**Проект по Разпределени ИТ Архитектури:**

**ТЕМА НА ПРОЕКТА**

**Алгоритъм на Хъфман за компресия на данни –**

 **честотни таблици**

**Автори:**

Александър Петров Арабаджиев, СУ “Св. Климент Охридски”,ФМИ, гр.София, Информационни Системи II курс,ф.н. 71315

Петя Огнянова Георгиева, СУ “Св. Климент Охридски”,ФМИ, гр.София, Информационни Системи II курс,ф.н. 71302

**Научен ръководител:**

Гл. ас. Христо Христов, катедра Компютърна информатика,СУ “Св. Климент Охридски”,ФМИ,гр. София

1. **Въведение.**

Не е нова идеята да се предава информация по възможно най-икономичен начин. За оптимално кодиране са разработени специални системи, каквито са например стенографската, морзовата азбука, азбуката за глухи, които са лишени от доста удобства. С навлизането на компютрите се появява възможност автоматично сравнително бързо да се "превежда" даден поток от информация на по - икономична

азбука и обратно. Бързо намират приложение алгоритмите за компресиране на информация, а те от своя страна се доразработват и оптимизират, за да навлязат във всекидневна употреба.

1. **Описание и решаване на задачата.**

Алгоритъмът на Хъфман е прост универсален алгоритъм за компресия без загуба на данни. При него се предполага, че е даден краен поток от числа в някакъв предварително фиксиран интервал. Ще считаме, че става дума за символи, кодирани със ASCII код, т.е. ще разглеждаме информацията като поредица от байтове (числа в интервала 0..255). Алгоритъмът се базира на простата идея, че най-често срещаните символи в поредицата трябва да се записват с най-малък брой битове. Така той построява нова азбука, която следва тази идея и след това превежда информацията в новата азбука. Кодирането е обратимо, тоест по кодираната последователност може да се декомпресира - да се намери първоначалната поредица.

Нека трябва да компресираме даден низ от символи. Въвеждаме в потребителският интерфейс името на файла, от който ще прочетем низа и броя нишки, на които искаме да работи задачата. Ако въвеждаме данните в command pront (cmd) въвеждаме името на файла, интервал, брой нишки.

* Първо определяме честотната таблица – броя срещания на всеки символ от низа.

Пример: Имаме низа "ABRACADABRA". Честотната таблица за низа е:

Символ: Брой срещания:

A 5

B 2

C 1

D 1

R 2

 Ще намерим честотната таблица използвайки паралелно обхождане на файла. Това предоставя възможност за оптимизиране и по-бърза работа на задачата. Отваряме файла, за да разберем големината на низа и разделяме обхождането на файла на някакъв брой подзадачи. Разделянето се състои в това да зададем на всички нишки да обхождат равни (или приблизително равни) по големина части на файла. Всяка нишка отваря файла и започва да чете символите, като брои срещанията на всеки символ. Когато задачата работи в „quiet” режим се извежда времето за изпълнение на цялата задача.

* Второ построяваме дървото на Хъфман – искаме да построим двоично дърво,

от което щи определим азбука за компресиране. Построяването на дървото се състои от следните стъпки:

1. Създава се честотна таблица на низа - за всеки символ се записва броят на

срещанията му.

1. Нека различните символи в низа са *n* на брой. Създаваме *n* дървета от по един

елемент, където всяко дърво съдържа символ и броя на срещанията му.

1. Намираме двете дървета, които в корените имат най-малко число. Обединяваме

дърветата в ново дърво, като в корена записваме сумата от стойностите в двете намерени дървета.

1. Повтаряме стъпка 3, докато не получим само едно дърво - дървото на Хъфман за

дадения низ.

Построеното дърво е двоично и е с точно *n* на брой листа, като на всяко листо отговаря един символ от честотната таблица. По начина на построение се вижда, че по-често срещаните символи се намират по-близо до корена от по-рядко срещаните.

* Построяваме азбуката по дървото на Хъфман, като съпоставяме на всеки

клон 0 или 1 – 0 отговаря на ляв клон, а 1 отговаря на десен клон.



 Така получаваме таблица за кодиране и извършваме кодирането на низа, като всеки символ заместваме с неговия код. По този начин получаваме последователност от 0 и 1.

1. **Анализ на резултатите**

Приложението, което построява честотна таблица е разработено на JAVA и е тествано на Intel Core 2 Quad Q6600 - 4 ядрен процесор, 2.4 GHz и оценява времето за изпълнение, ускорението (отношението на времети при използване на 1 нишка към времето за използване на p нишки) и ефективността (ускорението при използване на р нишки, разделено на р). Т.е.

Време – T(p)

Ускорение – S(p) = T(1) / T(p)

Ефективност – E(p) = S(p) / p

Големината на тестовият файл е 1,3 MB и съдържа множество различни символи.

Работа на 1 нишка:

* T(1) = 11987 ms

Работа на 2 нишка:

* T(2) = 8146 ms
* S(2) = 1,4714 ms
* E(2) = 0,7357 ms

Работа на 3 нишка:

* T(3) = 6271 ms
* S(3) = 1,91149 ms
* E(3) = 0,6371 ms

Работа на 4 нишка:

* T(4) = 5023 ms
* S(4) = 2,386422 ms
* E(4) = 0,596605 ms