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: Д.В. Семенов
Преподаватель: А.А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №1

Задача: Необходимо создать программную библиотеку, реализующую указанную красно-черное дерево поиска, на основе которой разработать программу-словарь. В словаре каждому ключу, представляющею 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 сопоставлен в соответствие один и тот же номер.

Вариант дерева: Красно-черное.

1 Описание

Программа должна обрабатывать строки входного файла до его окончания. Каждая строка может иметь следующий формат:

1. + word 34 - добавить слово «word» с номером 34 в словарь. Программа должна вывести строку «OK», если операция прошла успешно, «Exist», если слово уже находится в словаре.

2. - word — удалить слово «word» из словаря. Программа должна вывести «OK», если слово существовало и было удалено, «NoSuchWord», если слово в словаре не было найдено.

3. word — найти в словаре слово «word». Программа должна вывести «OK: 34», если слово было найдено; число, которое следует за «OK:» — номер, присвоенный слову при добавлении. В случае, если слово в словаре не было обнаружено, нужно вывести строку «NoSuchWord».

4. ! Save /path/to/file — сохранить словарь в бинарном компактном представлении на диск в файл, указанный параметром команды. В случае успеха, программа должна вывести «OK», в случае неудачи выполнения операции, программа должна вывести описание ошибки (см. ниже).

5. ! Load /path/to/file — загрузить словарь из файла. Предполагается, что файл был ранее подготовлен при помощи команды Save. В случае успеха, программа должна вывести строку «OK», а загруженный словарь должен заменить текущий (с которым происходит работа); в случае неуспеха, должна быть выведена диагностика, а рабочий словарь должен остаться без изменений. Кроме системных ошибок, программа должна корректно обрабатывать случаи несовпадения формата указанного файла и представления данных словаря во внешнем файле.

Для всех операций, в случае возникновения системной ошибки (нехватка памяти, отсутствие прав на запись и т.п.), программа должна вывести строку, начинающуюуся с «ERROR:» и описывающую на английском языке возникшую ошибку.

2 Исходный код

Основные этапы написания кода:

1. Считывание исходных данных
2. Определение операции
3. Выполнение операции
4. Вывод результата

Для хранения значений мы используем класс Node, который и является элементом дерева. В нем мы определяем $unsignedlonglongmvalue$ и $char * mkey$, где и храним значения.

После вставки элемента в само дерева, вызывается процедура балансировки, которая перестраивает дерево в соответствии с правилами построения красно-черного дерева.

При удалении вызывается аналогичная процедура.

Балансировка достигается за счет использования функций поворота узлов дерева.

```
1
2
3  typedef enum { BLACK, RED } nodeColor;
4
5  class Node {
6      public:
7          nodeColor color; // enumed color
8          Node *left;
9          Node *right;
10         Node *parent;
11         unsigned long long m_value = 0;
12         char *m_key = nullptr;
13
14         Node(char *key, unsigned long long value){
15             int i = 0;
16             if(key != nullptr){
17                 int len_key = 0;
18                 len_key = strlen(key);
19                 m_key = (char*)malloc((len_key+1) * sizeof(char));
20                 for(i = 0; i < len_key; i++){
21                     m_key[i] = key[i];
22                 }
23             }
24             m_key[i] = '\0'; // valgrind lenkey+1
25             m_value = value;
26             color = RED;
27         };
```

```
28
29          Node(nodeColor color, Node *left, Node *right,
30              Node *parent, char *key, unsigned long long value){
31              int i = 0;
32              if(key != nullptr){
33                  int len_key = 0;
34                  len_key = strlen(key);
35                  m_key = (char*)malloc((len_key+1) * sizeof(char));
36                  for(i = 0; i < len_key; i++){
37                      m_key[i] = key[i];
38                  }
39              }
40              m_key[i] = '\0'; // valgrind lenkey+1
41              m_value = value;
42              this->color = color;
43              this->left = left;
44              this->right = right;
45              this->parent = parent;
46          };
47
48          Node(){
49          };
50          ~Node(){
51              free(m_key); // valgrind 1- to free, 2 - in deletehidden
52
53          };
54      };
55
56  void TRBTree::RotateLeft(Node *node) {
57
58      Node *right_child = node->right;
59      node->right = right_child->left;
60
61      if (right_child->left != nil)
62          right_child->left->parent = node;
63
64      if (right_child != nil) right_child->parent = node->parent;
65
66      if (node->parent != nil) {
67          if (node == node->parent->left)
68              node->parent->left = right_child;
69          else
70              node->parent->right = right_child;
71      } else {
72          root = right_child;
73      }
74
75      right_child->left = node;
76      if (node != nil) node->parent = right_child;
```

```
77
78  }
79
80
81  void TRBTree::FixInsertRBTree(Node *node) {
82
83      while (node != root
84          && node->parent->color == RED)
85      {
86
87
88          if (node->parent == node->parent->parent->left) {
89
90              if (node->parent->parent->right->color == RED) {
91                  node->parent->parent->color = RED;
92                  node->parent->parent->right->color = BLACK;
93                  node->parent->parent->left->color = BLACK;
94                  node = node->parent->parent;
95              } else {
96
97                  if (node == node->parent->right){
98                      node = node->parent;
99                      RotateLeft(node);
100                 }
101                 node->parent->color = BLACK;
102                 node->parent->parent->color = RED;
103                 RotateRight(node->parent->parent);
104             }
105         } else {
106
107             if (node->parent->parent->left->color == RED) {
108                 node->parent->parent->color = RED;
109                 node->parent->parent->right->color = BLACK;
110                 node->parent->parent->left->color = BLACK;
111                 node = node->parent->parent;
112             } else {
113                 if (node == node->parent->left){
114                     node = node->parent;
115                     RotateRight(node);
116                 }
117                 node->parent->color = BLACK;
118                 node->parent->parent->color = RED;
119                 RotateLeft(node->parent->parent);
120             }
121         }
122     }
123
124     this->root->color = BLACK;
125 }
```

3 Консоль

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_da/2lab_debug/2lab ./2lab <tests/01.t
>result.txt

denis@denis-Extensa-2510G:~/labs_3sem/labs_da/2lab_debug/2lab diff tests/01.a
result.txt
```

Пример теста 01.t: QPXLHFZWLTFXXZNMZGXSTNJUYKCQIAXRHDVQMSXYOHDIYCRLVF

! Save test

EEZCKZGUEGADTBILUUFUWTDAGUNRYWUUDSHLQRQCDSTUBVEITSPTNGSGAZCODBU

+ MKNULGHFEJSNDKRIAPYEVGWEWWPQPWOGYDNKUNSMGRCKDTHATELLKVOCQTK

3784958133452695920

- QMLDOAXVYZXBXCOBGBOCETJBGDBNYYSOGRDRXDQLQDBLDKINUCPCKSMJUEWWS

+ CHTVMOEKLWTESHKEOOPFWOOZGAPCSDLBEASIBZOCNAKVQHXZWYZEYNGHDFWL

86137190829105695

- ZTEQEQCAMHOBQXUTRWXKEGXTLEVTMVULAEWWRPKCNZXJSGEUGNHEFWNVGXN

- MACYDKBVCXWLPBJWFJGQHOGEQZOSFYWIIGBMCMICAHPFQLSDUSJVFYZWVCUVFI

chtvmoeklwteshkeoopfwoozgapcsdlbeasibzocnakvqhxzwyzeynghdfwlajfvqlwntbrjwlodmizrxxsakecxhboi

! Save test

Полученный ответ 01.a:

NoSuchWord

OK

NoSuchWord

OK

NoSuchWord

OK

NoSuchWord

NoSuchWord

OK: 86137190829105695

OK

4 Тест производительности

```
clock_t start_my = clock();
 /* some work */
clock_t end_my = clock();
std::cout <<"map " << (double)(-(start_my - end_my)) / CLOCKS_PER_SEC << std::endl;
```

Первый результат - результат карманной сортировки. Второй - сортировки вставкой.

Кол-во тестов	map	RBTree
100	0.006953	0.00136
1000	0.023991	0.015864
10000	0.311768	0.12934

Как мы видим, наше красно-черное дерево работает гораздо быстрее.

5 Выводы

Выполнив вторую лабораторную работу по курсу «Дискретный анализ», я научился понимать различные сбалансированные деревья, а особенно красно-черные. Научился процессу разработки программ с использованием своих тестов и дополнительных утилит проверки качества программ (они нередко выручали меня, когда ситуация казалась безвыходной). Понял, что написание эффективных программ требует больших затрат временных ресурсов.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] — *Википедия.*
URL: `https://ru.wikipedia.org/wiki/Красночёпрное_дерево.`