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. В Спиридонов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2017

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 дерева: PATRICIA.

Вариант ключа: Строки от 0 до 256 символов, состоящие из символов английского алфавита.

Вариант значения: Числа от 0 до $2^{64} - 1$

1 Описание

Условие задачи подразумевает хранение словаря «ключ-значение», в структуре данных типа «PATRICIA». Для хранения регистронезависимых строк я отказался от привычного класса «TString», реализующего обертку над «char *», и класса «TPair», реализующего связку «TString, unsigned long long», для максимального повышения производительности, т.к. данная вариантом структура данных подразумевает постоянное изменение какой-то части ключа, как при вставке, так и при сериализации.
Суть алгоритма состоит в построении нагруженного дерева. Нагруженное дерево — структура данных реализующая интерфейс ассоциативного массива, то есть позволяющая хранить пары «ключ-значение». В большинстве случаев ключами выступают строки, однако в качестве ключей можно использовать любые типы данных, представимые как последовательность байт.
Нагруженное дерево отличается от обычных n-арных деревьев тем, что в его узлах не хранятся ключи. Вместо них в узлах хранятся части строк и односимвольные метки, а ключем, который соответствует некоему узлу является путь от корня дерева до этого узла, а точнее строка составленная из меток узлов, повстречавшихся на этом пути. В таком случае корень дерева, очевидно, соответствует пустому ключу.
Так как нагруженное дерево реализует интерфейс ассоциативного массива, в нем можно выделить три основные операции, а именно вставку, удаление и поиск ключа.
Как и многие деревья, нагруженное дерево обладает свойством самоподобия, то есть любое его поддерево также является полноценным нагруженным деревом. Легко заметить, что все ключи в таком поддереве имеют общий префикс, а значит можно выделить специфичную для этого дерева операцию — получение всех ключей дерева с заданным префиксом за время O(Prefix).
Я реализовал сразу создание «сжатого» префиксного дерева. Например, цепочка промежуточных узлов с метками s, m, k заменяется на один узел с меткой smk.

2 Описание программы

1. main.cpp (содержит основной метод main, ToLower - метод, переводящий char* в нижний регистр, LengthString - метод получения длинны char*)
2. PatriciaTree.h и PatriciaTree.cpp (описание и реализация класса TPatriciaTree)
3. StackContainer.h и StackContainer.hpp (описание и реализация шаблона TStack)

Для удобства сборки проекта были написаны:

1. makefile (для сборки и автоматического тестирования проекта из консоли)
2. CMakeLists.txt (для автоматической сборки проекта в CLion, Visual Studio Code, Visual Studio 17)

Также были взяты "test_generator.py"и "wrapper.sh"с GitHub'а[1] и переделаны.

main.cpp	
void ToLower (char *t, int len)	Перевод char* в нижний регистр
unsigned int LengthString (const char *string)	Получение длинны char*
int main ()	main
PatriciaTree.h	
TPatriciaTree ()	Пустой конструктор
const unsigned long long *Search (const char *, unsigned int)	Поиск по дереву
bool Insert (const char *, unsigned int, unsigned long long)	Вставка в дерево
bool Remove (const char *, unsigned int)	Удаление из дерева
void Load (const char *)	Загрузка дерева из бинарного файла
void Save (const char *)	Сохранение дерева в бинарный файл
virtual ~TPatriciaTree ()	Деструктор дерева
StackContainer.h	
template <class T> class TStackData	Шаблон элемента TStack
T &GetData ()	Получение хранимого объекта
TStackData <T> *GetNext ()	Получение следующего элемента
TStackData <T> *GetPrev ()	Получение предыдущего элемента
virtual ~TStackData ()	Деструктор
template <class T> class TStack : public TStackData <T>	Шаблон структуры данных «стек»
TStack ()	Пустой конструктор
T *Top ()	Получения элемента «сверху» стека
void Pop ()	Изъятие из стека
void Push (T &)	Помещение элемента на стек
bool Empty ()	Проверка является ли стек пустым
virtual TStack ()	Деструктор

3 Код дополнительных файлов

- test_generator.py

```
import sys
import random
import string
import copy
from random import choice
from string import ascii_uppercase
def get_random_key():
    return ''.join(choice(ascii_uppercase) for i in range(5))

if __name__ == "__main__":
    count_of_tests = 5
    actions = [ "+", "-", "?", "!"]
    acts_file = ["Load test", "Save test"]
    for enum in range( count_of_tests ):
        keys = {}
        save_file = {}
        test_file_name = "tests/{:02d}".format( enum + 1 )
        with open( "{0}.t".format( test_file_name ), 'w' ) as output_file, \
             open( "{0}.a".format( test_file_name ), "w" ) as answer_file:

            for _ in range( random.randint(10000, 10000) ):
                action = random.choice( actions )
                if action == "+":
                    key = get_random_key()
                    value = random.randint( 1, 10)
                    output_file.write("+ {0} {1}\n".format( key, value ))
                    key = key.lower()
                    answer = "Exist"
                    if key not in keys:
                        answer = "OK"
                        keys[key] = value
                    answer_file.write( "{0}\n".format( answer ) )

                elif action == "?":
                    search_exist_element = random.choice( [ True, False ] )
                    key = random.choice( [ key for key in keys.keys() ] ) if
                        search_exist_element and len( keys.keys() ) > 0 else
                        get_random_key()
                    output_file.write( "{0}\n".format( key ) )
                    key = key.lower()
                    if key in keys:
                        answer = "OK: {0}".format( keys[key] )
                    else:
                        answer = "NoSuchWord"
                    answer_file.write( "{0}\n".format( answer ) )

```

```
                elif action == "-":
                    key = get_random_key()
                    output_file.write("- {0}\n".format(key))
                    key = key.lower()
                    answer = "NoSuchWord"
                    if key in keys:
                        del keys[key]
                        answer = "OK"
                    answer_file.write("{0}\n".format( answer ) )

                elif action == "!":
                    act_file = random.choice(acts_file)
                    if act_file == "Save test":
                        output_file.write("{0} {1}\n".format( action, act_file ))
                        save_file = keys.copy()
                        answer = "OK"
                    elif act_file == "Load test":
                        output_file.write("{0} {1}\n".format( action, act_file ))
                        keys = {}
                        keys = save_file.copy()
                        answer = "OK"
                    answer_file.write( "{0}\n".format( answer ) )
```

- makefile

```
FLAGS=-g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long -O2
CC=g++
LINK=-lm

all: tree main

main: main.cpp
  $(CC) $(FLAGS) -c main.cpp
  $(CC) $(FLAGS) -o Patricia PatriciaTree.o main.o $(LINK)

maind: main.cpp
  $(CC) $(FLAGSD) -c main.cpp
  $(CC) $(FLAGSD) -o Patricia PatriciaTree.o main.o $(LINK)

mainp: main.cpp
  $(CC) $(FLAGS) -c main.cpp
  $(CC) $(FLAGS) -ftest-coverage -fprofile-arcs -o Patricia PatriciaTree.o main.o
       $(LINK)

tree: PatriciaTree.cpp
  $(CC) $(FLAGS) -c PatriciaTree.cpp

clear:
  rm -f *.o
  rm -fr *.dSYM
```

```
25 
26 runtests:
27   rm -rf tests
28   mkdir tests
29   sh wrapper.sh
```

- benchmark.cpp

```
1  #include <cstdio>
2  #include <map>
3  #include <fstream>
4  #include <iostream>
5  
6  void ToLower (std::string &buffer) {
7      int index = 0;
8      while (index < buffer.size()) {
9          if (buffer[index] >= 'A' && buffer[index] <= 'Z') {
10             buffer[index] += -'A' + 'a';
11         }
12         index++;
13     }
14 }
15 
16 int main () {
17     std::ios::sync_with_stdio(false); //make_faster
18     std::ofstream fout("benchTime", std::ios::app);
19     std::map <std::string, unsigned long long> map;
20     std::string buffer;
21     uint64_t data;
22     char choice;
23     clock_t start = clock();
24     if (!std::cin.get(choice)) {
25         return 0;
26     }
27     std::cin.putback(choice);
28 
29     while (std::cin >> choice) {
30         switch (choice) {
31             case '+': {
32                 std::cin >> buffer >> data;
33                 ToLower(buffer);
34 
35                 if (map.count(buffer) == 0) {
36                     map[buffer] = data;
37                     printf("OK\n");
38                 } else {
39                     printf("Exist\n");
40                 }
41             }
42                 break;
```

```
43              case '-': {
44                  std::cin >> buffer;
45                  ToLower(buffer);
46                  if (map[buffer] == 0) {
47                      printf("NoSuchWord\n");
48                  } else {
49                      map.erase(buffer);
50                      printf("OK\n");
51                  }
52                  break;
53              }
54              default: {
55                  std::cin.putback(choice);
56                  std::cin >> buffer;
57                  ToLower(buffer);
58                  if (map[buffer] == 0) {
59                      std::cout << "NoSuchWord\n" << std::endl;
60                  } else {
61                      std::cout << "OK" << map[buffer] << std::endl;
62                  }
63                  break;
64              }
65          }
66
67      }
68
69      clock_t finish = clock();
70      double time = (double) (finish - start) / CLOCKS_PER_SEC;
71      fout << "Std map time:" << time << std::endl;
72      fout.close();
73
74      return 0;
75  }
```

4 Консоль

```
$ CounterSort make
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c TString.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c Model.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c TVector.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c main.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -o CounterSort TString.o Model.o TVector.o main.o -lm

$ CounterSort cat 01.t
60693 nmbfyzfqu
43600 piqyjgkhf
25976 irudfyioj
3679 duqjersbu
38028 wuxtiiogk
3639 mefjaasbe
3915 loxlyyqcs
42023 sslttackw
58339 zoxkqedkb
$ CounterSort ./CounterSort < 01.t > 01.answ
$ CounterSort cat 01.answ
3639 mefjaasbe
3679 duqjersbu
3915 loxlyyqcs
25976 irudfyioj
38028 wuxtiiogk
42023 sslttackw
43600 piqyjgkhf
58339 zoxkqedkb
60693 nmbfyzfqu
$ CounterSort cat 02.t
14517 tyxzddarc
64044 xelwhtsro
39337 kwktcfiiu
9162 qjevarqqs
64981 mrwlghlcs
```

```
9480 vagjwzilk
44021 egaeqezha
41017 xpcfbjuly
51997 jyjntrrgf
41622 pkabowgux
44235 yborezsfg
$ CounterSort ./CounterSort < 02.t > 02.answ
$ CounterSort cat 02.answ
9162 qjevarqqs
9480 vagjwzilk
14517 tyxzddarc
39337 kwktcfiiu
41017 xpcfbjuly
41622 pkabowgux
44021 egaeqezha
44235 yborezsfg
51997 jyjntrrgf
64044 xelwhtsro
64981 mrwlghlcs
```

5 Проверка на утечки памяти

Тестирование проводилось на файле в 100 000 строк, на системе Arch Linux, с помощью утилиты Valgrind.

```
[Ср окт 04: 05:53] snowden@arch ~/Projects/DA/da1 [master] $ make
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c TString.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c Model.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c TVector.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -c main.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare
-Wno-long-long -O2 -o CounterSort TString.o Model.o TVector.o main.o -lm
[Ср окт 04: 05:54] snowden@arch ~/Projects/DA/da1 [master] $
valgrind ./CounterSort < tests/01.t > tmp
==1417== Memcheck, a memory error detector
==1417== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1417== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==1417== Command: ./CounterSort
==1417==
==1417==
==1417== HEAP SUMMARY:
==1417== in use at exit: 0 bytes in 0 blocks
==1417== total heap usage: 1,517,400 allocs, 1,517,400 frees, 9,380,563 bytes
allocated
==1417==
==1417== All heap blocks were freed -- no leaks are possible
==1417==
==1417== For counts of detected and suppressed errors, rerun with: -v
==1417== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[Ср окт 04: 05:54] snowden@arch ~/Projects/DA/da1 [master] $ wc tests/01.t
100000 200000 1083130 tests/01.t
```

6 Тест производительности

Чтобы увидеть разницу был созданы файлы размером 10^3, 10^5, 10^6, $10 * 10^6$ входных строк и был создан файл бенчмарка, с реализацией сортировки подсчетом и QSort из STD. Исходный код лежит в «дополнительных файлах» выше.

```
$ CounterSort make bench
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long -O2
-c benchmark.cpp
g++ -g -std=c++11 -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long -O2
-o benchmark benchmark.o -lm
$ CounterSort wc tests/01.t
1000    2000   15797 tests/01.t
$ CounterSort ./benchmark < tests/01.t
Std SortTime = 0.000080
Counting SortTime = 0.000241

$ CounterSort wc tests/02.t
100000  200000 1083096 tests/02.t
$ CounterSort ./benchmark < tests/02.t
Std SortTime = 0.010439
Counting SortTime = 0.005754

$ CounterSort wc tests/03.t
100000  200000 1083096 tests/03.t
$ CounterSort ./benchmark < tests/03.t
Std SortTime = 0.095891
Counting SortTime = 0.050415

$ CounterSort wc tests/04.t
10000000 20000000 108302115 tests/04.t
Std SortTime = 0.984407
Counting SortTime = 0.854098
```

7 Дневник отладки

1. 29.09.17; 0:07; Сел за написание лабораторной. Написал TString, TPair, TVector. Долго искал информацию по грамотному разделению шаблона на несколько файлов;

2. 1.10.17; 0:49; Первый провал на системе тестирования, убрал rm из makefile;

3. 1.10.17; 1:04; Неверный ответ на первом тесте, программа теряла где-то одно значение. Забыл убрать отладочную строчку с удалением корневого элемента;

4. 1.10.17; 23:24; Превышение временного порога на 10-м тесте. На самом деле я даже не удивился. Изменять размер вектора при каждом добавлении элемента само по себе не очень решение. Исправлял введением доп. переменной в вектор, которая считала сколько всего элементов реально заполнены и в случае её равенства размеру массива, увеличивал размер в 2 раза;

5. 2.10.17; 2:37; Решил дописать кучу никому не нужных методов и оберток, чтобы не нарушать инкапсуляцию;

6. 2.10.17; 4:55; Превышение временного порога на 12-м тесте. Поправил начальный размер вектора с 1-го элемента до 32-х;

7. 2.10.17; 4:59; Программа получила статус «Ок» на системе автоматического тестирования;

8. 2.10.17; 18:52; Новая версия программы с исправлениями утечек памяти провалила по времени 12-й тест, но это лишь потому что я решил попробовать заслать с начальным массивом в 8 элементов. Также запускал на разных машинках, дописывал код;

9. 2.10.17; 21:41; Решил самоутвердиться и заслал финальную версию программы, которую тестировал пару-тройку часов;

10. 3.10.17; 5:57; Финальная отправка на проверку, перед отправкой отчёта. Внесены мелкие правки по кодстайлу;

11. 3.10.17; 6:27; Засылаю отчёт;

12. 6.10.17; 14:54; Исправил отчёт. Нашёл ошибку в бенчмарке - он показывал не правильное время. Исправил замеры времени, также исправил нарушение инкапсуляции. Пришлось отказаться от шаблона в виде TVector.h+TVector.hpp и написать лишь одну его реализацию, которая зависит только от TPair. Также метод сортировки был перенесён в TVector.cpp.

8 Выводы

Сортировка подсчётом применима, если сортируемые числа имеют диапазон возможных значений, который достаточно мал по сравнению с сортируемым множеством. Само написание программы не вызвало особых затруднений. Куда больше сил и времени было потрачено на всевозможные тесты и приведение кода в презентабельный вид.
Весьма много времени ушло на сравнение моей сортировки с другими. Для себя я выяснил, что тестировать все на железе i7 4/8 + SSD (1400мб/1200мб чтение/запись в сек) глупая затея, ибо чтобы увидеть хоть какую-то разницу нужно создавать входные файлы $> 10^{18}$ строк. Также написание лабораторной мне показалось чем-то схожим с хакатонами, которые я регулярно посещаю: быстро и красиво написать что-то почти не реально. Написанный за ночь код, нужно доводить еще пару тройку дней до совершенства, проводя уйму времени за тестами.
На вооружение забрал себе Radix, включающий в себя сортировку подсчётом. Потому что все проведённые мною тесты показали лучшие результаты именно на нем.

Список литературы

[1] *GitHub одного из семинаристов.*
URL: `https://github.com/toshunster/da` (дата обращения: 29.09.2017).

[2] *Шаблоны функций в C++.*
URL: `http://cppstudio.com/post/5165/` (дата обращения: 30.09.2017).

[3] *Шаблоны функций в C++.*
URL: `http://cppstudio.com/post/673/` (дата обращения: 1.10.2017).

[4] *Функция strcpy.*
URL: `http://cppstudio.com/post/686/` (дата обращения: 1.10.2017).

[5] *Сортировка подсчётом — Википедия.*
URL: `http://ru.wikipedia.org/wiki/Сортировка_подсчётом` (дата обращения: 16.12.2013).

[6] 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.))

[7] *Bitbucket лектора.*
URL: `https://bitbucket.org/nkmakarov/da4students/` (дата обращения: 2.10.2017).