Kódování a komprese dat 2018/2019: Projekt

Téma: Komprese obrazových dat s využitím statického a adaptivního Huffmanova kódování
Datum odevzdání: nejpozději do 05. 05. 2019 včetně prostřednictvím IS FITu.
Forma odevzdání: elektronicky - soubory v archívu ZIP
Počet bodů: max. 30
Poznámka: k zadání projektu bude přes IS FITu zveřejněna sada testovacích obrázků
Dotazy: simekv@fit.vutbr.cz

Zadání:

Cílem projektu je v programovacím jazyce C/C++ vytvořit aplikaci pro kompresi šedo tónových obrazových dat, kde se uplatní principy statického a adaptivního Huffmanova kódování (anglicky Adaptive Huffman Coding) [1, 2]. Vytvořené řešení musí být spustitelné coby tzv. konzolová aplikace (tedy z příkazového řádku) pod OS Linux v prostředí počítačové sítě FIT VUT v Brně.

Podrobné pokyny k vypracování:

a) formát vstupních dat:

Vstupní data pro tuto aplikaci budou reprezentována sadou obrázků ve formátu RAW bez použití komprese. Jednotlivé obrazové body (pixely) těchto referenčních obrázků pak mohou nabývat 8-bit hodnot odstínů šedi v intervalu 0 až 255.

b) zpracování vstupních dat:

Vytvořená aplikace musí být schopna volitelně (na základě parametru v příkazovém řádku) pracovat se vstupními obrazovými daty dvěma různými způsoby. Tedy buď bereme v potaz přímo hodnotu samotných pixelů nebo bude použit vhodný model (např. výpočet diferencí sousedních pixelů), jehož prostřednictvím jsou vstupní data nejdříve předzpracována.

c) funkce a uživatelské rozhraní:

Výsledné řešení bude mít charakter aplikace spustitelné z příkazového řádku pomocí příkazu *huff_codec*, který získáme po překladu zdrojových kódů vlastní implementace. Funkci aplikace bude možné řídit pomocí sady přepínačů či parametrů zadávaných na příkazovém řádku:

-c aplikace bude vstupní soubor komprimovat,
-d aplikace bude vstupní soubor dekoprimovat,
-h static zvolí režim statického Huffmanova kódování
-h adaptive zvolí režim adaptivního Huffmanova kódování
-m aktivuje model (např. diference pixelů) pro předzpracování vstupních dat
-i <ifile> název vstupního souboru. Dle režimu činnosti aplikace, viz přepínače –c/-d výše, se jedná buď o data určená ke kompresi nebo dekompresi.
-o <ofile> název výstupního souboru. Dle režimu činnosti aplikace, viz přepínače –c/-d výše, se jedná buď o zkomprimovaná či dekomprimovaná data,
-h vypíše nápovědu na standardní výstup a ukončí se.

d) komentáře zdrojových kódů:

Zdrojový kód komentujte tak, aby nebylo potřeba dodatečně projekt obhajovat. Komentujte zejména parametry a činnost funkcí, datové typy, lokální proměnné (pokud to nebude z názvu přímo vyplývat).

Všechny zdrojové soubory budou obsahovat hlavičku, ve které bude jméno, příjmení a login autora. Dále bude v hlavičce datum vytvoření, název a stručný popis souboru.

e) kompilátor a prostředí:
Odevzdané řešení musí být možné přeložit/spustit na serveru ***merlin*** či ***kazi***, kde bude také probíhat testování funkčnosti a měření výkonnosti aplikace. Tedy údaje v dokumentaci vašeho řešení zmiňované v bodě g) by měly být naměřeny právě v prostředí těchto dvou serverů.

f) dokumentace:
Součástí odevzdaného řešení bude stručná dokumentace. Vytvořená dokumentace bude na začátku obsahovat stručný rozbor řešeného problému, kdy rozhodně není žádoucí opisovat dlouhé teoretické statě z odborné literatury či studijních materiálů. Dále je třeba se věnovat nástinu vlastního řešení včetně např. vysvětlení funkce důležitých datových struktur a dalších relevantních aspektů.
V neposlední řadě by měla dokumentace obsahovat i stručný návod k překladu/zprovoznění aplikace. Nezapomeňte v dokumentaci uvést i odkazy na použité informační zdroje, z nichž jste čerpali během řešení projektu.

Povinnou součástí aplikace je taktéž vyhodnocení činnosti aplikace v těchto režimech činnosti:

- statické Huffmanovo kódování bez modelu
- statické Huffmanovo kódování s modelem /(ly)
- adaptivní Huffmanovo kódování bez modelu
- adaptivní Huffmanovo kódování s modelem /(ly)

Vyhodnocením se zde myslí změření kompresní účinnosti vyjádřené např. coby průměrný počet bitů potřebný na zakódování bajtu obrazu a času komprese ve výše uvedených případech pro zpracování přiložených obrazových dat. Naměřená data prezentujte formou tabulky. Uveďte také průměrnou hodnotu pro jednotlivé metody vypočtenou pro všechna přiložená data.

Rozsah dokumentace by neměl přesáhnout max. 4 strany formátu A4. Formát dokumentace bude PDF.

g) odevzdává se:
Archív ve formátu ZIP s názvem *kko_"login".zip* (např. pro Pepu Zdepa kko_xzdepa97.zip) bude mít následující obsah:

- zdrojové soubory vlastní implementace
- Makefile soubor pro překlad
- dokumentaci ve formátu PDF

Pokud to bude z pohledu řešení účelné (lepší strukturovaní zdrojových kódů, modulárnost aplikace), je samozřejmě možné vytvořit i další soubory (různé pomocné knihovny, hlavičkové soubory, atd) kromě těchto požadovaných. Nezapomeňte je však umístit do odevzdávaného archívu.

h) bonusové body:
Prvních 10 řešitelů s nejefektivnější implementací z pohledu dosažené míry komprese a rychlosti komprese a dekomprese má možnost získat až 5 bonusových bodů.

Literatura:

[1] Salomon, D.: "Data Compression, The Complete Reference," NY, USA, Springer, 2000, ISBN-0-387-95045-1

[2] Učební texty do předmětu Kódování a komprese dat, Brno, FIT VUT, 2006

Tipy a rady na závěr:

- Pro řešení změny měřítka (Huffmanova stromu) tak, aby nedocházelo k neuspořádání (viz obr. 3.32i, s. 66 [2]), je možný následující postup. Bereme postupně listy od nejnižšího výskytu (zleva doprava) a u každého takového listu provedeme následující operace: vydělíme dvěma počet výskytu symbolu listu a upravíme váhy rodičovských uzlů. Poté zkontrolujeme, případně přestavíme strom dle postupu, který patří k obrázkům 3.32e,f,g, s. 66 [2].
- Důležitou vlastností je také přenositelnost. Pro různé platformy může být velikost typů int, long různá. Těmto problémům se vyhneme použitím předdefinovaných typů *int8_t, int16_t, int32_t, u_int_8_t* apod., které jsou definovány v knihovně sys/types.h.
- Pro ladění potíží s pamětí (Segmentation fault apod.) doporučuji použít nástroj *valgrind*, který se používá na aplikaci zkompilovanou pomocí *make debug* (příp. *gcc -ggdb3*).
- V licenci GPL existuje také pro účely ladění pohodlný debugger *DDD* (DataDisplayDebugger), který je grafickou nadstavbou nad *gdb*. Opět je vhodné kompilovat kód pomocí *make debug*.
- Pro zpracování parametru příkazové řádky je výhodné využít funkce *getopt* (resp. getopt_long), jejíž rozhraní je definováno v unistd.h (resp. getopt.h).
- Pro implementaci některých datových struktur a operací s nimi může být výhodné použít v aplikaci knihovnu STL (C++ Standard Template Library).
- Pokud se rozhodnete využít v projektu transformaci BWT, je možné použít již existující knihovnu, např. https://github.com/y-256/libdivsufsort. Knihovnu je však nutné přibalit k odevzdanému řešení, aby bylo projekt možné přeložit pomocí Makefile. BWT je jediný případ, kdy je možné použít kód třetí strany. Ostatní zdrojové kódy musí být vašim autorským dílem.