­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Лабораторна робота №1

з курсу «**Проблемно-орієнтовані методи та засоби**

**інформаційних тенологій**»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 10

Виконав студент гр. КНз-3

Чалий Михайло

­­

Львів 2015

## Мета роботи

Отримати практичні навики використання методів оптимального кодування.

## Теоретичні відомості

### Метод Шеннона-Фано

В цьому методі для кожного символа формується бітовий код, довжина якого залежить від частоти появи символа. Чим менша частота, тим довший код. Визначення частоти (ймовірності) символа буває статичне (на основі таблиці даних) або динамічне (коли відомості про ймовірність появи символів визначаються на основі обробки потоку даних).

Кодування здійснюється таким чином (рис. 1):

Всі символи записуються в таблицю по зменшенню їх частоти. Потім вони поділяються на дві групи так, щоб суми частот для отриманих груп були максимально близькі. Для першої групи перший біт коду встановлюється рівним 1, а для другої – 0. Потім групи знову поділяємо на дві і визначаємо наступні розряди коду. Процес продовжується поки в групі не залишиться тільки один символ.

Номер Символ Частота Код

1 a 10 11

2 b 8 10

-----------------------------‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑----------------------------

3 c 6 011

4 d 5 010

5 e 4 001

6 f 3 000

Рис. 1.

Можливий варіант програмної реалізації методу базується на формуванні і обробці такої таблиці

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nгр | Np | Nk | S | L | Cod |
| 1 | 1 | 2 | 18 | 1 | 1 |
| 3 | 6 | 18 | 1 | 0 |
| 2 | 1 | 1 | 10 | 2 | 11 |
| 2 | 2 | 8 | 2 | 10 |
| 3 | 3 | 4 | 11 | 2 | 01 |
| 5 | 6 | 7 | 2 | 00 |
| 4 | 3 | 3 | 6 | 3 | 011 |
| 4 | 4 | 5 | 3 | 010 |
| 5 | 5 | 5 | 4 | 3 | 001 |
| 6 | 6 | 3 | 3 | 000 |

Ця таблиця забезпечує зручний запис алгоритму поділу на підгрупи і формування кодів. Перша група (Nгр=1) складається з двох підгруп: перша - починається з першого символа (Np=1) і закінчується другим (Nк=2), друга – починається з третього символа (Nр=3) і закінчується шостим (Nк=6). Сума частот першої підгрупи S=18, другої S=18. Друга група (Nгр=2) формується в результаті поділу першої підгрупи з першої групи і складається теж з двох підгруп: перша – починається з першого символа (Nр=1) і ним закінчується (Nк=1), друга – починається другим символом (Nр=2) і ним закінчується (Nк=2). Третя група описує процес поділу другої підгрупи з першої групи. Процуес продовжується до тих пір поки кожна підгрупа не буде складатися тільки з одного символа (Nр=Nк). Відповідний новий біт коду кожної групи визначається таким чином: для першої підгрупи він встановлюється рівним одиниці, а для другої підгрупи – нулю.

Кодування Шеннона-Фано неоднозначне. В залежності від варіанту поділу на групи (при однаковій різниці частот між ними) можуть бути отримані різні коди для символів (рис. 2).

Символ Частота Код Символ Частота Код

с 22 11 с 22 11

e 20 101 e 20 10

h 16 100 --------------------------------------------

----------------------------------------------- h 16 011

i 16 011 і 16 010

a 10 010 a 10 001

k 10 001 k 10 0001

m 4 0001 m 4 00001

b 2 0000 b 2 00000

Рис. 2.

### Метод Хаффмана.

Метод полягає в побудові кодового дерева Хаффмана, положення символа на якому визначається частотою (ймовірністю) його появи. Реалізація методу здійснюється по таких кроках:

* Всім символам ставиться у відповідність одна з вершин дерева.
* Об’єднуємо дві вершини з мінімальними частотами (вершина з більшою частотою зліва, а з меншою – справа) і для нової вершини вказуємо сумарну частоту.
* Переходимо на пункт 2, доки не об’єднаємо всі вершини.
* Обходимо дерево зверху і визначаємо розряди коду по такому правилу: перехід вліво – розряд =1, перехід вправо – розряд = 0 (рис.3).

Для реалізації методу можна використати таку таблицю

c 22 22 22 26 32 42 58 100 01

e 20 20 20 22 26 32 42 00

h 16 16 16 20 22 26 111

l 16 16 16 16 20 110

a 10 10 16 16 100

k 10 10 10 1011

m 4 6 10101

b 2 10110

В цій таблиці сумуванням двох останніх елементів стовпця визначається новий елемент, який вставляється у відповідне місце нового стовпця. Сумування продовжується доки не будуть просумовані всі елементи. Отриманий результуючий елемент відповідає кореневій вершині дерева.

## Завдання

1. Ознайомтесь з методами побудови оптимальних кодiв.
2. Для заданого варіанту вручну побудуйте коди методом Шеннона-Фано і методом Хаффмана.
3. Розробiть алгоритм та реалiзуйте програму побудови оптимальних кодiв методом Шеннона-Фано та Хаффмана.
4. Перевiрте роботоздатнiсть програм на тестових прикладах.

|  |  |
| --- | --- |
| Варіант | 10 |
| Частота a | 25 |
| Частота b | 20 |
| Частота c | 15 |
| Частота d | 8 |
| Частота e | 7 |
| Частота f | 7 |
| Частота g | 3 |

## Реалізація

Лістінг 1. shannon\_fano.js

/\* jshint esnext:true \*/

function buildTree(abcFreq){

var rec = function(abc){

if (abc.length === 1){

return abc[0];

}

if (abc.length === 0){

return null;

}

var sum = abc.reduce(function(p,c){

return p + abcFreq[c];

}, 0);

var separatorIndex = 0;

var leftSum = 0;

var prevDiff = sum;

for(var i = 0; i < abc.length; i ++){

leftSum += abcFreq[abc[i]];

var newDiff = Math.abs(leftSum - (sum - leftSum));

if (newDiff > prevDiff){

separatorIndex = i;

break;

}

prevDiff = newDiff;

}

var left = abc.slice(0, separatorIndex);

var right = abc.slice(separatorIndex);

return {

l: rec(left),

r: rec(right)

};

};

var abc = Object.keys(abcFreq)

.sort(function(k1, k2){

return abcFreq[k2] - abcFreq[k1];

});

return rec(abc);

}

function buildCodeTable(tree){

var code\_table = {};

var rec = function(currentCode, node){

if (node){

if (typeof(node) === 'string'){

code\_table[node] = currentCode;

return;

}

rec(currentCode + '0', node.l);

rec(currentCode + '1', node.r);

}

};

rec('', tree);

return code\_table;

}

module.exports.buildCodeTable = function(abcFreq){

var tree = buildTree(abcFreq);

return buildCodeTable(tree);

};

Лістінг 2. huffman.js

/\* jshint esnext:true \*/

function buildSortedAbc(abc\_freq){

return Object.keys(abc\_freq)

.sort(function(k1, k2){

return abc\_freq[k2] - abc\_freq[k1];

});

}

function buildTree(abcFreq){

var queue = Object.keys(abcFreq)

.map(function(k){

return {

sym: k,

code: '',

weight: abcFreq[k]

};

});

var sortByPriority = function(){

queue.sort(function(a, b){

return b.weight - a.weight;

});

};

while(queue.length > 1) {

sortByPriority();

var right = queue.pop();

var left = queue.pop();

queue.push({

right: right,

left: left,

code: '',

weight: right.weight + left.weight

});

}

return queue[0];

}

function buildCodeTable(tree){

var codeTable = {};

var rec = function(code, node){

if (node){

if (node.sym){

codeTable[node.sym] = code;

} else {

rec('1' + code, node.left);

rec('0' + code, node.right);

}

}

};

rec('', tree);

return codeTable;

}

module.exports.buildCodeTable = function(abcFreq){

var tree = buildTree(abcFreq);

return buildCodeTable(tree);

};

Лістінг 2. l1.js

var abcFreq = {

a: 25,

b: 20,

c: 15,

d: 8,

e: 7,

f: 7,

g: 3

};

var sf = require('./lib/shannon\_fano');

var codeTable = sf.buildCodeTable(abcFreq);

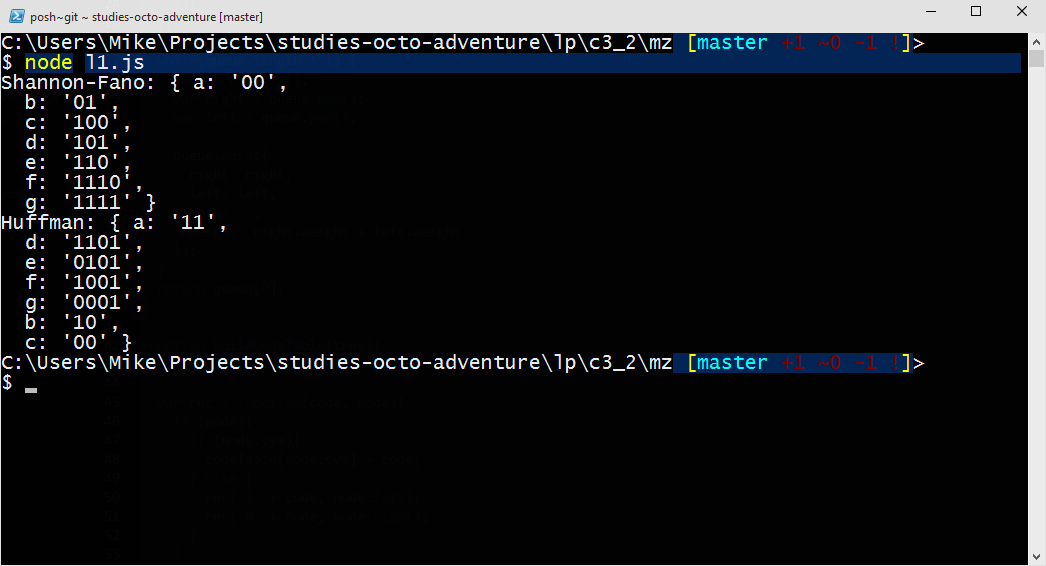
console.log('Shannon-Fano:', codeTable);

var hu = require('./lib/huffman');

var codeTable = hu.buildCodeTable(abcFreq);

console.log('Huffman:', codeTable);

## Результат



## Висновки

Отримав практичні навики використання методів оптимального кодування.