Министерство образования и науки РФ

Новосибирский государственный технический университет

Кафедра систем сбора и обработки данных

**Лабораторная работа №2**

Вариант 15.

Реализация класса «Префиксное дерево (Trie)»

Факультет: АВТФ Преподаватель: Квашнина Е. А.

Группа: АТ-33

Студент: Мельничук В.

Новосибирск, 2025

1. Формулировка задания
2. Спроектировать, реализовать и провести тестовые испытания Префиксного дерева. Элементов в дереве должно быть не менее 10ти.
3. Выполнить сравнительное тестирование трудоёмкости операций вставки, удаления и поиска для средней (случайной) и вырожденной (худший случай) структуры дерева.
4. Описание Префиксного дерева

Структура данных реализована в виде класса Tribios, имитирующего поведение префиксного дерева. Каждый узел дерева хранит:

* префикс name,
* флаг isWord, указывающий, завершает ли данный узел строку,
* ассоциативный контейнер kids для хранения потомков (по символам).

Поддерживаются основные операции:

* insert() — вставка строки;
* seek() — поиск строки;
* print() — вывод всех строк, сохранённых в дереве.

1. Оценка трудоемкости

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Кол-во строк** | **Длина строки** | **Случай** | **Вставка (усл. операции)** | **Поиск (усл. операции)** |
| 10 000 | 10 | Средний | 18 | 11 |
| 10 000 | 1…10 000 ('a') | Худший | 5002 | 5001 |

В среднем случае дерево ветвится равномерно, глубина около 10, операции почти линейные.

В худшем случае дерево превращается в цепочку из узлов, что эквивалентно списку.

1. Анализ трудоёмкости

Теоретическая оценка:

В среднем случае вставка и поиск занимают O(k), где k — длина строки. В худшем случае при последовательной вставке строк, где каждая следующая — префикс предыдущей ("a", "aa", …), дерево становится линейным. Трудоёмкость растёт до O(n).

Экспериментальная оценка:

Полученные значения полностью подтверждают теоретические. Среднее значение операций — около 13-14, в то время как в худшем — близко к 20 000, что соответствует линейной вложенности по числу символов.

1. Листинг программы

**main.cpp**

#include <iostream>

#include <string>

#include <map>

#include <cassert>

#include <chrono>

#include <vector>

#include <random>

class Tribios {

private:

    std::string name;

    bool isWord;

    std::map<char, Tribios\*> kids;

public:

    Tribios() : name(""), isWord(false) {}

    Tribios(const std::string& name) : name(name), isWord(false) {}

// Вставка строки

    void insert(const std::string& key, int\* count = nullptr, size\_t index = 0) {

        if (count) (\*count)++;

        if (index > key.length()) return;

        if (key.empty()) return;

        if (key.length() - index == 0) {

            isWord = true;

            return;

        }

        if (kids.find(key[index]) == kids.end()) {

            if (count) (\*count)++;

            kids[key[index]] = new Tribios(name + key[index]);

        }

        kids[key[index]]->insert(key, count, index + 1);

    }

// Поиск строки

    bool seek(const std::string& key, int\* count = nullptr, size\_t index = 0) {

        if (count) {

            (\*count)++;

        }

        if (index > key.length()) return false;

        if (key.empty()) return true;

        if (key.length() - index == 0) {

            return isWord;

        }

        if (kids.find(key[index]) == kids.end()) {

            return false;

        }

        return kids[key[index]]->seek(key, count, index + 1);

    }

// Вывод всех строк

    void print() {

        if (isWord) std::cout << name << " ";

        for (char c = 'a'; c <= 'z'; c++) {

            if (kids.find(c) != kids.end()) {

                kids[c]->print();

            }

        }

    }

};

// Генерация случайной строки

std::string generateRandomString(size\_t length) {

    static const char charset[] = "abcdefghijklmnopqrstuvwxyz";

    static std::mt19937 rng(std::random\_device{}());

    static std::uniform\_int\_distribution<> dist(0, 25);

    std::string s;

    s.reserve(length);

    for (size\_t i = 0; i < length; ++i)

        s += charset[dist(rng)];

    return s;

}

// Вставка и поиск с подсчётом операций

void benchmark(const std::string& name, std::vector<std::string>& dataset) {

    Tribios t;

    int time = 0;

    // Insertion

    for (const auto& word : dataset)

        t.insert(word, &time);

    std::cout << "[" << name << "] Insertion time: "

              << time / dataset.size() << " operations\n";

    time = 0;

    // Search

    for (const auto& word : dataset)

        assert(t.seek(word, &time));

    std::cout << "[" << name << "] Search time: "

              << time / dataset.size() << " operations\n";

}

int main() {

    Tribios t;

    std::vector<std::string> elems = {"aisubdf", "qnwovtb", "bvksjxhbi", "what", "i", "wow", "wow", "ggwp", "kvuzy", "ziuaewtbci"};

    for (std::string i : elems) {

        t.insert(i);

    }

    std::cout << "Words in Trie:\n";

    t.print();

    std::cout << std::endl;

    std::cout << std::endl;

    const size\_t N = 10000;

    const size\_t len = 10;

    std::vector<std::string> randomWords, worstWords;

    for (size\_t i = 0; i < N; ++i)

        randomWords.push\_back(generateRandomString(len));

    for (size\_t i = 1; i <= N; ++i)

        worstWords.push\_back(std::string(i, 'a'));

    benchmark("Average Case", randomWords);

    benchmark("Worst Case", worstWords);

    return 0;

}