Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»

Текстовый поиск (Булев поиск)

Студент: А. О. Дубинин
Преподаватель: А. А. Журавлёв
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2019

Курсовая работа

Задание: Необходимо реализовать программу, выполняющую поиск слов в статьях википедии.

Программа должна работать в двух режимах:

- Для обработки файла в котором будет происходит поиск.
- Для поиска в обработанном файле.

Доступны логические операции &(логическое и), |(логическое или), ∼(логическое не) для связывания слов в запросах.

Вывод программы на запрос может существовать в двух вариантах, либо программа выводит количество статей попадающих под запрос, либо количество и название всех статей в которых запрос был найден.

1 Описание

Реализация курсового проекта сводится к следующим задачам:

1. Изучить теоретическую часть.

2. Реализовать структуру данных инвертированный индекс для данных википедии. Закодировать данные в simple9 кодировку и сохранить структуру в файл.

3. Реализовать парсер для обработки выражения запроса. При выполнении логического выражения обеспечить корректное выполнение поиска по данным.

2 Теоретическая часть

Инвертированный индекс – структура данных, в которой для каждого слова множества документов в соответствующем списке перечислены все документы, в которых оно встретилось. Инвертированный индекс используется для поиска по текстам. В данной программе необходим инвертированный индекс, содержащий только список документов для каждого слова, но так же существует реализация, в которой помимо номера документа хранится позиция слова в документе. Так же используются дополнительные метки, которые применяются в дальнейшем при ранжировании. Преимущество инвертированного индекса состоит в том, что благодаря ему возможно быстро найти документы в которых встречается наши слова.

Simple9 – метод кодирующий целочисленные значения. Для сжатия данных используются машинные слова из 32 бит, 4 бита под управление и 28 под хранимые данные. Обычные числа хранятся в 32 битном слове, но множество битов не используются для хранения числа. Например число 7 использует только 3 первых бита, остальные 29 битов остаются неиспользуемыми, на этом и основан метод Simple9. То есть 9 чисел которые используют 3 бита могут хранится в одном слове simple9, один бит останется неиспользуемым. И так далее для всех чисел до 2^{28}.

Алгоритм сортировочной станции – способ разбора математических выражений, представленных в инфиксной нотации. Может быть использован для получения вывода в виде обратной польской нотации или в виде абстрактного синтаксического дерева. Алгоритм изобретен Эдсгером Дейкстрой и назван им «алгоритм сортировочной станции», поскольку напоминает действие железнодорожной сортировочной станции. Благодаря этому методу можно перевести перевести из инфиксной записи в постфиксную, а из постфиксной легко посчитать выражение. Этот алгоритм можно модифицировать и сразу считать выражение по мере разбора.

3 Реализация

Индексирование файла

Программа парсит данные википедии. Прежде чем обработать текст статьи программа сохраняет его название и дает этому названию уникальный идентификатор(id). Обрабатывая слово статьи программа добавляет id в инвертированные индекс с этим словом. При накоплении 28 номеров в одном слове, программа кодирует эти 28 номеров методом simple9. По окончанию парсинга программа сбрасывает таблицу номеров - названий статей и инвертированный индекс в файл.

Формат файлов для индексации:

```
<doc id="12" url="https://en.wikipedia.org/wiki?curid=12" title="Anarchism">
<Text>
</doc>
<doc id="25" url="https://en.wikipedia.org/wiki?curid=25" title="Autism">
<Text>
</doc>
```

Пример запуска программы для индексации:

```
./prog index --input <input file> --output <index file>
```

Поиск слов

Прежде всего загружаются данные из индексированого файла в инвертированный индекс. Поиск слов происходит обработкой файла с запросами. Выражение запроса обрабатывается модифицированным алгоритмом сортировочной станции. При вычислении выражения программа берет слова из этого выражения, достает номера статей в которых встречается это слово благодаря инвертированному индексу и выполняет выражение благодаря пересечению, объедению и отрицанию множеств. Результат запроса записывается в файл. Если опция –full-output не указана: на каждый запрос в отдельной строке выводится количество документов подпадающих под запрос. Если опция –full-output указана: на каждый запрос выводится отдельная строка, с количеством документов подпадающих под запрос, а затем названия всех документов подпадающих под запрос по одному названию в строке.

Формат файла с запросами

```
<word 1>
<word 1> & <word 2>
<word 1> | <word 2>
~<word 1>
~(<word 1> & <word 2> & <word 3>) | (<word 4> & (<word 5> | ~<word 6>))
```

Пример запуска программы для поиска:

```
./prog search --index <index file> --input <input file> --output <output file> [--full
    -output]
```

4 Пример работы программы

Данные:

```
<doc id="19004" url="https://en.wikipedia.org/wiki?curid=19004" title="Moscow">
Moscow

Moscow ( or ; Russian: Москва , "Moskva" ) is the capital and the
largest city of Russia with 12.2 million residents within the city
limits and 16.8 million within the urban area. Moscow is one of two
federal cities in Russia (the other is Saint Petersburg; Sevastopol
also has this status, although it is disputed due to the annexation
of Crimea by Russia).
</doc>
<doc id="1952253" url="https://en.wikipedia.org/wiki?curid=1952253"
title="Touhou Project">
Touhou Project

The Touhou Project (Project, Tōhō Purojekuto, lit. Eastern
Project) , also known as Toho Project or Project Shrine Maiden, is a
Japanese dōjin game series focused on bullet hell shooters made by
the one-man developer Team Shanghai Alice, whose sole member, known
as ZUN, is responsible for all the graphics, music, and programming
for the most part. The series was inducted into the Guinness World
Records in October 2010 for being the "most prolific fan-made
shooter series". The "Touhou Project" began in 1995 when "Highly
Responsive to Prayers" was first developed by the group Amusement
Makers for the Japanese NEC PC-9801 series of computers; the game
was later released in November 1996. The next four "Touhou" games
released between August 1997 and December 1998 also were released
on the NEC PC-9801. The "Touhou Project" was inactive for the next
three and half years until the first Microsoft Windows "Touhou"
game, "The Embodiment of Scarlet Devil", was released in August
2002 solely by ZUN after he split from Amusement Makers and started
Team Shanghai Alice. The "Touhou Project" became a media franchise
spanning a steadily increasing number of official games, in
addition to commercial fan books, music, light novels, and manga.
```

```
</doc>
```

Запросы:

```
asdfasdxcvaegerzsdwefw
москва
Project
russia | touhou
russia & touhou
~russia
```

Вывод программы:

С опцией –full_output:

```
0
1
Moscow
1
Touhou Project
2
Moscow
Touhou Project
0
1
Touhou Project
```

Без опции –full_output:

```
0
1
1
2
0
1
```

5 Выводы

Выполнив курсовой проект по курсу «Дискретный анализ», я приобрёл практические и теоретические навыки. Узнал о том, как работает поиск, а в частности инвертированный индекс, как работает целочисленная кодировка simple9.

Парсинг данных был интересен с точки зрения языка c++, я узнал как нужно считывать данные в формате unicode. Интересен был и метод кодирования, когда данные номера десятков статей я могу хранить в нескольких числах(машинных словах).

Во время разбора выражения запроса пришлось вспомнить, как корректно считать математическое выражение со скобочками и логическими операциями. Так же я узнал что существуют set_intersection, set_union, set_difference в языке c++, с помощью которых можно выполнить логические операции для контейнеров.

Реализация поиска была очень интересным проектом, я получил ценные знания и опыт в процессе.

6 Исходный код

main.cpp

```cpp
#include "TInvertedIndex.h"
#include <ctime>
#include <cstring>


int main(int argc, char **argv) {
    //unsigned int start_time = clock();

    std::locale::global(std::locale("en_US.UTF-8"));


    if (argc < 2) {
        std::cout << "Arguments error: missing name of action\n";
        return 0;
    }

    if (strcmp(argv[1], "index") == 0) {

        if (argc < 6) {
            std::cout << "Arguments error: wrong input or output arguments\n";
            return 0;
        }

        std::wifstream input_file;
        std::wofstream output_file;

        for (int i = 2; i < argc; i += 2) {
            if (strcmp(argv[i], "--input") == 0) {
                if (argc - 1 == i) {
                    std::cout << "Argument error\n";
                    return 1;
                }
                std::string input_file_name(argv[i + 1]);
                input_file.open(input_file_name);
                if (!input_file.is_open()) {
                    std::cout << "File error: file doesn't exist\n";
                    return 1;
                }
            } else {
                if (argc - 1 == i) {
                    std::cout << "Argument error\n";
                    return 1;
                }
                std::string output_file_name(argv[5]);
                output_file.open(output_file_name);
                if (!output_file.is_open()) {
```

```
                std::cout << "File error: output file error\n";
            }
        }
    }

    TInvertedIndex iv;
    iv.Read(input_file);
    iv.Save(output_file);

} else if (strcmp(argv[1], "search") == 0) {

    if (argc < 8) {
        std::cout << "Arguments error: wrong input or output arguments\n";
        return 0;
    }

    TInvertedIndex iv;

    std::wifstream index_file;
    std::wifstream input_file;
    std::wofstream output_file;



    for (int i = 2; i < argc; ++i) {
        if (strcmp(argv[i], "--index") == 0) {
            if (argc - 1 == i) {
                std::cout << "Argument error\n";
                return 1;
            }
            ++i;
            std::string index_file_name(argv[i]);
            index_file.open(index_file_name);
            if (!index_file.is_open()) {
                std::cout << "File error: file doesn't exist\n";
            }
        }
        else if (strcmp(argv[i], "--input") == 0) {
            if (argc - 1 == i) {
                std::cout << "Argument error\n";
                return 1;
            }
            ++i;
            std::string input_file_name(argv[i]);
            input_file.open(input_file_name);
            if (!input_file.is_open()) {
                std::cout << "File error: file doesn't exist\n";
            }
        }
```

```
96              else if (strcmp(argv[i], "--output") == 0) {
97                  if (argc - 1 == i) {
98                      std::cout << "Argument error\n";
99                      return 1;
100                 }
101                 ++i;
102                 std::string output_file_name(argv[i]);
103                 output_file.open(output_file_name);
104                 if (!output_file.is_open()) {
105                     std::cout << "File error: output file error\n";
106                 }
107             }
108             else if (strcmp(argv[i], "--full-output") == 0) {
109                 iv.full_output = true;
110             }
111         }
112
113         iv.Load(index_file);
114         iv.ReadQueries(input_file, output_file);
115
116
117     } else {
118         std::cout << "Arguments error: wrong action\n";
119         return 0;
120     }
121
122
123 // unsigned int end_time = clock(); //
124 // unsigned int search_time = end_time - start_time; //
125 // std::cout << search_time << "\n";
126     return 0;
127 }
```

TSimple9.h

```
1
2  #ifndef KP_SIMPLE9_H
3  #define KP_SIMPLE9_H
4
5
6  #include <iostream>
7  #include <vector>
8
9  #include <unordered_map>
10 #include <algorithm>
11 #include <fstream>
12 #include <string>
13 #include <stack>
14 #include <sstream>
15
```

```
16 #include <assert.h>
17 #include <stdint.h>
18 #include <stdlib.h>
19 #include <string.h>
20 #include <stdio.h>
21 #include <errno.h>
22 #include <limits.h>
23
24
25
26 #define SELECTOR_MASK 0x0000000F //15
27
28 #define SELECTOR_BITS 4
29
30 #define NSELECTORS 9
31
32 static const struct {
33     uint32_t nitems;
34     uint32_t nbits;
35     uint32_t nwaste;
36 } selectors[NSELECTORS] = {
37         {28, 1, 0},
38         {14, 2, 0},
39         {9, 3, 1},
40         {7, 4, 0},
41         {5, 5, 3},
42         {4, 7, 0},
43         {3, 9, 1},
44         {2, 14, 0},
45         {1, 28, 0},
46 };
47
48
49 void TSimple9_encode(std::vector<uint32_t> &array, std::vector<uint32_t> &encodeArray)
       ;
50
51 std::vector<uint32_t> TSimple9_decode(std::vector<uint32_t> &vec);
52
53
54 #endif //KP_SIMPLE9_H
```

TSimple9.cpp

```
1
2 #include "TSimple9.h"
3
4 void TSimple9_encode(std::vector<uint32_t> &array, std::vector<uint32_t> &encodeArray)
       {
5
6     uint32_t index;
```

```
7       uint32_t selector;
8       uint32_t data;
9       uint32_t shift;
10
11      size_t nitems;
12      size_t i;
13      size_t arraySize = array.size();
14
15
16      index = 0;
17
18
19      while (index < arraySize) {
20          for (selector = 0; selector < NSELECTORS; selector++) { // choose selector
21              data = selector; // control bits
22              shift = SELECTOR_BITS; // start 4 bits
23              nitems = 0; // count of elements in one uint_32
24
25              for (i = index; i < arraySize; ++i) { // filling selector
26
27                  if (nitems == selectors[selector].nitems) // selector is full
28                      break;
29
30                  // element in array more then max in this selector
31                  if (array[i] > (1UL << selectors[selector].nbits) - 1)
32                      break;
33
34                  // add element in data
35                  // {element 1001} or {element element 1000}
36                  data |= (array[i] << shift);
37
38                  shift += selectors[selector].nbits;
39
40                  nitems++;
41              }
42
43              // selector is full or elements are over
44              if (nitems == selectors[selector].nitems || index + nitems == arraySize) {
45
46                  encodeArray.push_back(data);
47                  index += nitems;
48
49                  break;
50              }
51
52          } /* End for selector ... */
53
54      } /* End while index < n */
55  }
```

```cpp

std::vector<uint32_t> TSimple9_decode(std::vector<uint32_t> &vec) {

    std::vector<uint32_t> ans;
    uint32_t lastElement = 0;
    uint32_t element;

    for (auto selector : vec) { //take selector
        uint32_t typeSelector = selector & SELECTOR_MASK;

        uint32_t countOfElements = selectors[typeSelector].nitems; // count of elements
             in one selector

        uint32_t shift = selectors[typeSelector].nbits;

        uint32_t mask = (1u << selectors[typeSelector].nbits) - 1;

        selector = selector >> SELECTOR_BITS; // skip control bits

        for (int i = 0; i < countOfElements; ++i) {
            element = (selector & mask);
            if (element) {
                if (lastElement != 0) {
                    // remove deltas
                    lastElement += element;
                    ans.push_back(lastElement);
                } else {
                    lastElement = element;
                    ans.push_back(element);
                }
            }
            selector = selector >> shift;
        }
    }
    return ans;
}
```

TInvertedIndex.h

```cpp

#ifndef KP_INVERTEDINDEX_H
#define KP_INVERTEDINDEX_H

#include "TEnc.h"

class TInvertedIndex {
public:

    TInvertedIndex();

```

```
12      void Read(std::wifstream &in);
13      void ReadArticle(std::wifstream &in, uint32_t &numOfArticle);
14      void Load(std::wifstream &in);
15      void Save(std::wofstream &out);
16
17
18      ////////////////
19      ////for queries
20      ////////////////
21      void ReadQueries(std::wifstream &in, std::wofstream &out);
22      void Query(std::wstring &str, std::wofstream &out);
23
24      int Priority(wchar_t &op);
25
26      void Evaluate(std::vector<uint32_t> &val1, wchar_t &op, std::vector<uint32_t> &val2
            , std::vector<uint32_t> &result);
27      void Evaluate(wchar_t &op, std::vector<uint32_t> &val1, std::vector<uint32_t> &
            result);
28
29      void QueryIntersection(std::vector<uint32_t> &lhs, std::vector<uint32_t> &rhs, std
            ::vector<uint32_t> &result);
30      void QueryUnion(std::vector<uint32_t> &lhs, std::vector<uint32_t> &rhs, std::vector
            <uint32_t> &result);
31      void QueryDifference(std::vector<uint32_t> &lhs, std::vector<uint32_t> &rhs, std::
            vector<uint32_t> &result);
32
33      bool full_output;
34  private:
35      // <word> <num of article>
36      std::unordered_map<std::wstring, TEnc> words;
37      // <num of article> <name of article>
38      std::vector<std::wstring> names;
39      std::vector<uint32_t > nums;
40
41
42  };
43
44
45  #endif //KP_INVERTEDINDEX_H
```

TInvertedIndex.cpp

```
1
2
3
4   #include "TInvertedIndex.h"
5
6   TInvertedIndex::TInvertedIndex() : full_output(false) {}
7
8   void TInvertedIndex::Read(std::wifstream &in) {
```

```
9       //find title
10
11      std::wstring str;
12      std::wstring nameOfDoc;
13      uint32_t numOfArticle = 1;
14
15      while (getline(in, str)) {
16          if (!str.empty()) {
17              //take name of doc
18              size_t pos = str.find(L"title");
19              nameOfDoc = str.substr(pos + 7, str.size() - pos - 7 - 2);
20
21              //read doc's inner
22              ReadArticle(in, numOfArticle);
23              numOfArticle++;
24
25              //push name of doc
26              names.push_back(nameOfDoc);
27          }
28      }
29
30  }
31
32  void TInvertedIndex::ReadArticle(std::wifstream &in, uint32_t &numOfArticle) {
33
34      std::wstring str;
35      std::wstring word;
36      while (in >> str) {
37
38          if (str == L"</doc>") {
39              break;
40          }
41
42          //convert the word into a normal form
43          for (auto &c : str) {
44              if (iswalnum(c)) {
45                  word += static_cast<wchar_t >(towlower(c));
46              }
47
48          }
49
50          if (!word.empty())
51              words[word].Push(numOfArticle);
52
53
54          word.clear();
55      }
56  }
57
```

```
58 void TInvertedIndex::Save(std::wofstream &out) {
59     for (auto &i : names) {
60         out << i << L"\n";
61     }
62     out << L"0\n";
63     for (auto &i : words) {
64         out << i.first << L"\n";
65 
66         i.second.PushDelay();
67         i.second.Save(out);
68     }
69     out << L"_";
70 }
71 
72 void TInvertedIndex::Load(std::wifstream &in) {
73     uint32_t numOfArticle = 1;
74     std::wstring str;
75     while (true) {
76         getline(in, str);
77         if (str == L"0")
78             break;
79         names.push_back(str);
80         nums.push_back(numOfArticle);
81         numOfArticle++;
82     }
83     while (true) {
84         in >> str;
85         if (str == L"_")
86             break;
87         words[str].Load(in);
88     }
89 
90 }
91 
92 
93 
94 ////////////////////////////////////
95 ////////////// for queries
96 ////////////////////////////////////
97 
98 int TInvertedIndex::Priority(wchar_t &op) {
99     if (op == L'~') return 3;
100    if (op == L'&') return 2;
101    if (op == L'|') return 1;
102    return 0;
103 }
104 
105 
106 void
```

```
107 TInvertedIndex::QueryIntersection(std::vector<uint32_t> &lhs, std::vector<uint32_t> &
        rhs,
108                               std::vector<uint32_t> &result) {
109
110     std::set_intersection(lhs.begin(), lhs.end(),
111                           rhs.begin(), rhs.end(),
112                           std::back_inserter(result));
113
114 }
115
116 void TInvertedIndex::QueryUnion(std::vector<uint32_t> &lhs, std::vector<uint32_t> &rhs
        , std::vector<uint32_t> &result) {
117
118     std::set_union(lhs.begin(), lhs.end(),
119                    rhs.begin(), rhs.end(),
120                    std::back_inserter(result));
121
122 }
123
124 void TInvertedIndex::QueryDifference(std::vector<uint32_t> &lhs,
125                                      std::vector<uint32_t> &rhs, std::vector<uint32_t> &
                                             result) {
126     std::set_difference(lhs.begin(), lhs.end(),
127                         rhs.begin(), rhs.end(),
128                         std::back_inserter(result));
129 }
130
131 void
132 TInvertedIndex::Evaluate(std::vector<uint32_t> &val1, wchar_t &op, std::vector<
        uint32_t> &val2,
133                          std::vector<uint32_t> &result) {
134
135
136     switch (op) {
137         case L'&':
138             QueryIntersection(val1, val2, result);
139             break;
140         case L'|':
141             QueryUnion(val1, val2, result);
142             break;
143         default:
144             break;
145     }
146 }
147
148 void TInvertedIndex::Evaluate(wchar_t &op, std::vector<uint32_t> &val1, std::vector<
        uint32_t> &result) {
149     QueryDifference(nums, val1, result);
150 }
```

```
151
152 void TInvertedIndex::ReadQueries(std::wifstream &in, std::wofstream &out) {
153     std::wstring str;
154     while (getline(in, str))
155         Query(str, out);
156 }
157
158 void TInvertedIndex::Query(std::wstring &str, std::wofstream &out) {
159
160     std::vector<uint32_t> emptyVec;
161
162     //parse and evaluate expression
163     std::stack<wchar_t> operators;
164     std::stack<std::vector<uint32_t>> values;
165
166     for (int i = 0; i < str.size(); ++i) {
167
168         //skip whitespaces
169         if (str[i] == L' ')
170             continue;
171
172         if (str[i] == L'(') {
173             operators.push(L'(');
174         } else if (iswalnum(str[i])) { // if it word
175             std::wstring word;
176             while (i < str.size() && iswalnum(str[i])) {
177                 word += static_cast<wchar_t >(towlower(str[i]));
178                 i++;
179             }
180             i--;
181             if (words.find(word) != words.end()) {
182                 values.push(words[word].Decode());
183             } else {
184                 values.push(emptyVec);
185             }
186         } else if (str[i] == L')') {
187             while (!operators.empty() && operators.top() != L'(') {
188                 std::vector<uint32_t> tmp;
189                 wchar_t op = operators.top();
190                 operators.pop();
191
192                 if (op == L'~') {
193
194                     Evaluate(op, values.top(), tmp);
195                     tmp.swap(values.top());
196
197                 } else {
198
199                     std::vector<uint32_t> val1 = values.top();
```

```
                values.pop();

                Evaluate(val1, op, values.top(), tmp);
                tmp.swap(values.top());
            }
        }
        operators.pop();

    } else { //else it operator
        wchar_t rightOp = str[i];
        while (!operators.empty() && Priority(operators.top()) >= Priority(rightOp)
            ) {
            std::vector<uint32_t> tmp;
            wchar_t op = operators.top();
            operators.pop();

            if (op == L'~') {
                Evaluate(op, values.top(), tmp);
                tmp.swap(values.top());
            } else {

                std::vector<uint32_t> val1 = values.top();
                values.pop();

                Evaluate(val1, op, values.top(), tmp);
                tmp.swap(values.top());
            }
        }
        operators.push(rightOp);
    }
}

while (!operators.empty()) {
    std::vector<uint32_t> tmp;
    wchar_t op = operators.top();
    operators.pop();

    if (op == L'~') {
        Evaluate(op, values.top(), tmp);
        tmp.swap(values.top());
    } else {
        std::vector<uint32_t> val1 = values.top();
        values.pop();

        Evaluate(val1, op, values.top(), tmp);
        tmp.swap(values.top());
    }
}

```

```
248     //write answer
249     if (!values.empty()) {
250         std::vector<uint32_t> ans = values.top();
251         if (!ans.empty()) {
252             out << ans.size() << L"\n";
253             if (full_output)
254                 for (auto &i : ans) {
255                     out << names[i - 1] << L"\n";
256                 }
257         } else {
258             out << 0 << L"\n";
259         }
260     } else {
261         out << 0 << L"\n";
262     }
263 
264 }
```

TEnc.h

```
1  
2  #ifndef KP_ENC_H
3  #define KP_ENC_H
4  
5  #include "TSimple9.h"
6  
7  class TEnc {
8  public:
9      TEnc() = default;
10 
11 
12     void Push(uint32_t &numOfText);
13     //encode all delays nums
14     void PushDelay();
15 
16     std::vector<uint32_t> Decode();
17 
18     void Save(std::wofstream &out);
19     void Load(std::wifstream &in);
20 
21 
22     std::vector<uint32_t> encoded;
23     std::vector<uint32_t> delay;
24     uint32_t lastNum = 0;
25 };
26 
27 
28 #endif //KP_ENC_H
```

TEnc.cpp

```
1
2  #include <stdint-gcc.h>
3  #include "TEnc.h"
4
5
6  void TEnc::Push(uint32_t &numOfText) {
7      if (lastNum != 0) { // not empty
8          if (lastNum != numOfText) { // no repeat
9              delay.push_back(numOfText - lastNum); // push delta
10             lastNum = numOfText;
11         } else {
12             return;
13         }
14     } else {
15         delay.push_back(numOfText);
16         lastNum = numOfText;
17     }
18
19
20     if (delay.size() == 28) {
21         TSimple9_encode(delay, encoded);
22         delay.clear();
23     }
24 }
25
26 void TEnc::PushDelay() {
27     if (!delay.empty()) {
28         TSimple9_encode(delay, encoded);
29         delay.clear();
30     }
31 }
32
33 std::vector<uint32_t> TEnc::Decode() {
34     if (!delay.empty()) {
35         TSimple9_encode(delay, encoded);
36         delay.clear();
37     }
38     std::vector<uint32_t> tmp = TSimple9_decode(encoded);
39
40     return tmp;
41 }
42
43 void TEnc::Save(std::wofstream &out) {
44
45     for (auto i : this->encoded) {
46         out << i << L"\n";
47     }
48     out << L"0\n";
49 }
```

```
50
51 void TEnc::Load(std::wifstream &in) {
52     uint32_t num;
53     while (true) {
54         in >> num;
55         if(num == 0)
56             break;
57         encoded.push_back(num);
58     }
59 }
```

Makefile

```
1  CC = g++
2  FLAGS = -std=c++11 -O2 -g -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long -lm
3  SOURCES = main.cpp TSimple9.cpp TInvertedIndex.cpp TEnc.cpp
4  EXECUTABLE = main
5  all: build
6
7  build: main.cpp TSimple9.cpp TInvertedIndex.cpp
8    $(CC) $(FLAGS) -o $(EXECUTABLE) $(SOURCES)
9
10 main.o: main.cpp TInvertedIndex.h TSimple9.h
11   $(CC) $(FLAGS) -c main.cpp
12
13 TInvertedIndex.o: TInvertedIndex.cpp TInvertedIndex.h
14   $(CC) $(FLAGS) -c TInvertedIndex.cpp
15
16 TSimple9.o: TSimple9.cpp TSimple9.h
17   $(CC) $(FLAGS) -c TSimple9.cpp
18
19 TEnc.o: TEnc.cpp TEnc.h
20   $(CC) $(FLAGS) -c TEnc.cpp
21
22 clean:
23   rm -f *.o
```