Projekt do předmětu KKO
Adaptivní Huffmanův kodér/dekodér

Autor: Jordán Jarolím, xjarol03
7. 5. 2017

1 Popis implementovaného algoritmu

Huffmanovo kódování je založeno na sestavení binárního stromu, kde nejkratší cesty po hranách stromu vedou ke znakům s nejvyšší frekvencí. Adaptivní Huffmanovou kódování funguje o stejném základu, není však nutné vstupní text procházet 2x. Naopak stačí pouze jeden jediný průchod a při načtení každého znaku dojde k rekonfiguraci stromu podle četnosti výskytu jednotlivých znaků.

1.1 Komprese dat

Algoritmus pro kompresi dat je poměrně přímočarý, je však nutné implementovat operace pro manipulaci se stromem univerzálně jak pro kompresi, tak pro dekompresi dat. Implementovaný algoritmus může být popsán následovně:

1. Načti symbol
2. Jedná se o první výskyt symbolu ve stromu? Pokud ano, tak krok 3, jinak krok 4
3. Do souboru zapiš cestu ke speciálnímu uzlu NYT a symbol samotný
4. Do souboru zapiš cestu k uzlu obsahujícímu symbol
5. Pokud se nejedná o konec souboru, pokračuj krokem 1

1.2 Dekomprese dat

Dekompresor rekonstruuje strom na základě čtení ze souboru takto:

1. Načti kořen stromu
2. Pokud se nejedná o list stromu, načti další bit a posuň se na levého/pravého syna a opakuj krok 2. Pokud se jedná o list, tak krok 3.
3. Pokud se jedná o list stromu zjisti, jestli se jedná o speciální NYT list. Pokud ano, krok 4. Pokud ne, krok 5.
4. Načti další symbol (s ohledem na pseudo-eof). Pokud se jedná o pseudo-eof, skonči, jinak krok 6.
5. Získej symbol z listu stromu
6. Symbol zapiš do souboru a přidej ho do stromu, následuje krok 6
7. Opakuj od kroku 1

1.3 Operace se stromem

Adaptivní huffmanovo kódování využívá binární strom, kde v každém uzlu jsou uloženy informace: Váha, Pořadí, Symbol, Levý syn, Pravý syn, Otec, kde pořadí slouží jako řadící systém rostoucí zleva doprava a zespodu nahoru a váha vyjadřuje počet výskytů symbolu v textu. Celkově by se vlastnosti správně sestaveného stromu dali shrnout takto: Kadý uzel má sourozence. Uzly s vyšším pořadím mají vyšší váhu. Na každé úrovni stromu jsou uzly řazeny zleva doprava s narůstajícím pořadím, mohou však mít stejné váhy. Listy obsahují symboly, kromě NYT uzlu, který slouží pro přidávání nového symbolu. uzly, které nejsou listy mají jako váhu součet vah svých synů. Všechny uzly se stejnou váhou jsou seřazeny zleva doprava.

Implementovaný algoritmus pro manipulaci a rekofiguraci stromu jde popsat následovně.

- Načti symbol, pokud je již ve stromu, tak krok 3, jinak krok 2
- NYT vytvoří 2 nové listy, které jej budou mít jako otce, vlevo bude nový NYT, vpravo bude list se symbolem
- Updatuj váhy listů, prohoď s pravým sourozencem pokud je potřeba, opakuj dokud nejsi u kořene

2 Testování

Implementovaný algoritmus byl úspěšně testován na testovacím souboru z WISu. Úspěšně proběhla jak komprese tak dekomprese. kompresní poměr je 768771B pro nekomprimovaný text vs 438514B pro komprimovaný text. Komprese testvacího souboru na serveru merlin trvá do 0.5s a dekomprese probíhá v podstatě okamžitě. Dekompresor však bylo nutné optimalizovat - v první implementaci se pro vnitřní reprezentaci požívalo velké množství operací nad vektory, které byly extrémně pomalé a dekomprese trvala v řádu desítek minut. Toto bylo odhaleno nástrojem gprof a tyto operace byli nahrazeny jednoduššími. Nyní dekomprese tedy probíhá téměř okamžitě.

3 Závěr

Byl úspěšne implementován kompresor i dekompresor využívající adaptivní Huffmanovo kódování. Testování na serveru Merlin proběhlo úspěšně. Inspirací algoritmu mi byly především webové stránky http://www.cs.duke.edu/csed/curious/compression/adaptivehuff.html a materiály předmětu KKO.