Алгоритъм на Хъфман

Изготвил: Алкан Мустафа, фн: 72018, Информационни Системи, 3 курс, 1 група

Алгоритъмът на Хъфман е сравнително прост универсален алгоритъм за компресия без загуба на данни. Базира се на простата идея, че най-често срещаните символи в поредицата трябва да се записват с възможно най-малък брой битове. Така той построява нова азбука, която следва тази идея и след това превежда информацията в новата азбука. Кодирането е обратимо, т.е кодираната последователност може да се декомпресира – да се намери първоначалната поредица.

Реализацията на алгоритъма се състои от следните класове: class HTree, class Comparator, class Encoder, class Decoder, class System, class BitSetHelper(използва се само на едно място) и class Menu.

1. Class HTree

Класът HTree се състои от следните член-данни и методи: Text

Description automatically generated

Има една единствена член-данна: указател към корена на дървото. Класът има дефолтен конструктор, конструктор с параметри, конструктор за копиране с два параметъра(два други HTree) и съответно с един параметър(друг HTree), оператор= и деструктор. Методът copy(Node\*& current, Node\* other) се използва в копи конструкторите и в оператор=, а методът clear(Node\*& current) се използва в деструктора. getCnt()const връща общия брой на срещанията на всички букви в съответното дърво(ако root == nullptr, връща 0). getCharacterCode(char searched) връща в кодиран вид търсения символ(ако дървото не съдържа търсения символ, се връща празен низ), getCharacterCodeHelper метода се използва тук. printByLevels()const печата възлите на дървото по нива. Методът isLeaf(Node\* current)const се използва в методите getCharacterCodeHelper и printByLevels()const, проверява дали подадения възел е листо.

1. Class Comparator

Text

Description automatically generated with medium confidence

Използва се, за да може да се построи приоритетна опашка от HTree.

1. Class Encoder

Състои се от следните член-данни и методи: Text

Description automatically generated

Член-данните са: масив с размер 128(за символите от ASCII таблицата от 0-127, използва се за честотна таблица), приоритетна опашка(за построяване на дърво на Хъфман), дърво на хъфман(крайното дърво) и таблицата за кодиране. Методите constructHistogram построява честотната таблица, а методът clearHistogram я занулява(след края на целия процес). ConstructHuffmanTree() и constructKeyTable() съответно построяват дървото на Хъфман и таблицата за кодиране. Всички тези методи се изполват в encodeMessage(const std::string& message), който връща като резултат кодираното съобщение, а методът getKeyTable() връща таблицата за кодиране.

1. Class Decoder

Класът се състои от следните член-данни и методи: Text

Description automatically generated

Класът има една единствена член-данна: keyTable(текущата таблица за кодиране). Методът setKeyTable присвоява на keyTable новата стойност от newKeyTable(ако подадената таблица за кодиране е празна, се хвърля изключение). Методът decodeMessage декодира подаденото съобщение въз основа на текущата таблица за кодиране(ако подаденото съобщение е празно се връща празен низ).

1. Class System

Класът се състои от следните член-данни и методи:

Text

Description automatically generated with medium confidence

Член данните на класа са: енкодер(за енкодиране на текущото съобщение), декоред(за декодиране на текущото съобщение, със съответната таблица за кодиране), два низа(един за името на входния файл и един за името на изходния файл, с които програмата ще работи), таблица за кодиране, една променлива за режим на работа(0 – за компресия, 1 – за декомпресия, 2 – в debug режим и -1 – за не настроен режим) и една булева променлива за това дали ще се работи с битове или стандартна компресия/декомпресия(използва се само в доктестовете). Методът clearMessageAndTable() изтрива текущото съобщение и текущата таблица за кодиране(извиква се след като програмата приключи работа или ако изникне проблем), doesExist(const std::string& fileName) проверява дали съществува файл с такова име, checkOpenInputFile и checkOpenOutputFile проверяват дали файловете в са отворени успешно в съответните режими. Методът readFile чете целия файл символ по символ и го запазва в член-данната message, а методът readKeyTable чете файла и запазва резултата в член-данната keyTable. WriteToFileCompress(), readFromFileCompress, пишат във файл и четат от файл в режим на компресия. Съответно writeToFileDecompress() и readFromFileDecompress() пишат във файл и четат от файл в режим на декомпресия. Методите writeToBits() и readFromBits() пишат във файл и четат от файл в режим на компресия/декомпресия, но по малко по различен начин. PrintToScreen() печата резултата на екрана(използва се в debug режима). Методът compress() извиква readFromFileCompress() и writeToFileCompress() или writeToBits() в зависимост от стойността на член-данната workWithBits. Decompress() извиква readFromFileDecompress() или readFromBits() пак в зависимост от стойността на член-данната и writeToFileDecompress(). Debug() извиква методите readFromFileCompress() и printToScreen(), т.е чете файла в режим на компресия и печата компресирания резултат на екрана, както и степента на компресия. В методите compress(), decompress() и debug() се хвърлят изключения и съответно се хващат изключения, ако методите стигнат своя край стойностите message и keyTable се изтриват(ако се хвърли изключение, също се изтриват). Следващите методи са setters & getters съответно за workWithBits, за workMode, за InputFileName, за OutputFileName. Последният метод start() стартира програмата в зависимост от това какъв режим е избран, ако не е избран някакъв режим(workMode == -1), тогава се хвърля изключение.

1. Class Menu

Класът Menu се състои от следните член-данни и методи: Text

Description automatically generated

Член-данните са: system, където е логиката за работата с файлове и като цяло за програмата, currentCommand за текущата въведена команда, commandNames и commandExplanations са масиви с размер 10, които пазят в себе си командите и техните обяснения, които програмата поддържа. Методите loadCommands() и printCommands() съответно зареждат имената на командите и техните обяснения в съответните масиви. ValidateCommand валидира текущата въведена команда, а после в commandCases() се решава кой метод да се извика. SetCompressionMode(), setDecompressionMode() и setDebugMode() съответно дават стойности на член-данната workMode на system(съответно 0, 1, 2). SetInputFile, setOutputFile дават стойности на inputFileName и outputFileName на system. Mode() и files() печатат съответно в какъв режим работи програмата и как се казва входния и изходния файл на програмата. Start() извиква методът start() на system. Единствения публичен метод run() стартира цялото CLI приложение.

1. Стартиране на приложението

Text

Description automatically generated

След стартиране на приложението ни се печатат всички команди, които приложението поддържа(само веднъж се печатат освен ако изрично не се въведе командата help, която служи точно за повторно печатане на всички команди).Като въведем съответно „c“, „d“ или „debug“ поставяме приложението в режим на работа

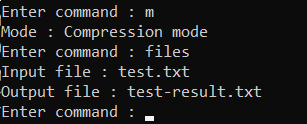
Text

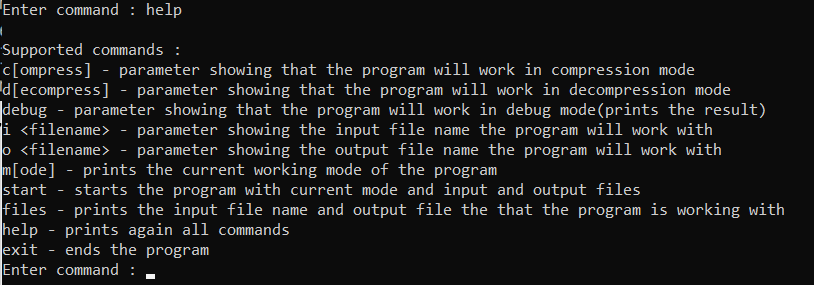
Description automatically generated

С командите „i (име на файл)“ и „o (име на файл)“ избираме имената на файловете, с които ще работи нашата програма Text

Description automatically generated

Съответно командите „m“ и „files“ печатат в какъв режим и с кои файлове работи програмата



Командата „help“ печата повторно всички команди 

Последната команда „exit“ прекратява изпълнението на програмата Text

Description automatically generated

1. Примерна работа в режим Компресия

Text

Description automatically generated

Слагаме в режим компресия, въвеждаме имената на файловете и въвеждаме старт. Това е съобщението на входния файл:

Graphical user interface, application

Description automatically generated

Това е енкодираното съобщение в изходния файл: Graphical user interface, application

Description automatically generated

И това е таблицата за кодиране в другия създаден изходен файл: A screenshot of a computer

Description automatically generated with medium confidence

1. Примерна работа в режим Декомпресия

Text

Description automatically generated

Слагаме програмата в режим декомпресия, въвеждаме име на входния файл и на изходния файл и пишем старт.

Енкодираното съобщение във входния файл: Graphical user interface

Description automatically generated

И таблицата за кодиране на входния файл:

A screenshot of a computer screen

Description automatically generated with medium confidence

И декодираното съобщение в изходния файл:

Graphical user interface, application

Description automatically generated

10. Примерна работа в режим Debug

Text

Description automatically generated

Слагаме програмата на режим debug, въвеждаме име на входен файл(име на изходен файл не е нужен, защото резултатът се изписва на екрана и не се запазва във файл). На екрана се печата кодираното съобщение, разделено на блокове от по 8 бита и след това конвертирано в число, изписва се разметър на оригиналното съобщение, размерът на компресираното съобщение и степента на компресия.

11. Построяване на дърво на Хъфман

Ако искаме да кодираме низа „ABRACADABRA“, то дървото на Хъфман по въведения алгоритъм ще изглежда по този начин: Diagram, shape

Description automatically generated

Където листата са изобразени по този начин: (брой срещания) / символ. Ако отпечатаме елементите на дървото по нива, то резултатът ще е такъв: Text

Description automatically generated

Съответно кодираното съобщение ще има вида: Graphical user interface, application

Description automatically generated

А таблицата за кодиране ще изглежда така: A screenshot of a computer

Description automatically generated with medium confidence

12. Class BitSetHelper

Graphical user interface, text, application

Description automatically generated

Класът BitSetHelper има само един метод, който приема като параметър(enteredNum и number), тоест представя числото number в бинарен вид с размерност enteredNum

13. WorkWithBits Компресиране и Декомпресиране

При това компресиране, ако пак имаме входен файл: Graphical user interface, text, application

Description automatically generated

Файлът с компресираното съобщение ще изглежда във вида: Graphical user interface, text, application

Description automatically generated

Тоест енкодираното съобщение от 0 и 1 е разделено на блокове от по 8 бита и са превърнати в десетичното им представяне.

Таблицата за кодиране е същата като при нормалната компресия, само че преди таблицата се записва и размерът на последния блок от 8 бита(дали последният блок е допълнен до 8 бита или не), ако е допълнен, то просто тези 8 бита се превръщат в блок от съответния размер(тук се използва класът BitSetHelper):

Text

Description automatically generated

Съответно декомпресията чете енкодираните числа от входния файл, чете размера на последния блок и чете таблицата за кодиране от съответния файл и декодира съобщението в изходен файл.