Komprese dat, rozdělení kompresních metod, princip kompresních metod (Huffmann, aritmetické kódování, LZW, JPG, fraktálová komprese)

Z FAV wiki

Algoritmy pro kompresi dat používáme pro snížení velikosti přenášených dat

- z paměťových důvodů
- ušetření šířky pásma
- urychlení přenosu a zkrácení doby odezvy
- archivace dat, zálohy
- obrana proti virům
- distribuce software
- ...

Kvalita komprese - úroveň zmenšení, poměr komprese = před/po. Symetrické metody potřebují stejný čas ke komprimaci jako k dekomprimaci

Obsah

Rozdělení kompresních metod

- **Bezztrátová** - po komprimaci a dekomprimaci je výsledek k nerozeznání od originálu, 100% shoda. Metody jsou méně efektivní. Použijeme na text, obecné soubory
- **Ztrátová** - po komprimaci a dekomprimaci je výsledek rozdílný, cílem je však aby byl co nejvíce podobný. Více efektivní. Použijeme na zvuk, obraz, video.

Metody komprese

- **Jednoduché** - založené na kódování opakujících se posloupností znaků (Run-Length Encoding - RLE)
- **Statistické** - založené na četnosti výskytu znaků v komprimovaném souboru (Huffmanovo kódování, Aritmetické kódování)
- **Slovníkové** - založené na kodování všech vyskytujících se posloupností (Lempel-Ziv-Welch - LZW)
- **Transformační** - založené na ortogonálních popř. jiných transformacích (JPEG, waveletová komprese, fraktálová komprese)

RLE

RLE (Run Length Encoding) – kódování délkou běhu

Nejjednodušší komprimační algoritmus, nahrazuje opakující se znaky jedním znakem a číslem vyjadřujícím počet okpakování (AAAAABBBBBCCD -> 5A5B2C1D) Pro text k ničemu, pro obraz celkem dobrá komprese (zvláště obrazy používající malou paletu barev) Problém: pokud se znak neopakuje, dochází k expanzi (ABC -> 1A1B1C) - Řešení pomocí escape sekvencí (označíme si, co je délka a co ne) a poté komprimujeme pouze opakující se znaky. Escape znak nesmí být obsažen v původním textu.

00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001111000000000000000000000001110	20 4 23 3 1
00000000000000000000011100000000000000000000111000	22 3 20 3 3

Ukázka komprese bitového souboru pomocí RLE. Ukládáme vždy počet nul a jedniček v řadě.

Huffmanovo kódování

Statistická metoda. Vyhledáme četnosti všech znaků v řetězci a tyto znaky podle četnosti zakódujeme tak, aby nejkratší kód odpovídal nejčastějšímu znaku. Kódy jsou binární, nejčastější znak bude tedy reprezentován 1 bitem na rozdíl od 8(16) bitů potřebných pro uložení ASCII(UTF) znaku. Jená se o prefixový kód, takže žádný znak není prefixem jiného znaku -> kódy pro jednotlivé znaky mají různou délku a není potřeba označovat jejich konec (každý znak kreprezentuje jeden list stromu).

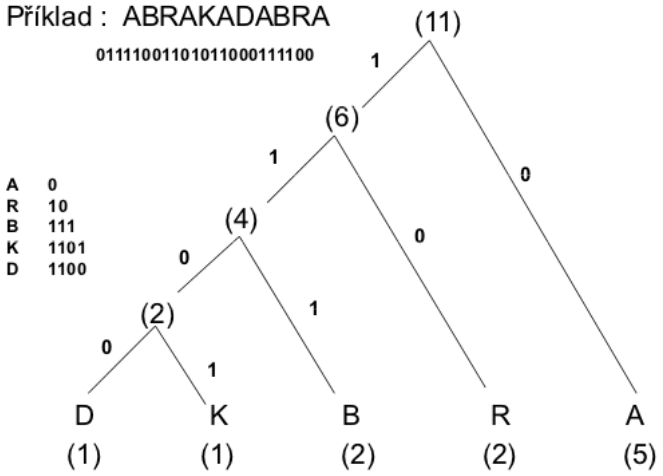
Kódy vytvoříme Huffmannovým stromem:

- Jednotlivé znaky označíme jako listy stromu
- Tyto listy uložíme do seznamu S (S obsahuje uzly stromu, na začátku tedy pouze listy)
- Seznam S seřadíme podle četnosti

- Vybereme z S dva prvky M,N s nejnižšími četnostmi m, n, m < n
- Vytvoříme uzel p, kde vlevo je M a vpravo je N, a jeho četnost bude m+n
- Vložíme p do S a zpět na bod 3 dokud v S není jen jeden uzel

Strom samozřejmě musíme přenášet spolu s komprimovanými daty. Strom můžeme reprezentovat řetězcem tak, že jej projdeme DFS algoritmem a pro haždý vrchol uložíme 0 a pro každý list 1 a znak tohoto listu. Při načítání čteme nuly a vytváříme uzly (směrem doleva). Pokud narazíme na 1, zapíšeme písmeno a vrátíme se do nejbližšího pravého uzlu (pravý uzel rodiče, případně nejvíce levý prvek podstromu - u rodiče vpravo)

Při dekompresi procházíme strom a rekonstruujeme zprávu.



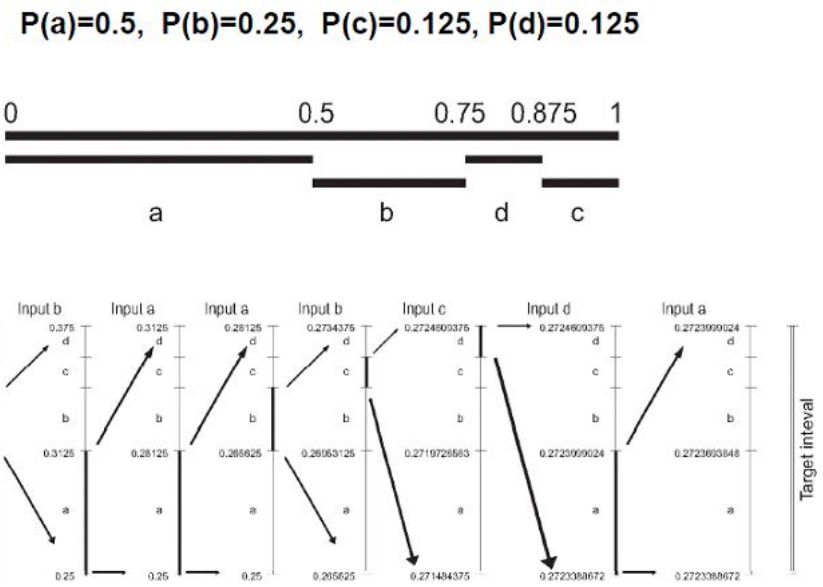
Aritmetické kódování

Také statistická metoda. Aritmetické kódování na rozdíl od jiných metod nepracuje na principu nahrazování vstupního znaku specifickým kódem. Místo toho se kódovaný vstupní proud znaků nahradí jediným reálným číslem z intervalu <0,1).

Na základě pravděpodobnosti výskytu jednotlivých symbolů vstupní abecedy je každému symbolu přiřazena odpovídající poměrná část intervalu <0,1). Při kódování je pak celý interval <0,1) postupně omezován z obou stran na základě postupně přicházejících symbolů (Interval ořízneme jen na tu část, které odpovídá pravděpodovnost písmene v textu, a tento interval opět rozdělíme podle pravděpodobností pro další písmeno).

Kódovaná hodnota se reprezentuje libovolným reálným číslem, které leží ve výsledném intervalu získaném po přečtení všech vstupních symbolů. Vzhledem k tomu, že z takto reprezentované hodnoty nelze při dekódování určit konec zprávy, je třeba navíc ke zprávě přidat zpeciální znak označující konec, případně musí být uložena i délka původní posloupnosti.

Příklad: kódujeme zprávu 'abaabcda'



Co z toho číst?

- První řádka ukazuje pravděpodovnosti písmen (ty vypočteme jako četnost písmene / písmen celkem)
- Dále vidíme, jak bude interval (a podintervaly) rozdělený. 'a' jako nejpravděpodobnější písmeno dostane největší část intervalu atd...
- A dále už kódujeme. Jdeme zleva do prava. Přijde písmeno 'a', vybereme tedy spodní interval. Tento opět rozdělíme na stené části jako interval původní a přijde 'b'. Opět vybereme odpovídající interval, který rozdělíme atd.
- Zvýrazněná část intervalu se na obrázku vždy zazoomuje ;) (jsou tam šipky).
- Na konci máme jen jeden interval, vybereme tedy číslo , které do něj patří. Zpráva je zakódována v desetinné části tohoto čísla (a ano, celá zpráva je reprezentována jedním číslem)

LZW

Z přednášek, doplněno: Slovníkový algoritmus. Využívá skupin znaků označených kódy a přenos kódů, které mají menší velikost než původní skupiny. Princip: Jednoprůchodová metoda (nevyžaduje předběžnou analýzu souboru. Ve výsledku se jen posunujeme po souboru jedním ukazatelem.) Vyhledávání opakujících se posloupností znaků, ukládání těchto posloupností do slovníku pro další použití a přiřazení jednoznakového kódu těmto posloupnostem. Kódy musí mít délku (počet bitů) větší než délka původních znaků (např. pro ASCII znaky (8 bitů) se obvykle používá délka kódu 12 bitů popř. větší. Na osmi bitech kódu by se dal uložit pouze slovník o 256 slovech(posloupnostech znaků), což odpovídá pouze počtu písmen, což je k ničemu.) Při průchodu komprimovaným souborem se vytváří slovník (počet položek slovníku odpovídá hodnotě 2(počet bitů kódu), kde prvních 2(počet bitů původních znaků) položek jsou znaky původní abecedy a zbývající položky tvoří posloupnosti znaků obsažené v komprimovaném souboru (používáme tedy i kódy z již komprimovaného souboru k vytvýření nových kódů). Při dekompresi procházíme kódy, a podle tabulky je rozebíráme na menší a menší, až se dostaneme na elementární znaky, které vypíšeme.

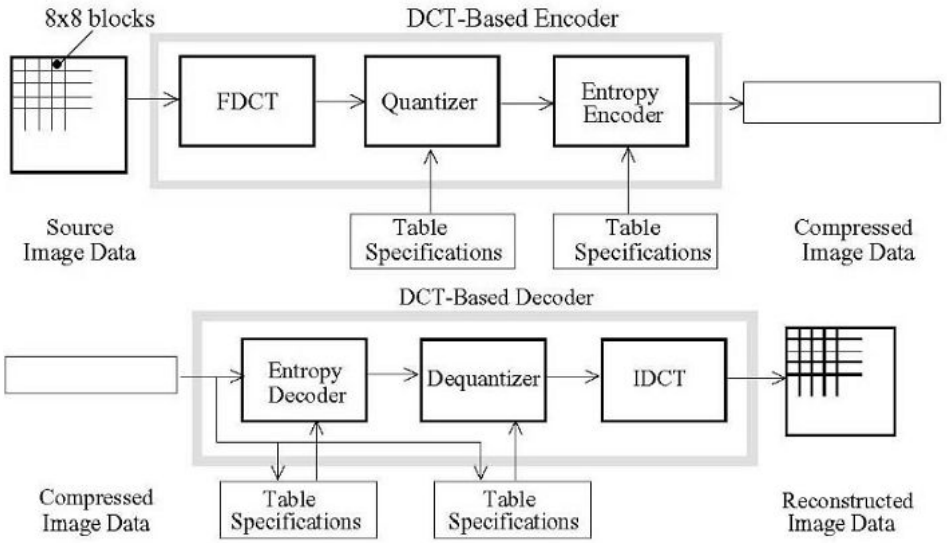
Aby to celé fungovalo, musíme slovník na dekomprimační straně "naučit" slova od menších po větší. Musíme proto na začátku souboru definovat elementární znaky, a z nich poté skládat slovník. Pro ASCII tedy prvních 256 znaků tabulky může odpovídat 1:1 ASCII abecedě, až poté následuje slovník. Pokud při dekompresi narazíme na neznámý kód, víme, že je složen z předchozích 2 kódů. Proto se jej naučíme, a příště už víme, co zapsat (případně jak jej rozložit). Tabulka na obou stranách tedy je stavěna stejně, jen používána z jiné strany (při komprimaci ukládáme kódy, pri dekomprimaci jejich znaky)

Pro obrázky viz přednášky z PT, jsou celkem snadno pochopitlené, i když místo "kód" občas autor používá "znak" či "nový znak", což je trochu matoucí.

JPG

V současné době patří mezi nejvíce používané komprese u obrázků Je vhodná pro komprimaci fotek, nevhodná pro např. technické výkresy (čarové výkresy) - dochází k viditelnému rozmazání (na webové aplikace pomalu převládá png (zejména kvůli neztrátovosti a průhlednosti))

Princip: části obrazu se transformují do frekvenční oblasti (výsledkem je matice „frekvenčních“ koeficientů, bezztrátový převod) z matice koeficientů se odstraní koeficienty odpovídající vyšším frekvencím (rychlejší změny jasu - např. hrany v obraze, zde jsou ztáty) zbývající koeficienty se vhodným způsobem zkomprimují (huffman, aritmeticky)



Fraktálová komprese

Fraktál je obraz, který je matematicky popsatelný a tím jej lze libovolně přibližovat a oddalovat bez stráty informace. Také lze fraktál popsat jako soubor matematicky popsaných sebepodobných částí, jejichž spojováním dostáváme také sebepodobný obraz ve větším detailu (jako když děláme koláž z fotek, které dohromady a z dálky vypadají jako fotka)

Toho lze využít pro kompresi. Pokud se podíváme na libovolný obrázek, často zjistíme, že některé jeho části jsou si podobné

Tyto malé kousky (frakce) lze pomocí rotace, změny velikosti a posunu otočit tak, že si budou velmi podobné (nebo dokonce budou stejné). V obrazu tedy můžeme na takový kousek mít pouze jednu, a v ostatních výskytech pouze uložit tyto operace.

V praxi obrázek rozřežeme na shluky 2x2 pixely (range bloky), a pak hledáme, kde jinde v obrazu je můžeme najít (domain bloky). Domain bloky se mohou překrývat, range bloky se nepřekrývají. Postupujeme tak, že Domain blok vytvoříme jako dvojnásobek range bloku, a v obrazu postupujeme zleva do prava. V bloku uděláme průměr hodnot pixelů pod ním tak, abychom získali blok velikosti range bloku, a poté hledáme, kterému range bloku je takto zmenšený domain blok podobný. Uložíme si operace, které s domainblokem musíme udělat, aby byl range bloku podobný (rotace, překlopení, posun...) Pokud je domainblok složen z domainbloků, postupujeme rekurzivně. Tím, pomocí podobnosti, získáme komprimovaný obraz.

Kategorie: Fav-kiv-bzinf