




Хъфманово кодиране

23.01.2015

Диан Тодоров

61670

Софтуерно инженерство



[Въведение](#)

[Milestones](#)

[Анализ на проблема](#)

[Документация](#)

[Представяне на данните](#)

[Помощни Функции](#)

[Кодиране](#)

[Декодиране](#)

[Употреба](#)

[Идеи за бъдещо подобрене](#)

Въведение

Целта на този документ е да обясни реализацията на курсовия проект Хъфманово кодиране, зададен по предмета Функционално програмиране. Всички обяснение се базират на програмния файл `huffman.scm`.

Самото хъфманово кодиране е алгоритъм за компресия на данните без загуби. Разработен е от David A. Huffman по време на докторантурата му във MIT и е публикуван за пръв път през 1952 година.

Milestones

1. Кодиране на съобщение чрез алгоритъма на Хъфман, по зададен предикат за сравнение. Върнатият резултат е наредена двойка от кодираното съобщение и самото дърво.
2. Декодиране по зададени хъфманово дърво и поредица от битове, която връща декодирано съобщение.

Анализ на проблема

Резултатът от изпълнението на хъфмановото кодиране може да се разглежда като таблица. Тя съдържа множеството от участващите в изпратеното съобщение символи, като на всеки един от тях е съпоставена кодировка. Самата таблица се извлича на база на честотен списък за срещанията на отделните символи във входящото съобщение. С помощта на тази таблица лесно се осъществява кодирането и декодирането на съобщението.

Особеното при този алгоритъм за компресия на данните е, че различните символи след провеждането на кодирането ще са с различна дължина. Тоест в едно представяне символът А може да се представи като 01, а символът В като 1010111. За да се осъществи декодирането трябва да се подаде и дървото. За едно съобщение съществуват различни кодировки в зависимост от хъфмановото дърво, което се построи. Всички те са валидни

Най-важно за генерирането на таблицата е генерирането на Хъфманово дърво. За целта се извежда честотен списък, където се пазят броя срещания на всеки от елементите в множеството на участващи символи. То се строи последователно като на всяко стъпка се обединяват в едно дърво двете поддървета с минимални тегла. Този процес продължава до получаване на едно единствено дърво в множеството.

След като хъфмановото дърво е налице лесно може да се генерира таблицата. Най-удобно за това се оказва обхождане в дълбочина до достигане на листо. Когато се удари дъно се добавя новата кодировка в таблица и се продължава напред по тривиалната схема.

Тъй като винаги трябва се вземат 2-та най-малки елемента, дървото ще се поддържа постоянно сортирано по тегло. Тоест всеки път, след кое да е обединение, новополученото поддърво ще се вмъква на правилно място. За целта ще се използва insertion Sort алгоритъм.

Документация

I. Представяне на данните

A. Хъфманово дърво - реализира се чрез двоично дърво

- **Листо.**

Листата се различават от останалите възли в дървото. Те представляват списък от следната наредена тройка. На първо място е етикетът 'leaf', след това е символа и накрая е честотата на срещанията му. С помощта на предиката leaf?, който очаква етикета leaf в главата на подадения обект, се осъществява разпознаването.

Интерфейс за листо:

Конструктор:

```
(define (create-leaf item weight)
```

Проверка за тип:

```
(define (leaf? object)
```

Функции за достъп:

```
(define (leaf-item x) (cadr x)) # връща елемента
```

```
(define (leaf-weight x) (caddr x)) # връща теглото
```

- **Двоичното дърво**

Представлява списък от четири елемента. В това число влизат лявото поддърво, дясното поддърво, обединение на множествата от символите на двете

поддървета и сбора на теглата на поддървета (където ако поддървото е листо, теглото представлява честота на неговото срещания в подаденото на входа съобщение, множеството символи - самият негов символ).

Конструктор:

```
(define (create-tree left-subtree right-subtree)
```

Функции за достъп:

```
(define (left-subtree tree) (car tree)) # ляво поддърво
```

```
(define (right-subtree tree) (cadr tree)) # дясно поддърво
```

```
(define (tree-items tree) # множеството на елементите на  
поддърво
```

```
(define (tree-weight tree) # теглото поддърво
```

В. Съобщение

Съобщението се представя чрез списък от символи.

II. Помощни Функции

Първо трябва да се изгради честотен списък. Това се осъществява чрез комбинация от функции, като основна и най-важна сред тях е frequency-list. Тя генерира желания честотен списък по съобщение и предикат за сравнение, като списъкът се състои от наредени двойки (символ - брой срещания).

```
(define (count-occurrences cmp-op el lst) # брой срещания  
(define (custom-member? cmp-op x lst) # предикат за наличност в  
списък  
(define (remove-duplicates cmp-op lst) # създава множество от мулти  
множеството  
(define (frequency-list cmp-op lst) # връща двойки от символ -  
срещания в съобщението
```

III. Кодиране

А. Построяване на Хъфманово дърво.

Чрез функцията frequency-list получаваме честотен списък. За удобство го подреждаме в растящ ред по броя срещания на символите. Това се прави, защото в алгоритъмът на Хъфман е заложено винаги да се обединяват двете поддървета с минимални тела. Списъкът се поддържа винаги сортиран и така по-лесно достъпват желаните два минимални елемента. Те ще се намират на първа и втора позиция съответно.

Реализирани са два алгоритъма за сортировка - insertion и selection sort. В крайна сметка се оказва по-удобно използването на сортировката чрез вмъкване. При нея имаме функция (insert-in-correct-position x set), която поставя подадения елемент на правилната му позиция. Тя намира употреба и в последствие, когато обединяваме поддърветата.

Чрез функцията (ordered-leaf-set cmp-op lst) се получава сортирано множество от листа.

Самото дърво се строи чрез извикване на (huffman-tree cmp-op lst) . Тя от своя страна опакова рекурсивната функция merge, на която се подава множество от листа f. Рекурсивно се обединяват дървета като се вземат двете дървета с минамални тегла до достигане на множество от 1 дърво.

B. Замяна на символите с техните кодове.

Генерира се таблица със символите и техните кодировки. Това се осъществява като се подава аргумент хъфмановото дърво на функцията all-encodings . Върнатия списък е съставен от двойки от символ - кодировка

```
(define (all-encodings tree)
```

Тази таблица се използва впоследствие за обхождане на подаденото съобщение и подмяна на символите със съответния код.

IV. Декодиране

Функцията по декодирането приема два аргумента - списък с оставащи битове и текуща позиция в дървото. На всеки възел се избира посока наляво или надясно. Това решение се взема от функцията choose-subtree и зависи от стойността на подадения бит (нула - ляво, единица - дясно). Когато се достигне листо, се залепя съответният символ и се продължава рекурсивно изпълнението със следващият бит и корена на дървото.


```
(define (decode bits tree)
```

V. Употреба

- Интерактивно през конзолата

Самите функции са публични могат да се използват самостоятелно през конзолата.

- Четене и писане от файл



При извикването (run) се стартира поредица от процедури, даващи цялостен достъп до функционалността на приложението. Чрез последователни въпроси към потребителя, се определя задачата, която той иска да изпълни, и се връща съответният резултат.

Идеи за бъдещо подобрене

Сортировка може да се смени на Quick Sort в случай, че е необходимо оптимизиране.