Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Дискретный анализ»

Студент: М. М. Касимов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №2

Задача: Необходимо создать программную библиотеку, реализующую указанную структуру данных, на основе которой разработать программу-словарь. В словаре каждому ключу, представляющему из себя регистронезависимую последовательность букв английского алфавита длиной не более 256 символов, поставлен в соответствие некоторый номер, от 0 до $2^{64}-1$. Разным словам может быть поставлен в соответствие один и тот же номер.

Вариант структуры данных: PATRICIA.

1 Описание

Основная идея PATRICIA в том, что мы избегаем однонаправленного ветвления, как это было в стандартных деревьях цифрового поиска, благодаря применению простого приёма: в каждый узел помещается индекс разряда, который должен проверяться с целью выбора пути из этого узла. Таким образом, мы переходим непосредственно к разряду, в котором должно приниматься важное решение, пропуская сравнения разрядов в узлах, в которых все ключи в поддереве имеют одинаковое значение этого разряда. PATRICIA имеет сложность по времени $O(h)$ для вставки, удаления и поиска, где h - высота дерева.

2 Исходный код

main.cpp	
void Lower(char *str)	Функция, которая переводит все буквы к нижнему регистру.
int main()	Главная функция, в которой происходит чтение данных и работа с нашим деревом.
TPatriciaTree.h	
TPatriciaNode* Search(char* indKey)	Поиск по ключу в PATRICIA.
~ TPatriciaTree()	Деструктор класса TPatriciaTree.
bool Insert(char* insertKey, unsigned long long insertValue)	Добавление нового элемента в PATRICIA.
bool Delete(char* deleteKey)	Удаление элемента по ключу в PATRICIA.
bool Save(std::ofstream*)	Сохранение PATRICIA в файл, чтобы потом мы могли восстановить наше дерево.
bool Load(std::ifstream*);	Собираем PATRICIA из файла.
TPatriciaTree()	Конструктор нашего класса, который создает терминатор.
bool IsEmpty()	Проверка на пустоту дерева.
bool CompareKey(char* key1,char* key2)	Проверка на равенство двух ключей.
int BitGet(char* key,int bit)	Возвращает бит ключа на позиции bit.
void DeleteTree(TPatriciaNode* node)	Рекурсивное удаление всех узлов PATRICIA.
void Index(TPatriciaNode* node, TPatriciaNode** nodes, int* depth)	Индексирование всех узлов дерева для дальнейшей записи на файл.

Класс узла PATRICIA.

```
1  class TPatriciaNode {
2  public:
3      TPatriciaNode(char* newKey, unsigned long long newValue, int newBit);
4      unsigned long long GetTheValue();
5      ~TPatriciaNode();
6      TPatriciaNode();
7  private:
8      char* Key;
9      unsigned long long Value;
10     int Bit;
11     int Index;
12     TPatriciaNode* Pointers[2];
13     friend class TPatriciaTree;
```

```
};
```

Класс самого дерева PATRICIA.

```
class TPatriciaTree {
public:
    TPatriciaTree();
    TPatriciaNode* Search(char* indKey);
    bool Insert(char* insertKey, unsigned long long insertValue);
    bool Delete(char* deleteKey);
    bool IsEmpty();
    bool Save(std::ofstream*);
    bool Load(std::ifstream*);
    ~TPatriciaTree();
    void Print(TPatriciaNode* node,int i);
private:
    TPatriciaNode* Root;
    TPatriciaNode* SearhForDelete(char* key,TPatriciaNode** backPtr);
    TPatriciaNode* Parent(TPatriciaNode* node);
    size_t Size;
    bool CompareKey(char* key1,char* key2);
    int BitLength(char* key);
    int BitGet(char* key,int bit);
    void DeleteTree(TPatriciaNode* node);
    TPatriciaNode* SearchR(TPatriciaNode *node, char *key, int bit);
    void Index(TPatriciaNode* node, TPatriciaNode** nodes, int* depth);
};
```

3 Консоль

```
magomed@magomed-pc ~/da_lab2/da_lab2 $ g++ -std=c++14 main.cpp
magomed@magomed-pc ~/da_lab2/da_lab2 $ vim test
magomed@magomed-pc ~/da_lab2/da_lab2 $ cat test
+ audi 2011
+ BMW 2014
+ LaDa 1999
-mersedes
-lAdA
-bmw
audi
magomed@magomed-pc ~/da_lab2/da_lab2 $ ./
a.out  .git/
magomed@magomed-pc ~/da_lab2/da_lab2 $ ./a.out <test >result
magomed@magomed-pc ~/da_lab2/da_lab2 $ cat result
OK
OK
OK
NoSuchWord
OK
OK
OK: 2011
```

4 Тест производительности

Тест производительности представляет из себя следующее: мы будем сравнивать время выполнения всех добавлений в PATRICIA с библиотечным RB деревом - map. Для этого я сгенерировал тест, с 1000000 строк. А поможет нам в этом библиотека chrono, которая может измеряять время в микросекундах.

```
magomed@magomed-pc ~/da_lab2/da_lab2 $ wc -l test.txt
1000000 test.txt
magomed@magomed-pc ~/da_lab2/da_lab2 $ g++ -std=c++14 insertBenchmark.cpp
magomed@magomed-pc ~/da_lab2/da_lab2 $ head -2 test.txt
sdaimefhyxrgxwnovlipbuxjnbkhedexieiuknflmwsmtgdsumkxhjhuktfoylohrxfcnknajimgrqymfjkost
1049063358
ihvdvsycltbnfpwnojslrktrroyujjdssbxquwsirwvymumdfhoyriskytfjgleyocqjalttipsunixupoviwc
1497172992
magomed@magomed-pc ~/da_lab2/da_lab2 $ ./a.out <test.txt
RB insert: 5.86328 seconds
PATRICIA trie insert: 3.42802 seconds
RB find: 5 microseconds
PATRICIA trie find: 2 microseconds
```

Мы видим, что PATRICIA выиграла по скорости у RB-Tree. Неудивительно, что моя структура данных быстрее чем map, ведь длина ключей была большой, а PATRICIA , в отличие от RB-Tree, проверяет только важные разряды. Справедливости ради, стоит отметить, что если уменьшать длину ключей, RB-Tree начнёт догонять PATRICIA.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился пользоваться стандартной библиотекой chrono. Также я узнал новую структуру данных PATRICIA, которая работает эффективнее других структур на символьных ключах большой длины. Смог лично посмотреть на скорость вставки и поиска PATRICIA и сравнить с RB-Tree. Улучшил навыки отладки своей программы. Стоит отметить, что PATRICIA довольна экзотическая структура данных, я нашёл лишь две книги, где говорилось о ней.

Список литературы

[1] Роберт Седжвик. *Фундаментальные алгоритмы, 3-я редакция.* — Издательский дом «ДиаСофт», 2001. Перевод с английского: С. Н. Козлов, Ю. Н. Артеменко, О. А. Шадрин. — 688 с. (ISBN 966-7393-89-5 (рус.))

[2] *Лекции по курсу «Дискретный анализ» МАИ.*