Adaptivní Huffmanovo kódování

David Durman, xdurma00

23. dubna 2008

1 Úvod

1.1 Historie

Adaptivní Huffmanovo kódování bylo poprvé publikováno nezávisle Fallerem a Gallagerem [Faller 1973, Gallager 1978]. Knuth tento původní algoritmus upravil Knuth [Knuth 1985] a výsledný algoritmus se začal nazývat algoritmus FGK. Nejnovější verze adaptivního Huffmanova kódování byla popsána Vitterem [Vitter 1987].

1.2 Popis

Huffmanovo kódování je založeno na tom, že nejčastěji vyskytující se znaky se zakódují nejkratším kódem. Tato metoda má dvě podoby. Statická, u níž se provedou dva průchody. První provede statistiku výskytů a vypočte prefixové kódy. Ve druhém průchodu se provede samotné zakódování. Adaptivní varianta kódování provádí pouze jeden průchod. Postupným procházaním zdrojových dat přizpůsobuje délku kódů. Dekodér zrcadlí činnost kodéru, musí tedy zůstat synchronizován s kodérem.

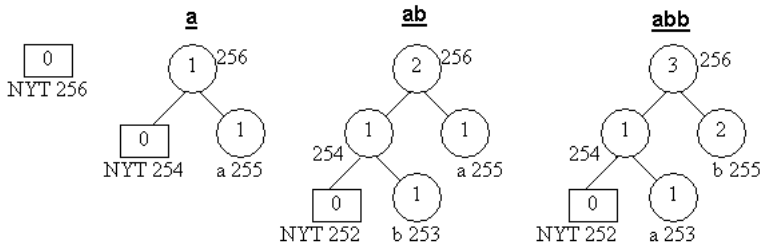
2 Implementace

Pro implementaci Adaptivního Huffmanova kódování jsem použil algoritmus FGK. Základ tohoto algoritmu tvoří tzv. Sourozenecká vlastnost binárního stromu.

Definice. *Binární strom má* ***sourozeneckou vlastnost*** *jestliže každý uzel (s výjimkou kořene) má sourozence a pokud lze uzly seřadit do monotónní posloupnosti (podle četnosti daného uzlu) tak, že má každý uzel v posloupnosti za souseda svého sourozence.*

Gallager potom dokázal, že binární prefixový kód je Huffmanovým kódem právě tehdy, když má odpovídající kódovací strom sourozeneckou vlastnost. Algoritmus pracuje tak, že vkládá symboly do stromu a pokud se detekuje porušení sourozenecké vlastnosti, tak se strom přeuspořádá do té podoby, ve které je sourozenecká vlastnost zachována. Algoritmus má dvě varianty přístupu k abecedám:

- **Inicializace se všemi znaky abecedy:** Na začátku obsahuje strom všechny znaky abecedy se zvolenou pravděpodobností
- **Použití speciálního uzlu ZERO:** Na začátku obsahuje strom pouze uzel ZERO. Pokud přijde symbol, který ještě nebyl zakódován, vypíše se do výstupu kód uzlu ZERO a tento se pak rozdělí na nový uzel ZERO a list s novým znakem.



Obrázek 1: Ukázka vytváření stromu. Uzel ZERO zde představuje uzel NYT(not yet transmitted); [převzato z wikipedie]

Na obrázku 1 je vidět, jak se postupně vytváří Huffmanův strom pro vstup **abb**. Na začátku strom obsahuje jediný uzel ZERO. Po příchodu symbolu **a** se rozdělí na nový uzel ZERO a list se symbolem **a**. Ten má v sobě informaci o frekvenci svého výskytu. Po příchodu uzlu **b** se situace opakuje. Zajímavá je situace po příchodu druhého **b**. Strom ztratil sourozeneckou vlastnost, protože frekvence výskytu symbolu **b** je již 2, to znamená, že se strom musí přeuspořádat na Huffmanův.

Celý postup zachycuje tento **pseudokód**:

```
begin
  create_ZERO_node();
  read( c );
  while ( c != EOF ){
    if ( c is in tree ) then begin
      output_code_for( c );
      actualize_tree( get_node_for(c) );
    end
    else        // první načtení symbolu c
      output_code_for( ZERO );
      output( c );
      u = new_node();
      u->left = ZERO;
      u->right = new_node( c );
      actualize_tree( u );
    end

    read( c );
  end
end
```

3 Závěr

Huffmanův strom jsem implementoval pomocí ukazatelů a zároveň ukládal do pole v pořadí od shora (kořene) dolů zprava doleva. Tímto jsem zajistil snadné vyhledání uzlu, který má větší frekvenci výskytu a nachází se dále ve stromě v opačném než uvedeném pořadí. Maximální velikost tohoto pole je 2*n + 1, kde n je počet různých hodnot vyjádřených 8 bity, tj. 256.
U textových souborů se kompresní poměr pohyboval okolo 0.6.