Алгоритъм на Хъфман документация

Компресацията по алгоритъма на Хъфман се разделя на 5 задачи :

1. Създаване на честотна таблица.
2. Създаване на дърво на Хъфман по честотна таблица.
3. Създаване на кодираща таблица.
4. Кодиране.
5. Разкодиране.

Всяка задача се решава от клас, който след това предава своя резултат на друг клас.Така се създава класова йерархия която решава цялостния проблем на компресация и декомпресация на текстов файл или низ. За решение на по малките задачи са използвани структури от данни, предимно от стандартни STL класове.

КЛАСОВА ЙЕРАРХИЯ

Class AlphabetMap

typedef pair<char, int> upair – дефинира конкретен pair за удобсво.
typedef list<upair>::iterator uiterator – дефинира конкретен итератор за удобсво.

ЧЛЕН ДАННИ

list<upair> map –честотния списък.

КОНСТРУКТОРИ

Класът има два конструктора – по подразбиране и с параметър string. Конструкторът с параметър извиква функцията **mapString(string)** за да намери честотната таблица на низа. Ако не се подаде string при конструктора, а се извика по подразбиране, може да бъде кодиран низ със същия метод след инициализация.

PUBLIC МЕТОДИ

Методът **void mapString(string inputString)** създава честотна таблица на низа, като минава през всеки символ и извиква помощтната private функция **void push(char character).**
Методите **uiterator begin()** и **uiterator end()** се използват за обхождане на списъка от външни класове.

PRIVATE МЕТОДИ

Методът **void push(char character)** е единствения private метод. Той прибавя единица към броя срещания на даден символ в списъка или създава нов **upair** в списъка с честота 1.

Class HuffNode

ЧЛЕН ДАННИ

int frequency – честотата на върха.
HuffNode* left , right - левия и десния наследник.
char character – символа който съдържа (празния ако не е листо).

КОНСТРУКТОРИ

Класът има два конструктора – с честота и символ или с честота и ляв и десен наследник.

МЕТОДИ

Класът има един оператор **bool operator<(HuffNode const& B) const** който сравнява два върха по тяхната честота (по-приоритетни са върховете които са листа).

Class Comparison

Този клас се използва само за сравнение при приоритетна опашка от **HuffNode.**

Class HuffTree

Този клас представлява дървото на Хъфман. Той съдържа само корена на дървото, което редставлява указател към HuffNode. Коренът може да бъде достъпен със селектор.

ЧЛЕН ДАННИ

HuffNode* treeRoot – корена на дървото.

КОНСТРУКТОРИ

Класът има само един конструктор с параметър **string** . Конструкторът извиква помощните private метод **buildTree** с параметър резултата от друга помощна private функция **buildMap.**

МЕТОДИ

public HuffNode* root() –връща корена на дървото.
private AlphabetMap buildMap(string input String) - създава честотен списък на даден низ и го връща.
private void buidTree(AlphabetMap) – създава дървото по честотен списък, което след това се присвоява от treeRoot.

Class Compressor

Класът се използва за компресиране на файлове по даден път към тях или директно на низове.

ЧЛЕН ДАННИ

list<cpair> charMap – кодиращ списък на файла или низа който се компресира.
string filePath – пътя на файла в който ще се запише компресираната информация.

КОНСТРУКТОРИ

Класът има един конструктор с параметър string към файла в който ще се записва информацията.

МЕТОДИ

void compressString(string)
void compressFile(string)

PRIVATE МЕТОДИ

string mapString(string) – създава кодиращата таблица на даден низ като използва дърво на хъфман и функцията **recursiveSearch** върху корена на дървото.

string findCharacter(char in) – намира символ в кодиращата таблица и връща кодиращия низ от нули и единици.
string convertTree(HuffTree) – конвертира дърво в последователност от символи в string. Дървото се записва като пълно двоично дърво, като празните места се записват с '\0'.
void writeToFile(string, int, string) – записва кодираната информация във файла подаден от конструктора. Първо записва колко бита е компресираната информация, след това конвертираното дърво на Хъфман и самата информация (с разделител символа '\a').
void recursiveSearch(HuffNode, string) – рекурсивно търси за листата на дърво на Хъфман и запазва техния кодиращ низ. Когато достигне листо записва символа в листото и кодиращия низ в кодиращата таблица.
string toBitString(string) – превръща низ в низ от нули и единици.

Class Decompressor

ЧЛЕН ДАННИ

string huffTree – конвертирано дърво на Хъфман прочетено от файл.
string filePath – пътя към файла където ще бъде записана разкодираната информация.
int numberOfBits – броя битове които трябва да се разкодират.

КОНСТРУКТОРИ

Класът има един конструктор по път към файл където да бъде записана разкодираната информация.

МЕТОДИ

void decompressFile(string) - разкомпресира файл по даден път към него.

PRIVATE МЕТОДИ

string decompressString(queue<uchar>) –разкомпресира низ подаден като опашка от uchar.
void writeToFile(string) – записва получения разкомпресиран низ във файла подаден на конструктора.

Class CommandPrompt

Класът представлява конзола която обработва подадените команди от потребителя. Съдържа Compressor и Decompressor и MODE (режим) на работа.

ЧЛЕН ДАННИ

Compressor **cs –** инстанция компресорът.
Decompressor **ds** – инстанция на декомпресорът.
bool done – – булева стойност която показва, че потребителя иска да спре да излезе от програмата. Става true когато потребителя напише quit в конзолата.
MODE **currentMode –** конкретния режим на работа.

КОНСТРУКТОР

Само един конструктор по подразбиране. Инициализира **cs** и **ds** със default пътища. Default режима е COMPRESS.

МЕТОДИ

void work() - главният цикъл на програмата, тук се извличат инструкциите и параметрите ред по ред и се подават на съответната функция.
void executeOP(string) –извършва операцията над подадения вход.

ПОДОБРЕНИЯ

Идеи за подобрения на тази имплементация са :

1. Да се намери начин за записване на кодиран двоичен файл. В този вид това няма как да стане тъй като дървото на хъфман се записва като пълно двоично в низ и празните върхове са символът ‘\0’. Ако двоичният файл съдържа байт със стойност 0, ще се получат грешки при записването на дървото на Хъфман.
2. Намиране на по-ефективен начин за изтриване на файлове.