Лабораторная работа №2 по курсу дискретного анализа: сбалансированные деревья.

Выполнил студент группы М80-208Б-20 Зубко Дмитрий Валерьевич

Условие

Кратко описывается 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. Программа должна обрабатывать строки входного файла до его окончания. PATRICIA.

Метод решения

Чтобы реализовать словарь, я реализовал структуру данных PATRICIA. PATRICIA расшифровывается как Practical Algorithm To Retrieve Information Coded In Alphanumeric. Каждый узел в PATRICIA содержит номер бита (индекс), который проверяется при выборе пути из узла, и две ссылки. Ссылки бывают обратными и прямыми. Прямая ссылка ведет в узел с большим индексом, обратная – с меньшим или равным. Самый первый узел дерева – header. Его правая ссылка всегда ведёт сама в себя, индекс равен -1, то есть из него всегда идём в левый узел. Сложность поиска, удаления и добавления элементов – O(k), где h – длина ключа.

Описание программы

Разделение по файлам, описание основных типов данных и функций.

main.cpp – реализация интерфейса программы

patricia.hpp и patricia.cpp – реализация дерева

Patricia();

~Patricia();

bool Empty();

```
    Node *GetHeader();
   Node *FindElement(const std::string &key);
    bool AddElement(const std::string &new_key, const unsigned long long &new_value);
    bool DeleteElement(const std::string &key);
    bool SaveToFile(Node *node, std::ofstream &oStream);
    bool LoadFromFile(std::istream &iStream);
    void Clear();
private:
    bool FirstCaseDeletion();
    bool SecondCaseDeletion(Node *parentParentDeleteNode, Node *deleteNode);
    bool ThirdCaseDeletion(Node *p, Node *q, Node *m, Node *r, Node *deleteNode);
    void DeleteNode(Node *node);
    Node *header;
```

node.hpp и node.cpp – реализация узлов

```
   Node(const std::string &new_key, const unsigned long long &new_value, int new_index);
    ~Node();
    void SetLeftNode(Node *newLeftNode);
    void SetRightNode(Node *newRightNode);
    void SetKey(const std::string &new_key);
```

```
    void SetValue(const unsigned long long &new_value);

  void SetIndex(int new_index);

    Node *GetLeftNode();

    Node *GetRightNode();

    [[nodiscard]] std::string GetKey() const;

    [[nodiscard]] unsigned long long GetValue() const;

    [[nodiscard]] int GetIndex() const;

private:

    std::string key;

    unsigned long long value;

    int index;

    Node *right;

    Node *left;
```

bits.h и bits.cpp – операции с битами

```
bool BitNumberIndexIsOne (const std::string& key, int index);

int FirstDifferenceInBit (const std::string& firstString, const std::string& secondString);
```

Дневник отладки

Что и когда делали, что не работало, как чинили. Этот пункт обязательный, если если было сделано несколько посылок. Кратко опишите суть проблемы и способ ее устранения.

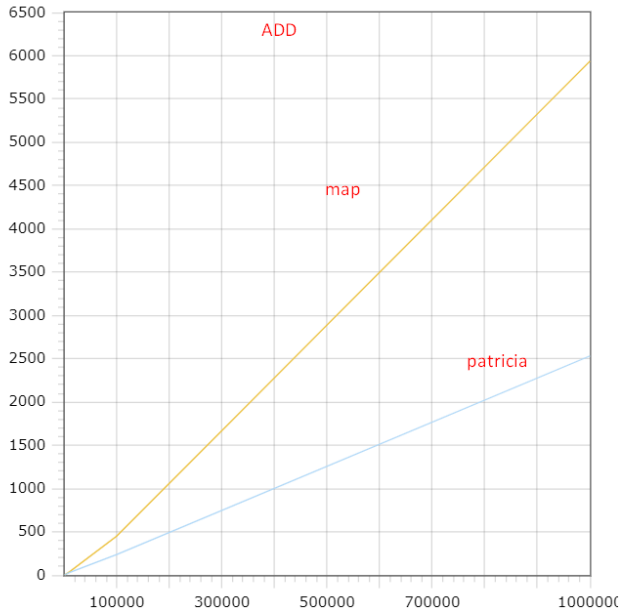
Изначально программа не работала из-за неправильного удаления в 3 кейсе. Ошибка оказалась в том, что я неправильно искал родителя родителя удаляемого элемента. Далее я получал segmentation fault при загрузке дерева. Это удалось починить рассчитав правильно размер считываемых данных. После исправления этих ошибок программа работала правильно, но не проходила по времени на чекере. Это удалось исправить, написав

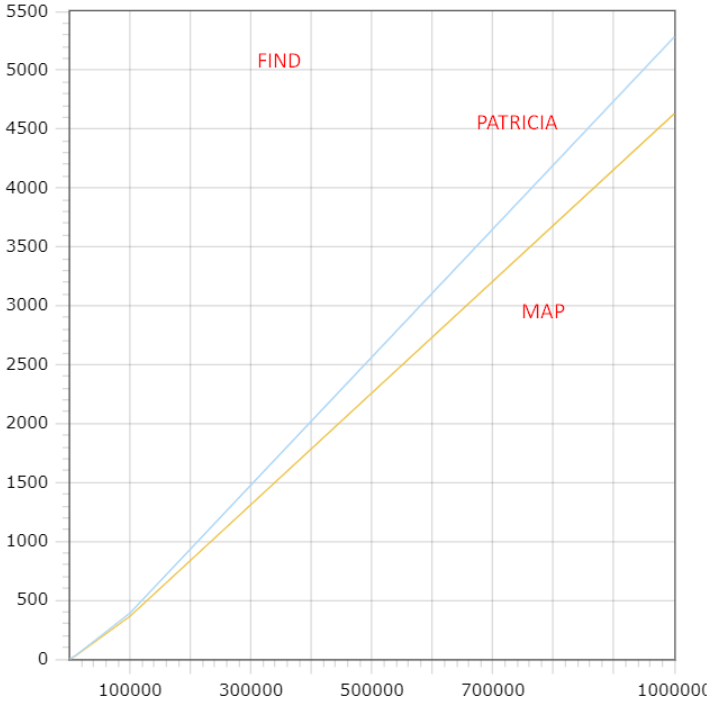
```
std::ios::sync_with_stdio(false);
std::cin.tie(nullptr);
std::cout.tie(nullptr).
```

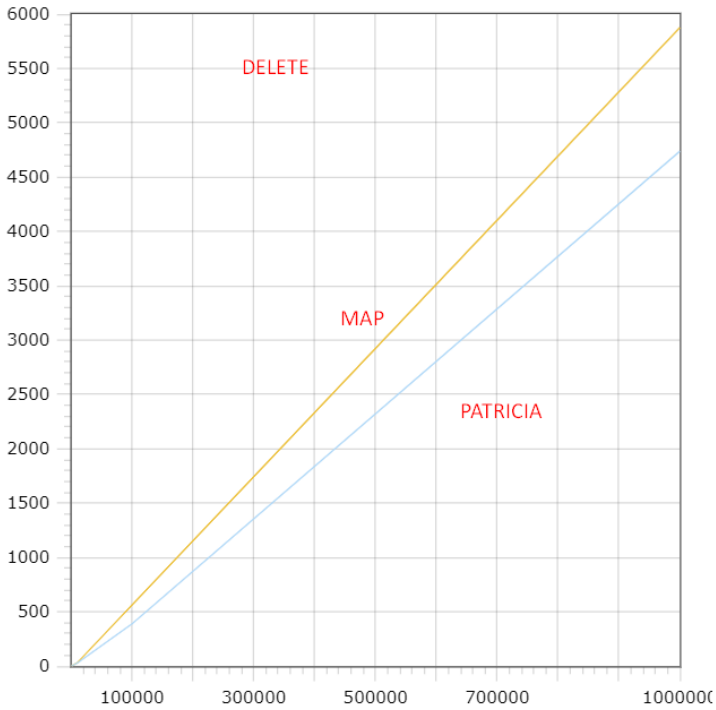
Программа ускорилась в 100 раз.

Тест производительности

Померить время работы кода лабораторной и теста производительности на разных объемах входных данных. Сравнить результаты. Проверить, что рост времени работы при увеличении объема входных данных согласуется с заявленной сложностью.




5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
FIND
PATRICIA
MAP
100000
300000
500000
700000


6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
DELETE
MAP
PATRICIA
100000
300000
500000
700000

Недочёты

Стоит немного декомпозировать код.

Выводы

Благодаря данной лабораторной работе я познакомился и реализовал структуру данных PATRICIA. Узнал о структуре данных trie, где в узлах хранятся не ключи, а символы. PATRICIA может применятся в любых задачах, где есть длинный ключ и часто происходит поиск элементов. Её можно использовать, например, в T9 или в подсказках при поиске в google.