МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Московский технологический университет"

МИРЭА

Институт информационных технологий (ИТ)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ
по дисциплине
«Анализ сложности алгоритмов»

Выполнил студент группы ИКБО-08-16 Вишняков А.А.

Лабораторные работы выполнены «___»_____2017 г. _____

«Зачтено» «___»_____2017 г. _____

Москва 2017

Содержание:

Постановка задачи

1. Изучить и реализовать алгоритмы сортировок:
   - Сортировка пузырьком
   - Сортировка вставками
   - Быстрая сортировка
   - Сортировка с помощью двоичного дерева.

2. Изучить и реализовать алгоритмы поиска:
   - Линейный поиск
   - Бинарный поиск
   - Интерполяционный поиск
   - Поиск по двоичному дереву.

3. Предусмотреть возможность ввода элементов с клавиатуры, чтения из файла.

4. Для каждого алгоритма посчитать время выполнения, количество перестановок и сравнений элементов при работе с файлами, содержащими 10000, 30000, 50000, 70000, 90000. На основе полученных данных сделать таблицу, построить графики.

5. Сделать вывод.

Алгоритмы сортировок

Сортировка пузырьком

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются $N - 1$ раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

Сложность: $O(n^2)$

- Число сравнений в теле цикла равно $(N - 1)\,\frac{N}{2}$
- Число сравнений в заголовках циклов равно $(N - 1)\,\frac{N}{2}$
- Суммарное число сравнений равно $(N - 1)\,N$
- Число присваиваний в заголовках циклов равно $(N - 1)\,\frac{N}{2}$
- Число обменов равно $(N - 1)\,\frac{N}{2}$

Сортировка вставками

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

Количество сравнений, очевидно, зависит не только от N, но и от их расположения. В худшем случае элементы массива расположены в обратном порядке, и алгоритм выполняет максимум сравнений и перестановок. В этом случае и количество сравнений, и количество перестановок равно $\frac{N(N-1)}{2}$.

Быстрая сортировка

Быстрая сортировка относится к алгоритмам «разделяй и властвуй».

Алгоритм состоит из трех шагов:

1. Выбрать элемент из массива. Назовём его опорным.
2. Разбиение: перераспределение элементов в массиве таким образом, что элементы меньше опорного помещаются перед ним, а больше или равные после.
3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один или отсутствуют элементы.

Сложность в среднем составляет $O(n * log(n))$.

Сортировка с помощью двоичного дерева

Универсальный алгоритм сортировки, заключающийся в построении двоичного дерева поиска по ключам массива (списка), с последующей сборкой результирующего массива путём обхода узлов построенного дерева в необходимом порядке следования ключей. Данная сортировка является оптимальной при получении данных путем непосредственного считывания с потока (например, из файла, сокета или консоли).

Алгоритм:

1. Построение двоичного дерева.

2. Сборка результирующего массива путём обхода узлов в необходимом порядке следования ключей.

Для n объектов сложность будет составлять $O(n * log(n))$.

Алгоритмы поиска

Линейный поиск

Поиск значения функции осуществляется простым сравнением очередного рассматриваемого значения (как правило, поиск происходит слева направо, то есть от меньших значений аргумента к большим) и, если значения совпадают (с той или иной точностью), то поиск считается завершенным.

Асимптотическая сложность алгоритма — $O(n)$.

Для списка из n элементов лучшим случаем будет тот, при котором искомое значение равно первому элементу списка и требуется только одно сравнение. Худший случай будет тогда, когда значения в списке нет (или оно находится в самом конце списка), в случае чего необходимо n сравнений.

Бинарный поиск

Алгоритм поиска элемента в отсортированном массиве (векторе), использующий дробление массива на половины.

1. Определение значения элемента в середине структуры данных. Полученное значение сравнивается с ключом.
2. Если ключ меньше значения середины, то поиск осуществляется в первой половине элементов, иначе - во второй.
3. Поиск сводится к тому, что вновь определяется значение серединного элемента в выбранной половине и сравнивается с ключом.
4. Процесс продолжается до тех пор, пока не будет найден элемент со значением ключа или не станет пустым интервал для поиска.

Интерполяционный поиск

Основан на принципе поиска в телефонной книге или, например, в словаре. Вместо сравнения каждого элемента с искомым, как при линейном поиске, данный алгоритм производит предсказание местонахождения элемента: поиск происходит подобно двоичному поиску, но вместо деления области поиска на две части, интерполирующий поиск производит оценку новой области поиска по расстоянию между ключом и текущим значением элемента. Другими

словами, бинарный поиск учитывает лишь знак разности между ключом и текущим значением, а интерполирующий ещё учитывает и модуль этой разности и по данному значению производит предсказание позиции следующего элемента для проверки. В среднем интерполирующий поиск производит $log(log(N))$ операций, где N есть число элементов, среди которых производится поиск.

Поиск по двоичному дереву

Задача: проверить, есть ли узел с ключом K в дереве T, и если да, то вернуть ссылку на этот узел.

Алгоритм:

- Если дерево пусто, сообщить, что узел не найден, и остановиться.
- Иначе сравнить K со значением ключа корневого узла X.
- Если K=X, выдать ссылку на этот узел и остановиться.
- Если K>X, рекурсивно искать ключ K в правом поддереве T.
- Если K<X, рекурсивно искать ключ K в левом поддереве T.

Ход работы:

1. Программа на языке C

При запуске программы пользователю дается на выбор два способа ввода элементов. Если выбрано считывание с файла, то необходимо ввести имя файла или путь до него. После программа выполняет последовательно все сортировки. Далее предлагается выбрать элемент для поиска. Есть возможность выбрать случайный элемент из массива. После выполняется поиск данного элемента с помощью всех алгоритмов поиска.

Для каждого алгоритма программа считает затраченное на выполнение время, количество сравнений и перестановок.

```
        Sort type: Bubble
Sort duration: 0.524666 s
Compares count = 99080180
Moves count = 24945431

        Sort type: Quick
Sort duration: 0.001352 s
Compares count = 133904
Moves count = 30891

        Sort type: Insertion
Sort duration: 0.109517 s
Compares count = 24955429
Moves count = 24955429

        Sort type: Binary
Sort duration: 0.002467 s
Compares count = 149920
Moves count = 10000

Choose search value getting method:
Type 'r' to use random
Type 'e' to enter from da keyboard
r

        Search type: Linear
752555423 has position 7919.
Search duration: 0.000047 s
Compares count = 7919

        Search type: Tree
752555423 has been found.
Search duration: 0.000011 s
Compares count = 19

        Search type: Interpolation
752555423 has position 7919
Total compares count: 1610
Search duration: 0.000045 s

        Search type: Binary
752555423 has position 7919.
Search duration: 0.000003 s
Compares count = 12
```

Рис. 1. Результаты выполнения алгоритмов сортировки и поиска для 10000 элементов

```
        Sort type: Bubble
Sort duration: 4.865502 s
Compares count = 894180384
Moves count = 224860122

        Sort type: Quick
Sort duration: 0.004359 s
Compares count = 441793
Moves count = 104466

        Sort type: Insertion
Sort duration: 0.951124 s
Compares count = 224890120
Moves count = 224890120

        Sort type: Binary
Sort duration: 0.009292 s
Compares count = 515493
Moves count = 30000

Choose search value getting method:
Type 'r' to use random
Type 'e' to enter from da keyboard
r

        Search type: Linear
97517962 has position 3066.
Search duration: 0.000026 s
Compares count = 3066

        Search type: Tree
97517962 has been found.
Search duration: 0.000009 s
Compares count = 20

        Search type: Interpolation
97517962 has position 3066
Total compares count: 1036
Search duration: 0.000029 s

        Search type: Binary
97517962 has position 3066.
Search duration: 0.000008 s
Compares count = 11
```

Рис. 2. Результаты выполнения алгоритмов сортировки и поиска для 30000 элементов

```
        Sort type: Bubble
Sort duration: 14.360405 s
Compares count = 2490200388
Moves count = 626960302

        Sort type: Quick
Sort duration: 0.008185 s
Compares count = 834518
Moves count = 181093

        Sort type: Insertion
Sort duration: 2.963753 s
Compares count = 627010300
Moves count = 627010300

        Sort type: Binary
Sort duration: 0.019872 s
Compares count = 910283
Moves count = 50000

Choose search value getting method:
Type 'r' to use random
Type 'e' to enter from da keyboard
r

        Search type: Linear
147546742 has position 7753.
Search duration: 0.000035 s
Compares count = 7753

        Search type: Tree
147546742 has been found.
Search duration: 0.000015 s
Compares count = 18

        Search type: Interpolation
147546742 has position 7753
Total compares count: 2510
Search duration: 0.000052 s

        Search type: Binary
147546742 has position 7753.
Search duration: 0.000011 s
Compares count = 14
```

Рис. 3. Результаты выполнения алгоритмов сортировки и поиска для 50000 элементов

```
        Sort type: Bubble
Sort duration: 27.911156 s
Compares count = 578643454
Moves count = 1228404746

        Sort type: Quick
Sort duration: 0.011847 s
Compares count = 1164791
Moves count = 262585

        Sort type: Insertion
Sort duration: 5.496273 s
Compares count = 1228474744
Moves count = 1228474744

        Sort type: Binary
Sort duration: 0.028003 s
Compares count = 1321962
Moves count = 70000

Choose search value getting method:
Type 'r' to use random
Type 'e' to enter from da keyboard
r

        Search type: Linear
595943930 has position 43997.
Search duration: 0.000113 s
Compares count = 43997

        Search type: Tree
595943930 has been found.
Search duration: 0.000016 s
Compares count = 26

        Search type: Interpolation
595943930 has position 43997
Total compares count: 9982
Search duration: 0.000204 s

        Search type: Binary
595943930 has position 43997.
Search duration: 0.000015 s
Compares count = 14
```

Рис. 4. Результаты выполнения алгоритмов сортировки и поиска для 70000 элементов

```
        Sort type: Bubble
Sort duration: 45.137119 s
Compares count = 3783163186
Moves count = 2030958761

        Sort type: Quick
Sort duration: 0.014846 s
Compares count = 1516604
Moves count = 346057

        Sort type: Insertion
Sort duration: 8.553547 s
Compares count = 2031048759
Moves count = 2031048759

        Sort type: Binary
Sort duration: 0.035239 s
Compares count = 1744507
Moves count = 90000

Choose search value getting method:
Type 'r' to use random
Type 'e' to enter from da keyboard
r

        Search type: Linear
374146765 has position 35615.
Search duration: 0.000091 s
Compares count = 35615

        Search type: Tree
374146765 has been found.
Search duration: 0.000020 s
Compares count = 22

        Search type: Interpolation
374146765 has position 35615
Total compares count: 9680
Search duration: 0.000210 s

        Search type: Binary
374146765 has position 35615.
Search duration: 0.000017 s
Compares count = 16
```

Рис. 5. Результаты выполнения алгоритмов сортировки и поиска для 90000 элементов

Обработка результатов

Время выполнения

	10000	30000	50000	70000	90000
Сортировка пузырьком	0.524666	4.865502	14.360405	27.911156	45.137119
Быстрая сортировка	0.001352	0.004359	0.008185	0.011847	0.014846
Сортировка вставками	0.109517	0.951124	2.963753	5.496273	8.553547
Сортировка двоичным деревом	0.002467	0.009292	0.019872	0.028003	0.035239
Линейный поиск	0.000047	0.000026	0.000035	0.000113	0.000091
Поиск по дереву	0.000011	0.000009	0.000015	0.000016	0.000020
Интерполяционный поиск	0.000045	0.000029	0.000052	0.000204	0.000210
Бинарный поиск	0.000003	0.000008	0.000011	0.000015	0.000017

Количество сравнений

	10000	30000	50000	70000	90000
Сортировка пузырьком	99080180	894180384	2490200388	578643454	3783163186
Быстрая сортировка	133904	441793	834518	1164791	1516604
Сортировка вставками	24955429	224890120	627010300	1228474744	2031048759
Сортировка двоичным деревом	149920	515493	1015611	1321962	1744507
Линейный поиск	7919	3066	7753	43997	35615
Поиск по дереву	19	20	18	26	22
Интерполяционный поиск	1610	1036	2510	9982	9680
Бинарный поиск	12	11	14	14	16

Количество перестановок

	10000	30000	50000	70000	90000
Сортировка пузырьком	24945431	224860122	626960302	1228404746	2030958761
Быстрая сортировка	30891	104466	181093	262585	346057
Сортировка вставками	24955429	224890120	627010300	1228474744	2031048759
Сортировка двоичным деревом	10000	30000	50000	70000	90000
Линейный поиск	-	-	-	-	-
Поиск по дереву	-	-	-	-	-
Интерполяционный поиск	-	-	-	-	-
Бинарный поиск	-	-	-	-	-

Заключение

Изучив алгоритмы сортировок и поиска, понятием их вычислительной сложности, я осознал, что вычислительная сложность - решающий фактор, который влияет на быстроту решения поставленной задачи. Проделанная работа показывает, что от выбранного алгоритма зависит время выполнения задачи. Это значит, что стоит с пониманием выбирать алгоритм под конкретно поставленную задачу.