Středoškolská technika 2016

Setkání a prezentace prací středoškolských studentů na ČVUT

Algoritmus pro bezztrátovou kompresi dat

Štěpán Pešout

Gymnázium, Vlašim, Tylova 271
Tylova 271, Vlašim

Úvod

Co je to komprese a jak ji dělíme

Když mluvíme v informatice o kompresi, jde o nějaké zmenšování dat. Rozlišujeme dva základní typy – ztrátovou a bezztrátovou. První druh se používá typicky u mediálních souborů, tedy u hudby, obrázků nebo videa. Nevýhodou je to, že tento typ komprese nenávratně poškodí takový soubor; zůstane sice běžně čitelný, ale sníží se jeho kvalita. Naopak je velmi výhodné to, že se data rapidně zmenší (tedy lepší kompresní poměr). Snižovat velikost (a zároveň kvalitu) je možné v podstatě do nekonečna (obr. 1).

Obr. 1: Ztráta kvality v závislosti na kompresním poměru
Zdroj: https://airsatone.com/img/products/flightstream/lossy_compression_example_720.jpg

Dalším druhem komprese je bezztrátová. Ta se používá, pokud je nutné zachování kvality nebo pokud by ztrátou byť jen jediné informace došlo k nečitenlosti dat – toto se objevuje typicky u textových souborů. Nevýhodou je obecně horší kompresní poměr, což znamená, že se data příliš nezmenší. Bezztrátové algoritmy zpravidla využívají redundantní data; to jsou taková, která se opakují (jsou v podstatě navíc) a je proto možné je uvést jen jednou a podruhé jen odkaz na ně. Zde je příklad:

Vstup: aaabbbbccc
Výstup: 3a4b3c

Důležitost komprimování

Zmenšování dat je důležité, protože se objem dat na celém světě zdvojnásobí za přibližně každé dva roky. Komprese souborů proto velmi šetří prostor – od kapesních zařízení po datová centra.

Skoro každý máme smartphone – představme si to tedy na tomto příkladu. Máme v paměti telefonu 64 GB místa a chceme prostor zcela zaplnit filmy ve FullHD rozlišení, které mají v průměru každý 2 GB. Pokud nepoužijeme kompresi, můžeme si stáhnout maximálně 32 filmů. Pokud s tím nejsme spokojeni, máme 2 možnosti:

1. Použít bezztrátovou kompresi a filmy zmenšit v průměru třeba na 70 % jejich původní velikosti – tedy na 1,4 GB pro každý z nich. Takových filmů potom do paměti vměstnáme asi 45.

2. Použít ztrátovou kompresi a spokojit se tak s horší kvalitou. Řekněme, že za kompromis mezi kvalitou a velikostí budeme považovat zmenšení na 40 % původní velikosti. Každý film bude potom zabírat 800 MB. Do paměti potom uložíme asi 80 takovýchto filmů.

Popis samotného kompresního algoritmu

Hlavní myšlenka

Algoritmus převede vstupní data na číslo, se kterým se dále pracuje (vizte podrobný popis níže). Poté je výsledek vyjádřen v původním kódování.

V podstatě neexistuje způsob, jak vyjádřit $k+1$ informací k informacemi, aby to fungovalo při jakémkoli k. Dokázal jsem, že to není možné v rámci jedné číselné soustavy. Z toho tedy vyplývá, že kompresní poměr bezztrátové komprese je vždy závislý na míře redundance vstupních dat.

Vstupní data

Vstupní a výstupní kódování je Base64. Toto kódování používá (jak už z názvu vyplývá) 64 různých znaků – jedná se o malá a velká písmena anglické abecedy (52 znaků), o čísla (10 znaků), o lomítko (/) a znak plus (+). Funguje to tak, že z každých 3 libovolných znaků (1 znak = 256 možností) v kódování Base64 vzniknou 4 znaky. Pokud tedy počet znaků není dělitelný třemi, po převodu se na konec řetězce Base64 přidá odpovídající počet znaků =, tak, aby byl počet znaků dělitelný čtyřmi. Znak rovnítka nepoužívám v tabulce z toho důvodu, že v delším řetězci jsou 1 až 3 nezkomprimované znaky zanedbatelné a po dokončení komprese se přidají na konec řetězce.

Důvodem, proč používám Base64 je mimo jiné i to, že 64 znaků se přesně vejde do tabulky 8 na 8 znaků. Poté se každý znak převede na dvouciferné číslo (viz tabulka). Pokud však použijeme hodnoty „popisků“ v horní řadě a levém sloupci, dostaneme číslo které je de facto v osmičkové soustavě. To je důležitá informace (důvod popisuji dále v textu).

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	*a*	*A*	*b*	*B*	*c*	*C*	*d*	*D*
1	*e*	*E*	*f*	*F*	*g*	*G*	*h*	*H*
2	*i*	*I*	*j*	*J*	*k*	*K*	*l*	*L*
3	*m*	*M*	*n*	*N*	*o*	*O*	*p*	*P*
4	*q*	*Q*	*r*	*R*	*s*	*S*	*t*	*T*
5	*u*	*U*	*v*	*V*	*w*	*W*	*x*	*X*
6	*y*	*Y*	*z*	*Z*	*1*	*2*	*3*	*4*
7	*5*	*6*	*7*	*8*	*9*	*0*	/	+

Je tedy zřejmé, že mohu reprezentovat libovolný znak z tabulky podle hodnoty v horním řádku a levém sloupci. Není přitom zapotřebí žádného oddělování mezi znaky vyjádřenými čísly, neboť zápis každého ze znaků je roven právě jednomu číslu, které je vždy dvouciferné. Výsledný číselný řetězec nazývejme x. Například:

Vstup: Ahoj
Výstup: 01163422

Informační znaky

Dále v textu IZ. Jedná se o n znaků v kódování Base64 (tedy 64^n kombinací), které se přidají na začátek řetězce a neprovádí se u nich komprese. Slouží k zapsání, zda a jakým způsobem proběhly obě metody komprese, které popisuji níže.

Číslo n se mění pode toho, kolik místa pro zápis informací v IZ je potřeba. Velikost n se uvede před IZ jako dvouciferné číslo. Pokud je například $n=5$, před řetězec se zapíše jako 05. Obecně je ale snaha o to, aby mělo n co nejnižší možnou hodnotu.

První metoda

Číslo je převedeno do dvojkové soustavy. Ve výsledku jsou nahrazeny nuly a jedničky počty těchto znaků. Využívá se také toho, že číslo ve dvojkové soustavě začíná vždy jedničkou.

Pokud bude tento počet roven 8, zapíše se jako nula. Od 10 výše se používají znaky malé anglické abecedy, tedy a až z. Ačkoli je následující situace velmi málo pravděpodobná, může se stát, že bude pohromadě více znaků jednoho druhu, než 35. V takovém případě se první metoda neprovede, což se zapíše do IZ. Vždy určíme číselnou soustavu o nejnižším možném základu a výsledek poté převedeme zpět do osmičkové soustavy. Uvedu příklad:

Vstup (tj. x): $(37017)_8$
Převod do dvojkové soustavy: $x_8 = (11111000001111)_2$
Nahrazení počty znaků: 554
Převod zpět do osmičkové soustavy: $(554)_6 = (326)_8$

Tato úprava řetězce se neprovede, pokud platí, že délka výstupu ≥ délka x. V opačném případě se do x přiřadí výstup této metody. Informace o tom, zda došlo k nahrazení a jestli se převádělo z jiné číselné soustavy se zapíše do IZ.

Pokud se to vyplatí, následuje další provedení první metody a takto do té doby, dokud nepřestane platit zmíněná nerovnice délka výstupu ≥ délka x. Kolikrát se povedlo metodu provést, je zaneseno v IZ.

Druhá metoda

Zde se pracuje s číselnými soustavami o základu v intervalu $\langle 8; 36 \rangle$, přiřemž základy leží v $\mathbb{Z}$.

Začněme se základem 8. Nejprve se určí, která dvojice čísel se nejčastěji v řetězci vyskytuje. Takováto dvojice je poté všude nahrazena číslem 8. Výsledek je převeden zpět do osmičkové soustavy a délka odečtena od délky původního řetězce. Výsledek tohoto rozdílu se zapíše společně s informací, že šlo o dvojice a osmičkovou číselnou soustavu.

Obdobně se pokračuje i u trojic, čtveřic, apod. Záleží na délce vstupu, jak dlouhé budou skupiny číslic. Nejdelší skupina je vždy třísetina délky vstupu (výjimkou jsou dvojice, které se hledají vždy). Najde se nejčastěji se opakující skupina číslic a ta se nahradí číslem 8. Poté dojde opět k převodu do osmičkové soustavy a k výpočtu rozdílu délky původního řetězce a výstupu. Zapíše se rozdíl zároveň s informací, o jak velkou skupinu čísel šlo.

Nyní máme v podstatě tabulku skupin čísel a rozdílů. Vybere se výstupní řetězec odpovídající nejvyššímu rozdílu. Informace o tom se zapíše.

Nyní vysvětím, jak funguje nahrazování v rámci soustav o vyšším základu. Funguje to v podstatě stejně jako v rámci devítkové soustavy, jen se hledají větší počty opakujících se skupin čísel, které se nahrazují znaky z příslušné soustavy. V číselné soustavě o základu r bude tedy algoritmus hledat a nahrazovat $r–8$ nejčastěji se opakujících skupin pomocí znaků, které osmišková soustava neobsahuje. Poté je výstupní řetězec převeden zpět do osmičkové soustavy a obdobně vypočítán rozdíl. Toto se provede (jak jsem již zmínil) v rámci všech číselných soustav o celočíselných základech v intervalu $\langle 8; 36 \rangle$.

Informace o tom, u jak velké skupiny čísel v rámci jaké číselné soustavy se povedlo dosáhnout nejvyššího rozdílu, se zapíše. Opět se nám tedy vytvořila tabulka, ve které díky rozdílům vidíme, kde došlo k největšímu zkrácení. Informace o tom se zapíše do IZ. Podle informací, které máme, se může znovu vytvořit nejkratší výstupní řetězec (abychom nemuseli všechny výsledné řetězce ukládat). Tento pak nahradí původní vstupní řetězec, který jsme měli před druhou metodou za předpokladu, že je nejkratší řetězec kratší, než vstupní. Výsledek této operace se opět objeví v IZ.

Uvedu příklad (pro jedenáctkovou soustavu při hledání dvojic). Volím záměrně takový vstup, aby bylo vidět zkrácení:

Vstup: 52323052722377316522431423
Vyhledání, které 3 se opakují nejčastěji: 52323052722377316522431423
Jedná se o dvojice 52, 23 a 31
Nahrazení: 839087297 7a68 24a49
Převod do osmičkové soustavy: $(352160215373303021411)_8$
Rozdíl délek: 5 (došlo tedy ke zkrácení)

Druhá metoda je zacyklená; provádí se do té doby, než neexistuje žádný výstupní řetězec, který by byl kratší, než vstupní. Kolikrát se metoda provedla a o jaké nejkratší řetězce šlo, se zapíše do IZ.

Zkomprimování díky výpočtům z druhé metody

Shrňme si, co máme v IZ:

1. kolikrát se provedla první metoda spolu s informací, jak se v jednotlivých opakováních převádělo na úrovni číselných soustav;
2. jaká kombinace délek nahrazovaných skupin v rámci jakých číselných soustav vede vždy po převodech na osmičkovou soustavu k nejkratšímu výstupnímu řetězci a zda tento řetězec není delší, než ten vstupní.

Pokud je délka výstupu z druhé metody větší nebo stejná než délka vstupu a zároveň se nepovedlo provést první metodu, můžeme prohlásit, že komprese se nevyplatí a algoritmus tedy spolu s touto informací vrátí původní řetězec.

Jestliže předchozí podmínka neplatí, vezme se řetězec ve stavu před druhou metodou (přičemž nezáleží na tom, zda se provedla metoda první) a podle kombinace uvedené v IZ jej algoritmus zkrátí.

Když se povede zkrátit řetězec pomocí první nebo druhé metody, celý algoritmus se zopakuje. Informace o tom, kolikrát se komprese opakovala, je zanesena v IZ.

Převod zpět do Base64

Pokud se komprese vyplatila, máme její výstup jako číslo v osmičkové soustavě. Aby se uživateli vypsala hodnota v původním kódování, převedeme jej podle tabulky uvedené u vstupních dat. Například:

Vstup: 01163422
Výstup: Ahoj

Může se stát, že číslo bude mít lichý počet míst. Poslední číslice se takovém případě odstraní, aby se mohla použít tabulka. Do IZ poté zapíšeme, zda došlo k odstranění a pokud ano, tak o jakou číslici šlo.

Dekomprese

V IZ máme napsány informace o průběhu komprese oběma metodami. Algoritmus tedy převede data z Base64 do číselného formátu pomocí tabulky uvedené na straně č. 3. Pomocí dat z IZ se provede dekomprese. Představme si, že jsme například komprimovali takto:

1. Převedení z Base64 na číslo podle tabulky
2. Provedla se první metoda, nepřevádělo se.
3. Provedla se druhá metoda, k největšímu zkrácení došlo v soustavě o základu 24 při nahrazování trojic.
4. Provedla se první metoda, převádělo se z trojkové soustavy.
5. Převedení na kódování Base64 podle tabulky

Dekomprese je tedy v podstatě obráceně provdená komprese v závislosti na datech v IZ:

1. Převedení z Base64 na číslo podle tabulky
2. Číslo se převede z trojkové do osmičkové soustavy a zapíše se pomocí střídavého nahrazování počtů znaků nulami a jedničkami. Následuje převod z dvojkové zpět do osmičkové soustavy.

3. Číslo se převede do soustavy o základu 24, kde nahradíme znaky nepatřící do osmičkové soustavy trojicemi znaků tak, jak je uvedeno v IZ.
4. Číslo se zapíše se pomocí střídavého nahrazování počtů znaků nulami a jedničkami. Následuje převod z dvojkové zpět do osmičkové soustavy.
5. Převedení na kódování Base64 podle tabulky

Závěr

Kompresní poměr

Vzhledem k tomu, že lze jen těžko říci, jaká bude míra redundance vstupních dat, nemůžeme spočítat kompresní poměr, který bude platit vždycky.

Co mohu tedy říci je to, že při kompresi textu dojde nejspíše k většímu zkrácení, než například u mediálních souborů. Pokud zvážíme, že text v anglickém jazyce používá pouze kolem 60 různých znaků, což je asi jedna čtvrtina z celkových 256, je jasné, že se znaky budou opakovat častěji i po převodu na Base64.

Využití

V tuto chvíli jde pouze o sepsanou koncepci, ale pokud bych toto později nakódoval a distribuoval, domnívám se, že by algoritmus mohl najít využití zejména v pamětech mobilních zažízení nebo na cloudu. Tam se uživatelé potýkají s malou kapacitou a za rozšíření prostoru si musí připlácet.