: Seznam přepínačů programu WinZip

Sestavíme a spustíme základní kompresní příkazy:

```
wzzip -a -ex "c:\Komprese\Vystup\maximum.zip" "c:\Komprese\s.txt"
wzzip -a -e0 "c:\Komprese\Vystup\no_compression.zip" "c:\Komprese\s.txt"
wzzip -a -ee "c:\Komprese\Vystup\enhanced_deflate.zip" "c:\Komprese\s.txt"
wzzip -a -es "c:\Komprese\Vystup\super_fast.zip" "c:\Komprese\s.txt"
wzzip -a -ez "c:\Komprese\Vystup\best_method.zipx" "c:\Komprese\s.txt"
wzzip -a -eb "c:\Komprese\Vystup\bzip.zipx" "c:\Komprese\s.txt"
wzzip -a -el "c:\Komprese\Vystup\lzma.zipx" "c:\Komprese\s.txt"
wzzip -a -ep "c:\Komprese\Vystup\ppmd.zipx" "c:\Komprese\s.txt"
wzzip -a -eq "c:\Komprese\Vystup\xz.zipx" "c:\Komprese\s.txt"
wzzip -a -et "c:\Komprese\Vystup\zstd.zipx" "c:\Komprese\s.txt"
wzzip -a -ex "c:\Komprese\Vystup\maximum.zipx" "c:\Komprese\s.txt"
wzzip -a -e0 "c:\Komprese\Vystup\no_compression.zipx" "c:\Komprese\s.txt"
wzzip -a -ee "c:\Komprese\Vystup\enhanced_deflate.zipx" "c:\Komprese\s.txt"
wzzip -a -es "c:\Komprese\Vystup\super_fast.zipx" "c:\Komprese\s.txt"
```

[20] Program je od stejné společnosti jako samotný WinZip.
[21] Konkrétně WinZip(R) Command Line Support Add-On Version 6.

Výstup naměřených hodnot reprezentuje tato tabulka:

Komprimační metoda	Formát archivu	
	zip	zipx
No compression	3 510 746	3 510 746
Super fast	651 403	651 403
Enhanced Deflate	508 565	508 565
Maximum	521 905	521 905
Best method	---	393 794
BZip	---	452 461
LZMA	---	368 126
PPMd	---	364 743
XZ	---	368 364
Zstd	---	458 444

Tabulka 29: Velikost komprimovaných souborů v závislosti na metodě (WinZip)

Je patrné že nejmenší velikosti komprimovaného souboru dosáhla metoda „*PPMd*“ archivu „*zipx*“. I v tomto případě je zřejmé, že volba typu archivu nemá vliv na velikost komprimovaného souboru, jak vyplývá z tabulky. Pro naměřená data opět vypočteme kompresní poměr:

Komprimační metoda	Formát archivu	
	zip	zipx
No compression	1	1
Super fast	0,18555	0,18555
Enhanced Deflate	0,14486	0,14486
Maximum	0,14866	0,14866
Best method	---	0,11217
BZip	---	0,12888
LZMA	---	0,10486
PPMd	---	0,10390
XZ	---	0,10493
Zstd	---	0,13059

Tabulka 30: Vypočtený kompresní poměr (WinZip)

Nejlepšího kompresního poměru jsme tedy logicky dosáhli právě u metody „*PPMd*“, a to hodnoty 0,10390.

4.5 Vyhodnocení metod jednotlivých programů a jejich efektivity komprese

4.5.1 Uživatelské rozhraní a příkazová řádka

Jak bylo řečeno výše, WinRar, 7-Zip a WinZip jsou archivačními programy. Jejich primárním účelem je tvorba archivů s využitím kompresních algoritmů.[22] Rozhraní aplikace tak představuje vstupní bránu

[22] Byť každý z nich umožňuje i vytvoření archivu bez komprese.

programu. Lze tedy od něho očekávat, že uživateli umožní minimálně: (1) zvolit typ archivu, který daný program podporuje, (2) zvolit kompresní metodu, se kterou soubor zpracuje a (3) poskytne intuitivní prostředí a ovládání. Níže uvedená tabulka shrnuje typy archivů a kompresních metod, které analyzované programy nabízejí. Dále pak přibližuje jmennou konvenci použitou autory softwaru.

Název programu	Název atributu [CZ]	Název atributu [EN]	Hodnoty atributu [EN]
WinRar	Formát archivu	Archive format	RAR, RAR4, ZIP
	Kompresní metoda	Compression method	Store, Fastest, Fast, Normal, Good, Best
7-Zip	Formát archivu	Archive format	7z, bzip2, gzip, tar, wim, xz, zip
	Komprimační metoda	Compression method	LZMA2, LZMA, PPMd, BZip2, * Deflate, * GNU, POSIX, Deflate64
WinZip	---	Zip Method	Maximum, Enhanced Deflate, Super fast, No compression
	---	Zipx Method	Best method, Maximum, Enhanced Deflate, Super fast, No compression, BZip, LZMA, PPMd, XZ, Zstd

Tabulka 31: Přehled metod a atributů programů 7-Zip, WinRar a WinZip

V případě možnosti volby typu archivu používají programy WinRar a 7-Zip označení „*Formát archivu*“ či anglické „*Archive format*“. Využívají tedy tento pojem analogicky. Oproti tomu program WinZip tento pojem nezavádí a uživateli nabízí pouze možnost volby mezi „*Zip File*“ či „*Zipx File*“. Samotné typy archivů, které každý z programů umožňuje vytvářet, závisí na konkrétní specializaci daného softwaru. Průnikem mezi těmito třemi programy je archiv typu zip v různých jeho verzích.

Komparace volby metody je komplikovanější. Všechny tři testované programy s pojmem metoda pracují. Liší se však v nabízených typech metod a v jejich pojmenování. Opět panuje shoda v označení atributu u programů WinRar a 7-Zip, alespoň v anglickém jazyce, tedy „*Compression method*“. Česká verze programů pracuje s pojmy „*Kompresní metoda*“ a „*Komprimační metoda*“, což jen dokládá, že v českém jazyce jsou tyto pojmy vnímány jako synonyma. Pokud jde o konkrétní typy nabízených metod, setkáváme se se dvěma druhy označení, a to na základě jejich vlastností[23] (např. „*good*“, „*best*“, „*fast*“) nebo podle kompresního algoritmu na jehož základě byly vytvořeny. První z přístupů je výhodný pro běžného uživatele, neboť nevyžaduje znalost problematiky komprese a jejích algoritmů. Nevýhodou však je, že tyto názvy nejsou nijak standardizované. Vyvstává tak legitimní otázka, zda například metoda „*Fast*“ programu WinRar je obdobná metodě „*Super fast*“ WinZipu. A pokud ano, pak kvantitativně, či kvalitativně? Jelikož však toto není předmětem našeho zkoumání, nebudeme se tím v této práci zabývat.

Příkazová řádka představuje pro kompresi dat přímočařejší přístup. Všechny tři výše uvedené programy tuto možnost nabízejí. Programy WinRar a 7-Zip ji nabízejí přímo, jako součást instalace. V případě programu WinZip se jedná o doplněk, který je nutné nainstalovat zvlášť.

[23] Ve vztahu k velikosti komprimovaného souboru nebo rychlosti procesu komprese.

U zkoumaných programů je samotná syntaxe kódu obdobná. Obecně je možno ji shrnout jako *příkaz-přepínač-původní_soubor-komprimovaný soubor*. Příkaz je instrukce programu, jaká operace má být vykonána. V našem případě jsme využili základního příkazu pro kompresi, který je shodně ve všech programech označen jako „*a*“, což patrně odkazuje k anglickému „*add*“ – přidat. Pomocí přepínače pak definujeme konkrétní metodu komprese. Programy WinRar a WinZip využívají pro označení konkrétní metody definovaný dvojmístný kód. Avšak program 7-Zip metodu definuje jedním přepínačem („*-mm*“) s využitím symbolu = (například „*-mm=LZMA2*“).

Porovnání výhod či nevýhod jednoho nebo druhého přístupu reprezentuje níže uvedená tabulka.

Uživatelské rozhraní	Příkazový řádek
Nevyžaduje znalost konkrétních kompresních algoritmů.	Nevyžaduje znalost konkrétních kompresních algoritmů.
Vždy součástí instalace.	V jednom případě nutná instalace dodatečného programu, jinak součást programu.
Nevyžaduje znalost programování.	Vyžaduje alespoň elementární obecnou znalost programování.
Nevyžaduje znalost syntaxe daného jazyka.	Nutnost seznámit se se syntaxí daného jazyka.

Tabulka 32: Výhody a nevýhody uživatelského rozhraní a příkazového řádku

Jak je patrné, pro běžné užití softwaru ke kompresi je výhodnější využít uživatelského rozhraní než napsat skript do příkazového řádku. Zpracování dat skrze příkazový řádek má smysl zejména v procesech, či systémech, kde se automaticky ukládá/zpracovává velké množství dat – například v DMS systémech.[24]

Porovnáme-li mezi s sebou způsob komprese skrze příkazovou řádku, tak lze říci, že mezi jednotlivými programy nenajdeme výraznější rozdíl. Naopak v případě uživatelského rozhraní rozdíly nalezneme. Programy WinRar a 7-Zip mají obdobný způsob práce – dialogové okno, kde tzv. „*naklikáme*“ požadovanou operaci a potvrdíme. Oproti tomu WinZip intuitivně nepůsobí a ve své jednoduchosti je v konečném důsledku složitější. Ostatně k provedení komprese skrze uložení souboru na disk jsme byli nuceni nejprve nastudovat v dokumentaci (tutorial).

4.5.2 Kompresní poměr

Ukazatel kompresní poměr představuje velikost komprimovaného souboru ku velikosti původního souboru. Jak bylo řečeno v teoretické části, čím je jeho hodnota nižší, tím bylo dosaženo úspěšnější komprese. Naopak přesáhne-li jeho hodnota číslo jedna, došlo k zvětšení velikosti komprimovaného souboru oproti původnímu. Tedy komprese nebyla úspěšná a pro daný soubor by měl uživatel zvolit jiný typ metody.

[24] Data Management System

Na třech vybraných archivačních programech jsme provedli měření jejich metod. Nyní výsledné hodnoty porovnáme v souhrnné tabulce,[25] a to podle vypočteného kompresního poměru:

Program	Metoda	Archiv	Kompresní poměr	Pořadí
WinZip	PPMd	zipx	0,10390	1
WinZip	LZMA	zipx	0,10486	2
WinZip	XZ	zipx	0,10493	3
7-Zip	LZMA2	xz	0,11066	4
7-Zip	LZMA	7z	0,11067	5
7-Zip	LZMA	zip	0,11068	6
7-Zip	LZMA2	7z	0,11068	7
WinZip	Best method	zipx	0,11217	8
7-Zip	PPMd	zip	0,11584	9
WinRar	Nejlepší	RAR	0,12429	10
WinRar	Nejlepší	RAR4	0,12429	11
WinRar	Nejlepší	ZIP	0,12429	12
WinRar	Dobrá	RAR	0,12484	13
WinRar	Dobrá	RAR4	0,12484	14
WinRar	Dobrá	ZIP	0,12484	15
WinRar	Normální	RAR	0,12639	16
WinRar	Normální	RAR4	0,12639	17
WinRar	Normální	ZIP	0,12639	18
7-Zip	BZip2	bzip2	0,12886	19
WinZip	Bzip	zipx	0,12888	20
7-Zip	BZip2	7z	0,12889	21
7-Zip	BZip2	zip	0,12890	22
7-Zip	PPMd	7z	0,13007	23
WinZip	Zstd	zipx	0,13059	24
WinRar	Rychlá	RAR	0,13181	25
WinRar	Rychlá	RAR4	0,13181	26
WinRar	Rychlá	ZIP	0,13181	27
WinZip	Enhanced Deflate	zip	0,14486	28
WinZip	Enhanced Deflate	zipx	0,14486	29
WinRar	Bez komprese	RAR	0,14550	30
WinRar	Bez komprese	RAR4	0,14550	31
WinRar	Bez komprese	ZIP	0,14550	32
7-Zip	Deflate64	zip	0,14568	33
WinZip	Maximum	zip	0,14866	34
WinZip	Maximum	zipx	0,14866	35
7-Zip	Deflate	gzip	0,14966	36
7-Zip	Deflate	zip	0,14969	37
WinRar	Nejrychlejší	RAR	0,15131	38
WinRar	Nejrychlejší	RAR4	0,15131	39
WinRar	Nejrychlejší	ZIP	0,15131	40

[25] Pro úplnost uveďme, že zápis názvů metod a archivů uvádíme, tak jak je uvedl výrobce v rozhraní daného programu.

Program	Metoda	Archiv	Kompresní poměr	Pořadí
WinZip	Super fast	zip	0,18555	41
WinZip	Super fast	zipx	0,18555	42
WinZip	No compression	zip	1,00000	43
WinZip	No compression	zipx	1,00000	44
7-Zip	Nespecifikováno	wim	1,00036	45
7-Zip	Nespecifikováno	tar	1,00048	46

Tabulka 33: Úplný přehled kompresních metod a vypočteného kompresního poměru

Výstupní tabulka je seřazena vzestupně podle vypočteného kompresního poměru. Jak je patrné, nejefektivnější metodou, co se týče velikosti zkomprimovaného souboru, byl program WinZip s metodou „*PPMd*“ a archivem „*zipx*“. Kompresní poměr dosáhl hodnoty 0,10390. Druhá a třetí pozice patří stejnému programu a opět archivu „*zipx*“. Metoda „*LZMA*“ dosáhla kompresního poměru 0,10486 a metoda „*XZ*“ pak 0,10493. Čtvrtého nejlepšího kompresního poměru dosáhl program 7-Zip s metodou „*LZMA2*“ a archivem typu „*xz*“, a to s hodnotou 0,11066. Třetí z programů – WinRar dosáhl nejlepšího kompresního poměru až na desátém místě, a to s metodou „*Nejlepší*“ a archivem „*RAR*“. Konkrétní kompresním poměr této metody byl 0,12429.

Porovnáme-li tedy jednotlivé programy mezi sebou podle jejich nejefektivnější komprese – viz tabulka níže – vidíme, že neexistuje průnik mezi metodou ani archivem.

Program	Metoda	Archiv	Kompresní poměr	Pořadí
WinZip	PPMd	zipx	0,10390	1
7-Zip	LZMA2	xz	0,11066	4
WinRar	Nejlepší	RAR	0,12429	10

Tabulka 34: Porovnání nejefektivnějších metod komprese mezi jednotlivými programy

Rozdíl kompresních poměrů mezi programy WinZip a WinRar dosahuje absolutní hodnoty 0,02039.

Jak zároveň demonstruje přehledová tabulka, každý z archivačních programů používá vlastní terminologii a značení. Vzniká tak jistá diskrepance v pojmech metoda a archiv.[26] Z tohoto důvodu je porovnání bez znalosti kompletní dokumentace prakticky nemožné. Programy však lze porovnat podle metod/archivů, které výrobci označují popisně, často pomocí přídavného jména. Porovnáme tedy software podle způsobů komprese, který výrobci označují variantami slova „*nejlepší*“.

Zde je třeba podotknout, že program 7-Zip toto značení nevyužívá u atributů „*metoda“* a „*archiv*“, ale definuje je v atributu, jenž označuje „*Compression level*“.[27] Po vybrání dané úrovně nám rozhraní nastaví konkrétní metodu a archiv. V tomto případě jsme zvolili úroveň „*Ultra*“ a program nám automaticky přednastavil metodu „*BZip2*“ a archiv „*bzip2*“.

[26] Srov. Například WinZip a jeho metoda s označením „*XZ*“ vs. 7-Zip a jeho archiv s označením „*xz*“.
[27] Hodnoty, které může uživatel zvolit, jsou „*Fastest“; „Fast“*; *„Normal“*; „*Maximum“* a „*Ultra“*.

Program	Metoda	Archiv	Kompresní poměr	Pořadí
WinZip	Best method	zipx	0,11217	8
WinRar	Nejlepší	RAR	0,12429	10
7-Zip	BZip2 (Ultra)	bzip2	0,12886	19

Tabulka 35: Porovnání metod deklarovaných výrobcem programu jako "nejlepší"

Jak je z tabulky patrné, tak pouze v případě programu WinRar odpovídala jeho metoda „*Nejlepší*“ i skutečně nejefektivnější metodě komprese, kterou jsme naměřili. U programu WinZip byla dle kompresního poměru nejefektivnější metoda „*PPMd*“ s archivem „*zipx*“ a hodnotou kompresního poměru rovnou číslu 0,10390. Při využití jeho metody „*Best method“* bychom získali kompresní poměr roven hodnotě 0,11217, což v rámci tohoto programu činí rozdíl 0,00827. Obdobné je to i v případě programu 7-Zip, kde rozdíl mezi metodou „*LZMA2*“, archivem „*xz*“, kompresním poměrem 0,11066 a druhou metodou „*Ultra*“ s naměřenou hodnotou 0,12886, činí absolutní rozdíl kompresních poměrů 0,0182.

Z měření tedy vyplývá, že vhodnou či nevhodnou kombinací metody a volby archivu lze dosáhnout různě efektivní komprese. Nelze se tedy spoléhat pouze na popisné značení metod/archivů (typu „*nejlepší*“), ale vždy je třeba brát v úvahu konkrétní povahu dat, které jsou komprimovány a raději provézt více typů komprese. I bez znalosti konkrétního algoritmu lze vždy vyzkoušet různé kombinace parametrů, které program nabízí a následně zvolit metodu, která je z našeho hlediska optimální. Náš testovací vzorek dat byl relativně malý a rychlost komprese jsme nebrali v úvahu. V případě větších souborů by však bylo nutné brát do úvahy i tento faktor a nastavit optimum mezi časem potřebným pro kompresi (či dekompresi) a velikostí komprimovaného souboru.

4.5.3 Softwarová dokumentace

Softwarová dokumentace představuje významný zdroj informací, který charakterizuje konkrétní program, způsob jeho využití, instalace, kompatibility, programovém jazyce apod. V obecné rovině existuje mnoho typů dokumentace, které vznikají v průběhu vývoje programu či *ex post*. Z tohoto hlediska by pro účel této práce bylo vhodné využít tzv. „technickou dokumentaci“ k programu. Jelikož vybrané kompresní programy jsou komerčního původu, je rozsah poskytovaných informací k jednotlivým programům značně omezen a jednotlivé implementace kompresního algoritmu není možné získat (zdrojový kód).

Program WinRar nabízí na svých oficiálních webových stránkách[28] základní informace k programu. Spíše než technické specifikace obsahuje informace komerčního charakteru. Základní údaje k programu nalezneme především v sekci „*FAQ*“, která bohužel není nijak obsáhlá, a popisuje spíše rozhraní programu, licenční politiku apod. Mnohem podrobnější informace nabízí program sám, a to ve své sekci

[28] https://www.win-rar.com/

„*Nápověda*“. Zde nalezneme základní informací pro práci s rozhraním programu, ale i velmi užitečné informace o ovládání skrze příkazovou řádku. Informace týkající se jednotlivých metod jsou velmi stručné a bohužel neidentifikují jaké algoritmy používají.

Na webových stránkách programu 7-Zip[29] nalezneme mimo základních informací k programu, také možnost stáhnout si dokumentaci k programu. Ta obsahuje poměrně detailní informace o jednotlivých metodách, jejich popis a specifikaci, ale opět, až na několik výjimek, bez vyjmenování konkrétních algoritmů (vyjma těch pojmenovaných po konkrétní metodě – viz výše). Dále zde nalezneme informace ohledně práce s programem samotným, či příkazovou řádkou. Web dále obsahuje odkaz na fórum, kde mohou uživatelé klást otázky či řešit problémy a chyby v programu 7-Zip. Jednotlivým metodám se v obecné rovině věnuje i samotná nápověda programu.

Webové stránky programu WinZipu[30] obsahují sekci „*Learning center*“, kde nalezneme poměrně přehledně zpracované návody na práci s rozhraním programu včetně příkladů. Dokumentace k programu samotnému a užitých metodách však chybí. Ani nápověda programu není v tomto směru detailnější. Program oproti WinRaru a 7-Zipu nemá záložku s nápovědou v rámci rozhraní, nicméně po stisknutí klávesy F1, nás přesměruje na webovou verzi nápovědy.[31] Zde nalezneme detailnější informace o programu samotném a práci s ním. Také se věnuje jednotlivým typům podporovaných archivů, nicméně algoritmy blíže nespecifikuje.

[29] https://www.7-zip.org/
[30] https://www.winzip.com/
[31] https://kb.winzip.com/help/EN/WZ/

5. ZÁVĚR

V teoretické části práce jsme představili obecnou terminologii užívanou v souvislosti s kompresními algoritmy. Vysvětlili jsme základní členění komprese na „*ztrátovou*“ a „*bezeztrátovou*“. Ukázali jsme že, pro potřeby komprese dat textového charakteru jsou prakticky využitelné pouze metody bezeztrátové. Metody ztrátové komprese lze využít pouze jako podpůrné, například pro ořezání prázdných řádků na konci textu. Dále jsme představili výběr několika algoritmů užívaných ke kompresi, a to jak těch jednoduchých (metoda potlačení nul, RLE) tak poměrně sofistikovaných (Lempel-Ziv-Welsch, Huffmanovo kódování). U každého algoritmu jsme na konkrétním příkladu demonstrovali způsob, jakým funguje. Ukázali jsme, že samotná povaha textových dat má vliv na výsledek komprese, a proto se často před komprimací využívají podpůrné metody, kdy se data transformují do lépe komprimovatelné podoby. Jednu z takovýchto metod jsme detailněji představili, konkrétně Burrows-Wheelerovu transformaci.

V praktické části jsme se zaměřili na konkrétní archivační a kompresní programy dostupné na trhu. Na základě statistik stahování ze serverů poskytujících software jsme zvolili tři konkrétní produkty WinRar, 7-Zip a WinZip. Hledali jsme odpovědi na několik otázek.

První z otázek, kterou jsme si položili bylo, zda uživatel se znalostí konkrétních algoritmů může skrze uživatelské rozhraní nebo příkazovou řádku ovlivnit výběr konkrétního algoritmu popsaného v teoretické části. S touto oblastí souvisí i druhá otázka, tedy zda je terminologie mezi jednotlivými programy jednotná či nikoliv. Jak vyplynulo z analýzy jednotlivých programů, stěžejními pojmy jsou „*kompresní*“ či „*komprimační*“ metoda a „*formát archivu*“. U všech tří programů určuje volba archivu dostupnost konkrétní metody. V případě názvů jednotlivých metod kombinují jejich tvůrci dva přístupy v jejich označování. První přístup vychází z konkrétního algoritmu, na kterém je jeho metoda založena. Druhý přístup je popisný a je založen na efektivitě či času potřebného ke kompresi („*Maximum*“, „*Nejlepší*“, „*Rychlá*“). Konkrétně tedy analýza ukázala, že podle názvu metody může uživatel alespoň částečně poznat konkrétní algoritmus pouze u programů 7-Zip a WinZip (např. „*LZMA*“, „*LZMA2*“). V případě programu WinRar se tuto informaci z názvu nedozví. Podíváme-li se na tuto problematiku skrze užití příkazového řádku, pak ze syntaxe kódu lze vyčíst konkrétní metodu pouze u programu 7-Zip a to v parametru „*mm*“ (např. „*-mm=LZMA*“). WinRar a WinZip používají pouze kódové označení metody, které tuto informaci nenese.

Další ze zkoumaných otázek se týkala efektivity komprese jednotlivých metod, a to na námi zvoleném testovacím souboru. Konkrétně jsme vypočetli kompresní poměr, tedy jednu z možných metrik. Celkem bylo provedeno 46 kompresí zvoleného testovacího souboru. Jak vyplývá z výsledků v kapitole 4.5.2, pak nejlepšího kompresního poměru dosáhla metoda „*PPMd*“ programu WinZip, a to s hodnotou 0,10390. Naopak nejhůře dopadla komprese při využití archivu „*tar*“ (při kterém není možno zvolit typ metody) programu 7-Zip. Kompresní poměr v tomto případě činil 1,00048, tedy komprimovaný soubor

byl větší než původní soubor. Porovnáme-li jednotlivé programy mezi sebou podle seřazeného kompresního poměru (vzestupně), pak první tři místa obsadil program WinZip, s archivem „*zipx*“ a metodami „*PPMD*“, „*LZMA*“ a „*XZ*“. Na čtvrté pozici jej následoval program 7-Zip s hodnotou kompresního poměru 0,11066 při použití metody „*LZMA2*“. Program WinRar se poprvé umístil na desáté pozici s hodnotou kompresního poměru 0,12429, a to při použití metody „*Nejlepší*“ a archivu typu „*RAR*“. Z měření dále vyplynulo, že u programů WinZip a WinRar, které používají popisné označování metod, podle kterých by se uživatel intuitivně řídil, tyto metody nedosáhly nejefektivnější komprese v rámci daného programu. U programu WinZip se jeho metoda „*Best method*“ s kompresním poměrem 0,11217, umístila až za metodami „*PPMd*“, „*LZMA*“ a „*XZ*“. Nejednalo se tedy o nejefektivnější metodu, jak název napovídá. Naopak u programu WinRar byla jeho metoda označená jako „*Nejlepší*“ skutečně nejefektivnější s kompresním poměrem 0,12429.

Jak ukázala analýza oficiálních webových stránek u zkoumaných programů a sekcí jejich nápovědy, tak obsahují spíše informace pro práci s programem samotným. Detailnější informace o algoritmech, které jednotlivé metody využívají na těchto stránkách prakticky nenajdeme. Pokud tedy bude běžný uživatel chtít tyto informace získat, musí počítat s časově náročnou rešerší zejména v IT fórech a IT literatuře, tedy jiných zdrojích, než které poskytuje výrobce programu.

Závěrem je tedy možné říci, že pokud uživatel užívá komerční programy, není téměř možné, aby si zvolil konkrétní algoritmus, i kdyby s konkrétními algoritmy byl obeznámen. Možnou cestou k aplikaci konkrétního algoritmu je vlastní implementace v některém z programovacích jazyků, nebo využití jejich již existujících knihoven. Do jaké míry tyto knihovny odpovídají originálnímu algoritmu, však nebylo předmětem této práce, a vyžadovalo by tak další zkoumání. V případě že uživatel zvažuje tvorbu archivu za použití komerčního software, jako například banka, která ukládá na disk výpisy z běžných účtů, doporučuji nejprve vyzkoušet všechny metody daného programu na reprezentativním vzorku dat a následně výsledek vyhodnotit. Uživatel musí vždy zvažovat, zda upřednostňuje velikost výstupního souboru na úkor například rychlosti komprese či dekomprese souboru. Tedy předem si definovat povahu archivu, který buduje a zodpovědět si, zda bude spíše statickým archivem, sloužícím spíše jako záloha dat, nebo dynamickým archivem, kde budou data často dotazována, a tedy bude třeba brát v úvahu rychlost dekomprese souboru. Tento prvotní průzkum a definici základních parametrů archivu je vždy nutné učinit před samotným procesem jeho tvorby, protože případná náprava chyb je náročná jak z hlediska časového, tak finančního.

BIBLIOGRAFIE A ZDROJE

7-Zip, 2019. 7-zip. *7-Zip* [online]. Dostupné z: https://www.7-zip.org/. [cit. 2020-10-25].

BZip2, 1996. bzip2. *Bzip2* [online]. Dostupné z: http://www.bzip.org/. [cit. 2020-10-22].

ČAPEK, Jan a FABIÁN, Peter. *Komprimace dat: principy a praxe*. Internet. Praha: Computer Press, 2000. ISBN 80-722-6231-9.

FOLTÝNEK, Tomáš a Jan PŘICHYSTAL, 2008. Komprimace a šifrování. *Mendelova univerzita v Brně* [online]. Dostupné z: https://is.mendelu.cz/eknihovna/slozky_objekty.pl?slozka=21;zobrazit=621;typ=opora. [cit. 2020-10-20].

GÁLA, Libor; POUR, Jan a ŠEDIVÁ, Zuzana. *Podniková informatika*. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2009. ISBN 978-80-247-2615-1.

JIROUŠEK, Radim, Jiří IVÁNEK, Petr MÁŠA, Jan TOUŠEK a Norbert VANĚK, 2006. *Principy digitální komunikace*. Voznice: Leda. ISBN 978-80-7335-084-0.

KRAUS, Jiří. *Nový akademický slovník cizích slov: [A-Ž: studentské vydání] / kolektiv autorů pod vedením Jiřího Krause*. Vyd. 1., dotisk [i.e. 1. brož. vyd.]. Praha: Academia, 2006. 879 s. ISBN 80-200-1415-2.

META, 2018. *5 ways Facebook improved compression at scale with Zstandard* [online]. META. Engineering at Meta. 2018. Dostupné z: https://engineering.fb.com/2018/12/19/core-infra/zstandard/. [cit. 2023-11-23].

MLÝNKOVÁ, Irena, Jaroslav POKORNÝ, a ČESKÁ SPOLEČNOST PRO SYSTÉMOVOU INTEGRACI, 2008. *XML technologie: principy a aplikace v praxi*. Praha: Grada. ISBN 978-80-247-2725-7[32].

MORKES, David, 1998. *Komprimační a archivační programy: jak funguje komprese binárních, textových a grafických dat : uživatelský popis programů: WinRAR, WinZIP, ARJ, JAR, ACE*. Praha: Computer Press. ISBN 978-80-7226-089-8.

POKORNÝ, Jaroslav, Václav SNÁŠEL, Michal KOPECKÝ, a UNIVERZITA KARLOVA, 2005. *Dokumentografické informační systémy*. Praha: Karolinum. ISBN 978-80-246-1148-8.

SALOMON, David, 2005. *Coding for data and computer communications*. New York, NY: Springer. ISBN 978-0-387-21245-6.

SALOMON, David, 2007. *Data compression: the complete reference*. 4th ed. London: Springer. ISBN1-84628-602-6.

SAYOOD, Khalid, 2017. *Introduction to data compression*. 5th edition. Cambridge, MA: Elsevier. ISBN 978-0-12-809474-7.

SLUNEČNICE.CZ, 2020. Slunečnice.cz. *Slunečnice.cz: programy rychle a zadarmo* c. Dostupné z: https://www.slunecnice.cz/sprava-a-zabezpeceni-pocitace/prace-se-soubory/komprese-souboru-zipovani/. [cit. 2020-05-31].

STAHUJ.CZ, 2020. Stahuj.cz. *Stahuj.cz Svět software* [online]. Dostupné z: https://www.stahuj.cz/utility_a_ostatni/komprese/. [cit. 2020-05-31].

WinRar, 2020. RARLAB WinRAR. *RARLAB WinRAR* [online]. Dostupné z: https://www.win-rar.com/. [cit. 2020-10-20].

WinZip, 2020. COREL CORPORATION. *WinZip* [online]. Dostupné z: https://www.winzip.com/en/. [cit. 2020-10-16].

[32] Jméno autorky Ireny Mlýnkové je dle citačního nástroje Citace Pro aktuálně Irena Holubová.

SEZNAM POUŽITÝCH TABULEK A OBRÁZKŮ

SEZNAM ZKRATEK

Bzip	Kompresní algoritmus využívající Burrowsovu–Wheelerovu transformaci a aritmetické kódování
BZip2	Kompresní algoritmus využívající Burrowsovu–Wheelerovu transformaci a Huffmanovo kódování
LZ77	Kompresní algoritmus publikovaný A. Lempelem a J. Zivem v roce 1977
LZ78	Kompresní algoritmus publikovaný A. Lempelem a J. Zivem v roce 1978
LZMA	Kompresní algoritmus založený na metodě LZ77 kombinovaný s Markovovými řetězci
LZMA2	Vylepšená verze algoritmu LZMA
PPMd	Varianta kompresního algoritmu, odvozeného ze skupiny tzv. PPM metod (z. angl. Prediction with Partial Match)
XZ	Kompresní metoda programu WinZip
Zstd	Kompresní metoda Zstandard vyvynutá společností Meta