Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Дискретный анализ»

Студент: А. В. Барсов
Преподаватель: Н. С. Капралов
Группа: М8О-307Б-19

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №3

Задача: Для реализации словаря из предыдущей лабораторной работы необходимо провести исследование скорости выполнения и потребления оперативной памяти. В случае выявления ошибок или явных недочётов, требуется их исправить.

Используемые утилиты: valgrind, gprof.

1 Valgrind

Согласно [1], Valgrid — это программа для поиска ошибок обращения с памятью, поиска утечек памяти и профилирования. Для поиска ошибки я использовал средство memcheck.

Я обнаружил, что моя первая версия программы может выдавать неправильный ответ. Для поиска ошибки я использовал Valgrind.

```
[artem@IdeaPad solution]$ valgrind ./solution <tests/test100000.txt >/dev/null
==5244== Memcheck,a memory error detector
==5244== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==5244== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==5244== Command: ./solution
==5244==
==5244==
==5244== HEAP SUMMARY:
==5244==     in use at exit: 1,367,808 bytes in 6,576 blocks
==5244==   total heap usage: 30,757 allocs,24,181 frees,57,590,688 bytes allocated
==5244==
==5244== LEAK SUMMARY:
==5244==    definitely lost: 105,216 bytes in 2,192 blocks
==5244==    indirectly lost: 1,262,592 bytes in 4,384 blocks
==5244==      possibly lost: 0 bytes in 0 blocks
==5244==    still reachable: 0 bytes in 0 blocks
==5244==         suppressed: 0 bytes in 0 blocks
==5244== Rerun with --leak-check=full to see details of leaked memory
==5244==
==5244== For lists of detected and suppressed errors,rerun with: -s
==5244== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Диагностика показала, что программа неверно обращается с удалённой памятью, из-за чего и возникает ошибка.

Запустив valgrind с опцией –leak-check=full я проанализировал проблемное место:

```
[artem@IdeaPad solution]$ valgrind --leak-check=full ./solution <tests/test100000.txt
>/dev/null
==5310== Memcheck,a memory error detector
==5310== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==5310== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==5310== Command: ./solution
==5310==
```

```
==5310==
==5310== HEAP SUMMARY:
==5310==     in use at exit: 1,367,808 bytes in 6,576 blocks
==5310==   total heap usage: 30,757 allocs,24,181 frees,57,590,688 bytes allocated
==5310==
==5310== 31,200 (2,400 direct,28,800 indirect) bytes in 50 blocks are definitely
lost in loss record 4 of 9
==5310==    at 0x4846013: operator new(unsigned long) (in /usr/lib/valgrind/vgpreload
==5310==    by 0x10ACF8: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10AA68: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10951C: main (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/s
==5310==
==5310== 280,176 (21,552 direct,258,624 indirect) bytes in 449 blocks are definitely
lost in loss record 7 of 9
==5310==    at 0x4846013: operator new(unsigned long) (in /usr/lib/valgrind/vgpreload
==5310==    by 0x10ACF8: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10AA68: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10AA68: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10951C: main (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/s
==5310==
==5310== 1,056,432 (81,264 direct,975,168 indirect) bytes in 1,693 blocks are
definitely lost in loss record 9 of 9
==5310==    at 0x4846013: operator new(unsigned long) (in /usr/lib/valgrind/vgpreload
==5310==    by 0x10ACF8: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10AA68: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10AA68: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10AA68: void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*
const&) (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/solution)
==5310==    by 0x10951C: main (in /home/artem/Discrete-Analysis/B-tree(2:4)/solution/s
==5310==
==5310== LEAK SUMMARY:
==5310==    definitely lost: 105,216 bytes in 2,192 blocks
==5310==    indirectly lost: 1,262,592 bytes in 4,384 blocks
```

```
==5310==      possibly lost: 0 bytes in 0 blocks
==5310==    still reachable: 0 bytes in 0 blocks
==5310==         suppressed: 0 bytes in 0 blocks
==5310==
==5310== For lists of detected and suppressed errors,rerun with: -s
==5310== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 0 from 0)
```

Отсуюда видно, что потерялась память, выделенная оператором operator new(unsigned long). Я нашёл место в коде, где я забыл удалять ненужный укзатель, и исправил это

```
delete newToInsertNode;
```

2 Gprof

Как сказано в [3], Gprof — средство профилирования в Unix системах. Используется для измерения времени работы отдельных функций программы и общего времени работы программы.

Профилировщик показывает, сколько процентов от общего времени работы программы работает и сколько раз вызывается каждая функция (и ещё много данных, которые не так важны для нашей задачи).

Диагностика проводилась на тесте из 10000 строк с запросами на поиск, добавление и удаление.

```
[artem@IdeaPad solution]$ gprof ./solution -p ./gmon.out
Flat profile:
Each sample counts as 0.01 seconds.
%
time calls name
75.04 27903 TItem::operator=(TItem const&)
25.01 23463 Clear(char*)
0.00 168425 unsigned short Min<unsigned short>(unsigned short const&,unsigned
short const&)
0.00 146858 TVector<TItem>::operator[](unsigned long long const&)
0.00 109344 operator<(TItem const&,TItem const&)
0.00 80572 TVector<TItem>::Size()
0.00 52618 TVector<TBtreeNode<TItem>*>::operator[](unsigned long long const&)
0.00 39119 unsigned long long BinSearch<TItem>(TVector<TItem>&,TItem const&)
0.00 28638 operator==(TItem const&,TItem const&)
0.00 13463 TItem::TItem()
0.00 10147 void FindNode<TItem>(TBtreeNode<TItem>*,TBtreeNode<TItem>*&,unsigned
long long&,TItem const&)
0.00 10000 TBtree<TItem>::Find(TBtreeNode<TItem>*&,unsigned long long&,TItem
const&)
0.00 8997 TVector<TBtreeNode<TItem>*>::Size()
0.00 6916 TVector<TBtreeNode<TItem>*>::PushBack(TBtreeNode<TItem>* const&)
0.00 4772 TVector<unsigned long long>::PushBack(unsigned long long const&)
0.00 3387 TVector<TBtreeNode<TItem>*>::PopBack()
0.00 2330 unsigned long long Max<unsigned long long>(unsigned long long
const&,unsigned long long const&)
0.00 2195 TVector<TItem>::PushBack(TItem const&)
0.00 2123 TVector<TItem>::Insert(unsigned long long const&,TItem const&)
0.00 2123 TVector<TBtreeNode<TItem>*>::Insert(unsigned long long const&,TBtreeNode4
const&)
```

```
0.00 2030 TVector<TBtreeNode<TItem>*>::~TVector()
0.00 1899 TVector<unsigned long long>::Size()
0.00 1899 TVector<unsigned long long>::operator[](unsigned long long const&)
0.00 1826 TBtree<TItem>::Insert(TItem const&)
0.00 1826 TVector<TItem>::PopBack()
0.00 1825 void InsertNode<TItem>(TBtreeNode<TItem>*&,TBtreeNode<TItem>*&,TItem
const&)
0.00 1635 TVector<TBtreeNode<TItem>*>::TVector()
0.00 1635 TVector<unsigned long long>::TVector()
0.00 1635 TVector<unsigned long long>::~TVector()
0.00 1633 TVector<TItem>::Erase(unsigned long long const&)
0.00 1633 TVector<TBtreeNode<TItem>*>::Erase(unsigned long long const&)
0.00 1612 TVector<unsigned long long>::PopBack()
0.00 1488 void EraseNode<TItem>(TBtreeNode<TItem>*&,TItem const&)
0.00 1488 TBtree<TItem>::Erase(TItem const&)
0.00 395 TBtreeNode<TItem>::TBtreeNode()
0.00 395 TVector<TItem>::TVector(unsigned long long)
0.00 395 TVector<TItem>::~TVector()
0.00 395 TVector<TBtreeNode<TItem>*>::TVector(unsigned long long)
0.00 370 TVector<TBtreeNode<TItem>*>::Resize(unsigned long long const&)
0.00 359 TBtreeNode<TItem>::~TBtreeNode()
0.00 298 TVector<TItem>::Resize(unsigned long long const&)
0.00 147 TItem::TItem(TItem const&)
0.00 82 TVector<unsigned long long>::Resize(unsigned long long const&)
0.00 1 TBtree<TItem>::TBtree()
0.00 1 TBtree<TItem>::~TBtree()
```

Видно, что программа тратит большую часть времни не на операции с деревом, а на операции с ключом. Действительно, моя реализация B-дерева использует динамический массив в узлах, что требует большого количества копирований при вставке и удалении. Исходя из диагностика, наиболее приоритетным для оптимизации является работа со структурой «ключ-значение».

3 Выводы

Выполнив третью лабораторную работу по курсу «Дискретный анализ», я изучил и применил средства диагностики и профилирования, в частности Valgrind и Gprof. Исходя из их популярности, понятно, что они часто используются на практике.

Valgrind помог мне найти неправильное обращение с памятью в программе и исправить эту ошибку.

Gprof показал, какие функции дольше всего выполняет моя программа, а именно работа со строками.

Эти утилиты помогают оптимизировать код с целью уменьшения потребляемой памяти и времени работы программы. Учитывая, что это основные ресурсы компьютера, то таким образом можно в целом улучшить качество работы и быстродейтсвие программы.

Список литературы

[1] *Valgrind* — Wikipedia
URL: `https://en.wikipedia.org/wiki/Valgrind` .

[2] *Справочник по алгоритмам и структурам данных.*
URL: `https://e-maxx.ru/` .

[3] *Gprof* — Wikipedia
URL: `https://en.wikipedia.org/wiki/Gprof` .