Лабораторная работа № 3 по курсу дискретного анализа:

Исследование качества программ

Выполнил студент группы М8О-208Б-20 МАИ *Борисов Ян*.

Условие

Для реализации словаря из предыдущей лабораторной работы, необходимо провести исследование скорости выполнения и потребления оперативной памяти.

gprof

Основная информация

Утилита gprof измеряет время работы всех функций, методов и операторов программы, количество их вызовов и процентное соотношение работы конкретной функции от работы программы.

Команды для работы с утилитой

Компилирую исходную программу с ключом *−pg*:

g++ main.cpp −pg −o main

Затем я запускаю программу, передав ей на ввод файл test.txt, в котором содержится 5000 тестов на вставку, поиск и удаление:

./main < test.txt > out.txt

Вся информация от утилиты gprof сохранилась в файл gmon.out, далее, чтобы получить текстовый файл, выполним следующую команду:

gprof main gmon.out > profile-data.txt

Результат работы утилиты

Ниже приведена таблица с результатами работы утилиты gprof.

% time	self seconds	calls	name
18.19	0.02	1629939	std::string::compare
9.09	0.01	3913824	std:: string::size
9.09	0.01	1938334	operator++()
9.09	0.01	1938334	operator*()
9.09	0.01	15054	TItem::TItem(std::string, unsigned long long)
9.09	0.01	4988	TNode::AddNodeNonFull(TItem)
9.09	0.01	836	TNode::BorrowFromPrev(int)
9.09	0.01	53	std::vector::operator=

Все остальные функции, по данным результатам измерений утилиты gprof, работали примерно 0 секунд, поэтому в таблицу внесены не были. Из полученных данных следует, что большая часть времени работы программы тратится на сравнение строк, так как эта операция нужна для сравнения ключей, что часто применяется в реализации.

valgrind

Valgrind является самым распространённым инструментом для отслеживания утечек памяти и других ошибок, связанных с памятью. Для проверки программы на утечки выполняю команду:

valgrind ./main < test.txt > out.txt

В результате выполнения этой команды получаем следующее сообщение:

```
==9334== Memcheck, a memory error detector
==9334== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9334== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==9334== Command: ./main
==9334==
==9334== error calling PR_SET_PTRACER, vgdb might block
==9334== Invalid read of size 8
==9334==    at 0x110176: std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::size() const (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x10F165: __gnu_cxx::__enable_if<std::__is_char<char>::__value, bool>::__type std::operator==<char>(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocat
or<char> > const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x10B7D2: TItem::operator==(TItem const&) const (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x10BC42: TNode::Search(TItem) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x10BC9C: TNode::Search(TItem) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x10D26D: TBtree::Search(TItem) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x10E40D: main (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==  Address 0x4dde700 is 8 bytes after a block of size 5,560 alloc'd
==9334==    at 0x483BE63: operator new(unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==9334==    by 0x113E89: __gnu_cxx::new_allocator<TItem>::allocate(unsigned long, void const*) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x113383: std::allocator_traits<std::allocator<TItem> >::allocate(std::allocator<TItem>&, unsigned long) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x1124AF: std::_Vector_base<TItem, std::allocator<TItem> >::_M_allocate(unsigned long) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
```

```
==9334==    by 0x111137: TItem* std::vector<TItem, std::allocator<TItem>
>::_M_allocate_and_copy<__gnu_cxx::__normal_iterator<TItem const*, std::vector<TItem, std::allocator<TItem> > >
 >(unsigned long, __gnu_cxx::__normal_iterator<TItem const*, std::vector<TItem, std::allocator<TItem> > >,
__gnu_cxx::__normal_iterator<TItem const*, std::vector<TItem, std::allocator<TItem> > >) (in
/mnt/c/Users/boris/CLionProjects/Algorithms and data structures/lab2_in_one_file/main)
==9334==    by 0x10FB7E: std::vector<TItem, std::allocator<TItem> >::operator=(std::vector<TItem,
std::allocator<TItem> > const&) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data
structures/lab2_in_one_file/main)
==9334==    by 0x10BA88: TNode::TNode(int, bool) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data
structures/lab2_in_one_file/main)
==9334==    by 0x10D31A: TBtree::Add(TItem const&, std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >) (in /mnt/c/Users/boris/CLionProjects/Algorithms and data
structures/lab2_in_one_file/main)
==9334==    by 0x10DF87: main (in /mnt/c/Users/boris/CLionProjects/Algorithms and data
structures/lab2_in_one_file/main)
==9334==
==9334==
==9334== HEAP SUMMARY:
==9334==     in use at exit: 1,500,765 bytes in 3,442 blocks
==9334==   total heap usage: 60,274 allocs, 56,832 frees, 9,776,260 bytes allocated
==9334==
==9334== LEAK SUMMARY:
==9334==    definitely lost: 64 bytes in 1 blocks
==9334==    indirectly lost: 0 bytes in 0 blocks
==9334==      possibly lost: 0 bytes in 0 blocks
==9334==    still reachable: 0 bytes in 0 blocks
==9334==         suppressed: 0 bytes in 0 blocks
==9334== Rerun with --leak-check=full to see details of leaked memory
==9334==
==9334== For lists of detected and suppressed errors, rerun with: -s
==9334== ERROR SUMMARY: 4 errors from 1 contexts (suppressed: 0 from 0)
```

С помощью Valgrind обнаружили несколько незначительных ошибок и неосвобождённую память после выполнения программы. Still reachable блоки я убрал путем удаления из кода строк, ускоряющих работу программы:

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

Выводы

В ходе выполнения лабораторной работы №3, я познакомился с утилитой gprof, а утилиту Valgrind я активно использовал и до выполнения данной работы. Лабораторная работа так же позволила выявить некоторые проблемы в программе, которые можно исправить.