Алгоритъм на Хъфман  
Курсов проект по Структури от Данни и Програмиране на Борислав Кавалов, 82100

<https://github.com/bobivk/AdaptiveHuffmanCompression>

Алгоритъмът на Хъфман се използва за компресиране на информация без загуба. Има два вида имплементации на този алгоритъм – статична и две адаптивни. Адаптивните се наричат FGK Algorithm и Vitter Algorithm. Статичната имплементация трябва да има на разположение целия файл предварително, за да създаде таблицата на срещанията. Това може да ограничи случаите, в които компресията може да се приложи. Затова в този проект е осъществена имплементацията на Vitter – тя е подобрение на алгоритъма на FGK, предложена от Jeffrey Scott Vitter(1).

Дърво на Хъфман е двоично дърво, в което всяко листо има стойност (символ) тегло, което обозначава колко пъти е бил срещнат символа на листото до този момент. За всеки уникален символ, срещнат до момента, има точно едно листо в дървото.

Алгоритъмът на Витер строи дървото на Хъфман в реално време, четейки информацията символ по символ. Той се дели на три части – закодиране на информацията, четене на информацията и опресняване на дървото. При закодирането и четенето се използва един и същ метод за опресняване на дървото, което позволява преизползването на код.

Основното в този алгоритъм е т.нар. „свойство на братята“:

1. Всеки възел в дървото (освен корена) има „брат“ – възел, който споделя същия родител.
2. Възлите могат да бъдат подредени отляво надясно и отдолу нагоре с ненамаляващи тегла

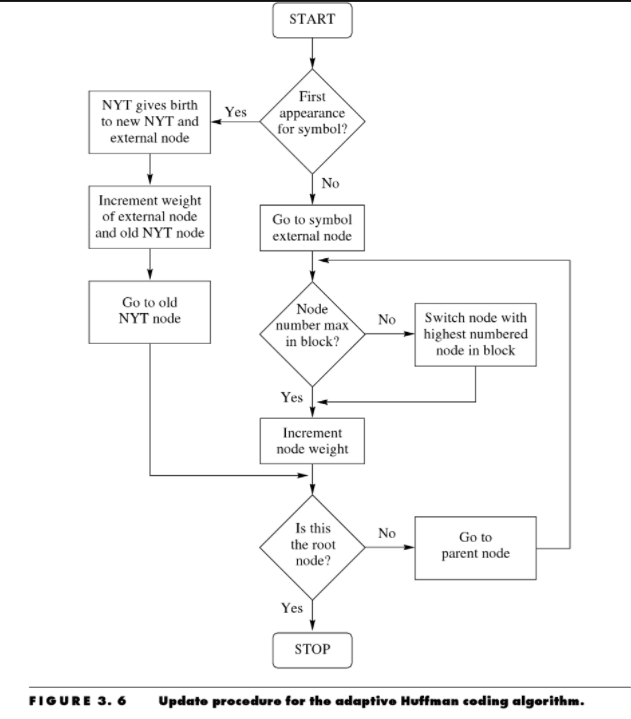
Това свойство трябва да се поддържа във всеки един момент, за да бъде валидно дървото.

Свойството на братята се поддържа чрез помощта на система за подреждане. Всеки възел има свой ред в дървото, като коренът започва от 512 (брой на 8-битовите символи(листа): 256, брой на вътрешните възли: 255 и един NYT възел). Поддържаме реда на възлите ненамаляващ отляво надясно и отдолу нагоре в дървото. Използваме реда и за индексиране на възлите в std::vector за по-бързо изпълнение. Използваме и отделен вектор за листата за оптимизиране на изпълнението откъм време.

Алгоритъмът на Витер също използва т.нар. NYT (Not Yet Transmitted) възел. Той показва къде трябва да сложим ново листо при срещането на нов символ, който не е бил срещан досега. За да построим нов възел, NYT възелът „ражда“ новия NYT възел като своето ляво дете и новото листо с тежест 1 и стойност новия символ като свое дясно дете.

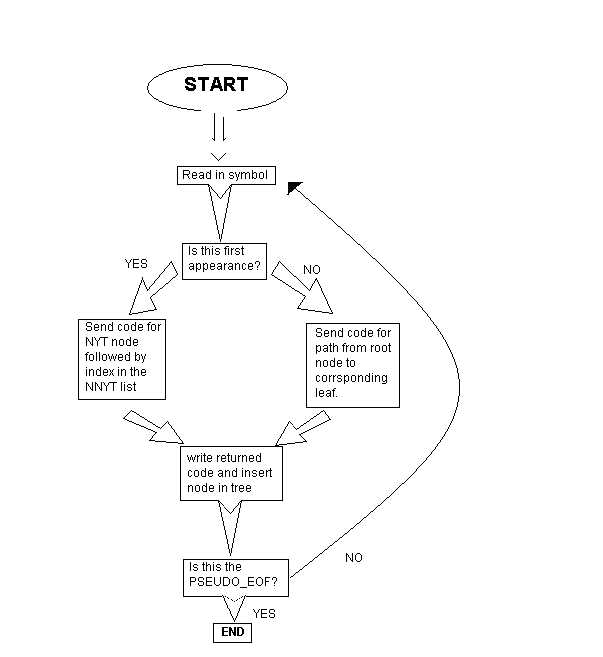
Ако символът е бил срещан досега, увеличаваме теглото на неговото листо и на всеки негов родител нагоре в дървото. Преди да увеличим теглото на негов родител обаче, трябва да проверим дали той с най-високия ред в неговия теглови клас: ако има възел със същото тегло като него и с по-висок ред, той трябва да бъде разменен с него. Така запазваме свойството на братята. При размяната трябва да сме сигурни, че не се опитваме да разменим с директен родител на текущия възел или с корена на дървото, защото това ще наруши свойството на братята.

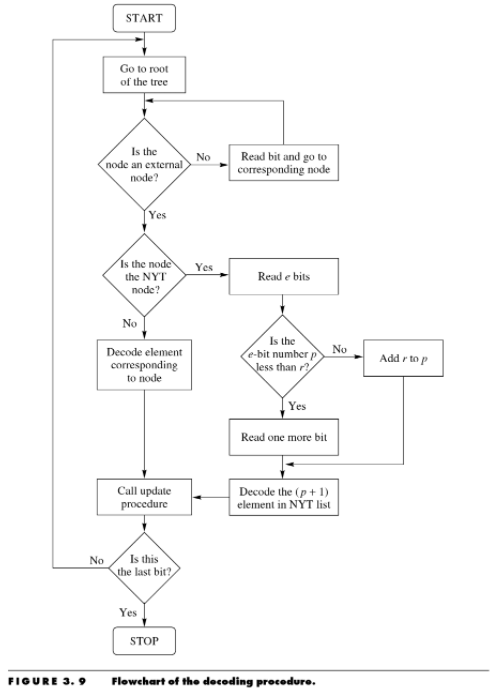
Диаграма на процеса на опресняване на дървото:



Khalid(2), стр.59

Диаграма на процеса на закодиране:

Khalid(2)

Диаграма на процеса на декодиране:

Khalid(2)

Източници:

1. Vitter, Jeffrey Scott , [Design and Analysis of Dynamic Huffman Codes](http://www.ittc.ku.edu/~jsv/Papers/Vit87.jacmACMversion.pdf), 1987

2. Sayood, Khalid. [Introduction to Data Compression](https://books.google.bg/books?redir_esc=y&id=044wLaqZ8twC&q=vitter#v=onepage&q&f=true). Morgan Kaufmann Publishers, New York, 2000

3. [https://www2.cs.duke.edu/csed/curious/compression/adaptivehuff.html#tree](https://www2.cs.duke.edu/csed/curious/compression/adaptivehuff.html%23tree)

4. [https://www.ics.uci.edu/~dan/pubs/DC-Sec4.html#Sec\_4.2](https://www.ics.uci.edu/~dan/pubs/DC-Sec4.html%23Sec_4.2)