Хъфманово кодиране
Проект по ФП

Светослав Богданов, Инф., I гр., ФН: 45657

1 Представяне на дървото на Хъфман.

За представяне на дърво на Хъфман в този проект се използват списъци. То се определя със следната рекурсивна дефиниция:

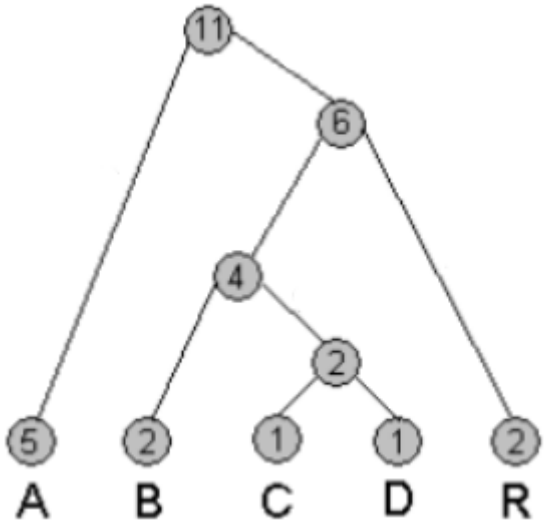
• Ако `count` е положително цяло число, а `symbol` е произволен елемент, то двуелементният списък `(count symbol)` е представяне на дърво на Хъфман, което наричаме *листо*;

• Ако `count` е положително цяло число, а `left` и `right` са представяния на дървета на Хъфман, то списъкът `(count left right)` е представяне на дърво на Хъфман.

Например

```
(11 (5 A) (6 (4 (2 B) (2 (1 C) (1 D))) (1 R)))
```

е представяне на



От сега нататък под „дъво на Хъфман“ се има предвид представяне на такова дърво.

• `ht-count`, `ht-left`, `ht-right` са синоними съответно на `car`, `cadr` и `caddr` и предполагат да се използват за получаване съответно на стойността на корена, лявото поддърво и дясното поддърво. Функцията предполага да ѝ бъде подадено дърво на Хъфман. В противен случай резултатът не е дефиниран.

• `ht-leaf-symbol` е синоним на `cadr` и служи за получаване на символа на дадено листо, като предполага да се ползва само върху листа. В противен случай резултатът не е дефиниран;

• Функцията `cons-ht` приема три параметъра: положително цяло число `count`, две представяния на дървета на Хъфман `left` и `right` и връща представяне на дърво на Хъфман със стойност на корена `count` и ляво и дясно поддърво съответно `left` и `right`;

• Функцията `cons-ht-leaf` приема два параметъра: положително цяло число `count` и произволен елемент `symbol` и връща листо със стойност `count` и символ `symbol`;

• Предикатът `ht-leaf?` приема единствен параметър: дърво на Хъфман `ht` и връща истина тогава и само тогава, когато то е листо. Предикатът предполага, че подаденият му параметър е дърво на Хъфман. В противен случай резултатът не е дефиниран.

2 Кодиране.

Наричаме списъка, който искаме да кодираме `l`, а списъка от уникалните елементи на `l` наричаме `ul`. Идеята е следната: на базата на `l` и `ul` се създава списък, който наричаме `ht-set`. Всеки елемент на този списък е наредена двойка, чийто втори елемент отговаря на елемент на `ul`, а първият елемент е положително цяло число: колко пъти вторият се среща в `l`. Елементите на `ul` са сортирани във възходящ ред по първия си елемент. На практика `ht-set` се явява честотен списък на `l`, но за функциите, показани по-долу, той е представяне на множество от листа. На базата на това множество се построява дървото на Хъфман за `l` по указания в условието алгоритъм, като във всеки нов списък новополученото дърво е на правилното, за да остане този списък сортиран. Накрая на базата на дървото на Хъфман се създава и низът, представящ компресираните данни.

• Функцията `insert-in-ht-set` приема два параметъра: дърво на Хъфман `ht` и списък представящ множество от дървета на Хъман, сортирани във възходящ ред по стойността на корена `ht-set` и връща списък, който се получава от `ht-set`, като `ht` се прибавя към него на правилното място според стойността на корена си така, че списъкът да остане сортиран;

• Функцията `cons-ht-set` приема три параметъра: списък от произволни елементи `l`, списък от елементи `ul`, като функцията предполага, че това е списък състоящ се точно от уникалните елементи на `l`, и предикат за сравняване `pred-eq?` (например `equal?`). Функцията връща списъка `ht-set`, споменат по-горе;

• Функцията `combine-ht` приема два параметъра: дървета на Хъфман `ht1` и `ht2` и създава ново дърво на Хъфман, чиято стойност на корена е сумата от стойностите на корените на `ht1` и `ht2`, лявото поддърво е `ht1`, а дясното – `ht2`. Функцията предполага, че стойността в корена на `ht1` е по-малка или равна от тази на `ht2`;

• Функцията `compress` приема три параметъра: дърво на Хъфман `huffman-tree`, символ (произволен елемент) `symbol` и предикат за сравняване `pred-eq?` и връща списък, чиито елементи са от множеството { `#\0`, `#\1` }, който представя кода на `symbol`. Функцията предполага, че `symbol` се среща в някое листо от `huffman-tree`. В противен случай резултатът не е дефиниран;

• Функцията `compressed-data` приема три параметъра: списък от произволни елементи `l`, дърво на Хъфман `huffman-tree` и предикат за сравняване на елементите `pred-eq?` и връща списък, чиито елементи са от множеството { `#\0`, `#\1` }, който представя компресираните данни;

• Функцията `encode` приема два параметъра: списък `l` и предикат за сравняване на елементите `pred-eq?` и връща наредена двойка, чийто първи елемент е дървото на Хъфман за този списък, а вторият елемент е низ, представящ компресираните данни. Ако списъкът `l` е празен, първият елемент на двойката е празният списък, което според дефиницията не е дърво на Хъфман, но при декодирането то не влияе по никакъв начин, така че го пренебрегваме.

3 Декодиране.

• Функцията `decode-first-element` приема два параметъра: дърво на Хъфман `huffman-tree` и списък `compressed-data-list`, чиито елементи са от множеството { `#\0`, `#\1` } и който представя компресираните данни. Тази функция връща наредена двийка, чийто първи елемент е символът, който първи се среща в `compressed-data-list`, а вторият елемент е списък от останалите елементи на `compressed-data-list`, които трябва да се декодират;

• Функцията `decode-helper` приема два параметъра: дърво на Хъфман `huffman-tree` и списък `compressed-data-list`, чиито елементи са от множеството

{ `#\0`, `#\1` } и който представя компресираните данни. Функцията връща декодираните данни;

• Функцията `decode` приема два параметъра: дърво на Хъфман `huffman-tree` и низ `compressed-data-стринг`, състоящ се само от `#\0` и `#\1` и който представя компресираните данни. Функцията връща декодираните данни;

4 Помощни функции за работа със списъци.

• Предикатът `singleton?` приема единствен параметъ: списък `l` и връща истина тогава и само тогава, когато списъкът `l` се състои от точно един елемент. Този предикат предполага, че подаденият му списък е непразен. В противен случай резултатът не е дефиниран;

• Предикатът `member?` приема три параметъра: произволен елемент `x`, списък от произволни елементи `l` и предикат за сравняване на отделните елементи `pred-eq?` и връща истина тогава и само тогава, когато елементът `x` се среща в `l`;

• Функцията `unique` приема два параметъра: списък от произволни елементи `l` и предикат за сравняване на отделните елементи `pred-eq?` и връща списък, състоящ се само от уникалните елементи на списъка `l`;

• Функцията `count` приема три параметъра: произволен елемент `x`, списък от произволни елементи `l` и предикат за сравняване на отделните елементи `pred-eq?` и връща броя на срещанията на `x` в списъка `l`.