**Софийски университет**

**''Св. Климент Охридски''**

**Проект**

Тема: Алгоритъм на Хъфман за компресия на данни

Студент:

*Кристиан Лалев*

- https://github.com/kris-lalev/ProjectHuffmanCoding

СОФИЯ

2021

Увод

* 1. **Описание и идея на проекта**

Алгоритъмът на Хъфман, разгледан тук, е сравнително прост универсален алгоритъм за компресия без загуба на данни (за разлика от алгоритмите със загуба, стоящи в основата на MPEG, например). При него се предполага, че е даден краен поток от числа в някакъв предварително фиксиран интервал. В рамките на настоящия проект ще разглеждаме информацията като поредица от байтове (числа в интервала [0; 255]) като ще считаме, че символите са кодирани с ASCII код. Алгоритъмът се базира 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.

* 1. **Цел и задачи на разработката**

Целта на проекта е да се имплементира алгоритъма „Алгоритъмът на Хъфман“ описан в горната точка. Ще се използва езика C++, като ще се реализират и използват различни структури от данни.

Проектиране

Създава се структурата struct huffman\_node ,която представя възлите в двоичното дървото на Хъфман. В нея се съдържат

* пойнтъри към левия и десния подвъзел(huffman\_node\* left;huffman\_node\* right;)
* char id – буквата на възела, ако е листо или нищо, ако е междинен възел
* int freq – честотата на срещане на буквата, ако е листо или сбора на freq на листат, ако е междинен възел
* string code; - кодирането на бъквата
* huffman\_node() – констуктор

Създава се класа class huffman ,който ще реализира основната функционалност на програмата. В него се съдържат следните функции и член данни

* node\_ptr node\_array[256]; -масив, чрез който запазваме честотна таблица
* std::unordered\_map<char, node\_ptr> letters; -
* fstream in\_file, out\_file;
* node\_ptr child, parent, root – възли, нужни за създаване и запазване на дървото в програмата
* char id – временна променлива за четене/писане във файл
* string in\_file\_name, out\_file\_name; -имена на подадените файлове

class compare – клас, чрез който сравняване два обекта от тип node\_ptr

* priority\_queue<node\_ptr, vector<node\_ptr>, compare> pq; - опашка, в която седят наредени всички дървета, генерирани от честотната таблица
* void create\_node\_array(); - запазваме честотна таблица
* void traverse(node\_ptr, string); - задаваме код на възела
* int binary\_to\_decimal(const string&); - от двоично число в дума
* string decimal\_to\_binary(int); - от дума в двоично число
* inline void build\_tree(string&, char); - построява дърво от файл
* huffman(string, string); констуктор – приема два файла – входен и изходен
* Функции за компресиране:
  + void create\_pq(); създава опашката
  + void create\_huffman\_tree(); - създава дървото
  + void calculate\_huffman\_codes(); - задаваме кодове на дървото
  + void coding\_save(); -запазваме във файл
* Функции за декомпресиране:
  + void decoding\_save(); - четем от файл и създаваме оригинала
  + void recreate\_huffman\_tree();четем от файл и създаваме оригиналното дърво
* Допълнителни функции:
  + double get\_comp\_rate(string, string);
  + void see\_comp\_input();
  + void coding\_show();

Заключение

Бяха постигнати целите на задачата. За бъдеще алгоритъма може да се направи адаптивен.