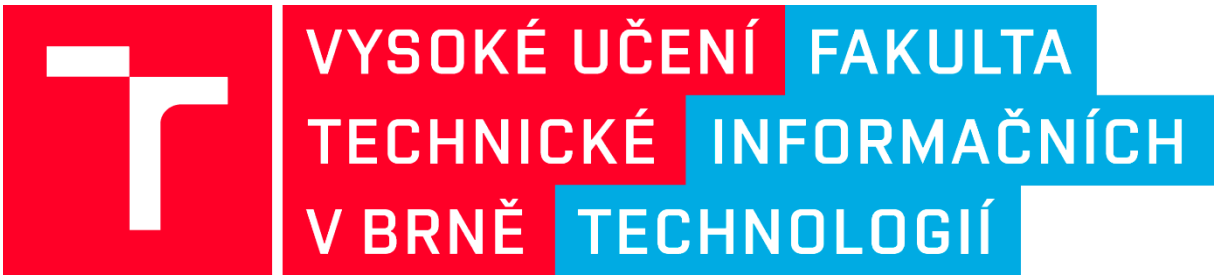

Dokumentace k projektu pro předmět KKO

Projekt č. 1

Adaptivní Huffmanovo kódování

5. května 2017

Autor: Bc. Lukáš Pelánek, xpelan03@stud.fit.vutbr.cz

Úvod

Cílem projektu bylo seznámit se a implementovat algoritmus realizující adaptivní Huffmanovo kódování nad 8 bitovými znaky. Jako implementační jazyk byl použit jazyk C++11.

Adaptivní huffmanovo kódování

Algoritmus vychází z původního Huffmanova kódování. Je využíván pro bezeztrátovou kompresi dat. Algoritmus funguje tak, že konvertuje znaky vstupního souboru do bitových řetězců, které mají různou délku. Znaky, které se ve vstupním souboru vyskytují nejčastěji, mají odpovídající bitové řetězce s nejkratší délkou. Zatímco znaky, které nejsou tolik frekventované, mají odpovídající bitové řetězce delší. Oproti statické metodě huffmanova kódování, jenž prochází vstupní soubor dvakrát, tak adaptivní huffmanovo kódování prochází vstupní soubor pouze jedenkrát. Kódování je založeno na budování stromu během zpracovávání vstupních dat. Na počátku strom obsahuje jediný symbol NYT (Not Yet Transfered). Pokud se symbol objeví na vstupu poprvé, tak je zapsán nezakódovaný a vložen do stromu. Při každém dalším načtení znaku, který je již v stromě obsažen, je vypsán jeho kód a inkrementován počet výskytů. Následně je nutné zkontrolovat, zda strom splňuje sourozeneckou vlastnost. Binární strom má sourozeneckou vlastnost, pokud splňuje následující podmínky:

- Každý uzel má sourozence (kromě kořenového uzlu)
- Součet ohodnocení sourozenců se rovná ohodnocení otce
- Uzly lze uspořádat do rostoucího seznamu podle ohodnocení tak, že sourozenci spolu sousedí

Pokud tato vlastnost není splněna, tak je potřeba strom přeuspořádat, aby opět splňoval sourozeneckou vlastnost. Při dekódování dekodér zrcadlí operaci kodéru. Když přečte znak NYT, tak následuje znak, který následně uloží do stromu a případně jej přeuspořádá.

Implementace

Aplikace nejprve zpracuje vstupní parametry pomocí *getopt()*. Otevře vstupní soubor nebo standartní vstup a zahájí kódování nebo dekódování podle zvoleného přepínače.

Kódování začíná vytvořením kořenového uzlu a pole uzlů o velikosti dvojnásobku maximálního počtu znaků (zde budou uloženy znaky a rodičovské uzly, které neobsahují znak). Do kořenového uzlu je vložen znak NYT a kodér začne zpracovávat vstupní soubor. Pokud se na vstupu objevil symbol, který dosud nebyl vložen do stromu, tak vytiskne kód NYT do souboru a následně nezakódovaný symbol. Vytvoří pro nový znak uzel a vloží jej do stromu. Pokud se na vstupu objevil symbol, který byl již načten, tak je do souboru uložena cesta stromem k uzlu a inkrementován počet výskytů. Cesta začíná od uzlu a postupně ukládá cestu na zásobník, dokud nedojde ke kořenu. Pokud je aktuální uzel pravým potomkem svého rodiče, uloží na zásobník 1, jinak 0. Následně je cesta vypsána od kořene k uzlu do výstupního souboru. Nakonec zkontroluje, zda strom stále splňuje sourozeneckou vlastnost. Tento postup probíhá do té doby, než je přečten celý vstupní soubor. Nakonec po přečtení celého vstupního souboru je do výstupního souboru vytisknut znak NYT signalizující konec souboru.

Dekódování rovněž začíná vytvořením kořenového uzlu a pole uzlů. Následně je rovněž vložen symbol NYT do stromu. Dekodér čte vstupní soubor a pro jednotlivé bity přečteného bajtu vykonává následující operace. Pokud předchozí symbol byl NYT, tak dekodér přečte celý bajt ze vstupního souboru. Tím získá nový znak, který vytiskne do výstupního souboru a vloží nový znak do stromu a provede kontrolu, zda se jedná o huffmanův strom. Pokud předchozí symbol nebyl NYT, tak načítáme jednotlivé bity a procházíme stromem. Když narazíme na znak, tak jej vypíšeme do výstupního souboru, inkrementujeme počet výskytů daného znaku a opět zkontrolujeme, zda se jedná o huffmanův strom. Pokud jsme stromem došli k symbolu NYT, tak opakujeme předchozí krok. Po

přečtení vstupního souboru aplikace zkontroluje, zda poslední znak byl NYT. Pokud ano, tak dekódování bylo úspěšné.

Při kontrole, zda strom splňuje sourozeneckou vlastnost, aplikace prochází polem symbolů a hledá symbol, který má stejný počet výskytů, ale menší ohodnocení než aktuálně modifikovaný uzel. Pokud takový uzel najde, tak musí provést výměnu. Nejprve provede výměnu synovských uzlů rodiče aktuálního uzlu. Poté zamění synovské uzly rodiče prvku, se kterým bude vyměněn. Nakonec prohodí rodičovské uzly a ohodnocení daných uzlů. Díky této výměně strom opět splňuje sourozeneckou vlastnost.

Závěr

Nastudoval jsem problematiku a naimplementoval adaptivní Huffmanovo kódování v jazyce C++11. Aplikaci jsem otestoval na referenčním testovacím souboru. Velikost vstupních dat před kompresí byla 768771 bajtů. Po kompresi byla velikost vstupních dat 438511 bajtů. Následně po dekódování zakódovaného souboru byl soubor porovnán s původním vstupním souborem. Soubory byly totožné, a tak aplikace správně realizuje kódování a dekódování metodou adaptivního Huffmanova kódování.