Лабораторная работа №1.
Исследование алгоритмов поиска.

Цель работы: Анализ и исследование характеристик различных алгоритмов поиска в таблицах.

Содержание работы: В процессе выполнения лабораторной работы должны быть разработаны приложения, анализирующие следующие характеристики алгоритмов поиска:

- Время выполнения;
- Количество сравнений.

При выполнении трансляции исходного текста в объектный код программы – ассемблеры и компиляторы языков высокого уровня выполняют различные операции с многими таблицами, основной из которых является операция поиска элементов.

Существует несколько различных алгоритмов поиска, зависящих от структур данных, над которыми выполняется данная операция. При дальнейшем рассмотрении предполагается, что группа данных, в которой необходимо отыскать заданный элемент, фиксирована. Эта группа может быть организована в виде массива или таблицы.

Простые методы поиска

1. Линейный поиск. Если нет никакой дополнительной информации о разыскиваемых данных, то очевидный подход – простой последовательный просмотр таблицы с увеличением шаг за шагом той его части, где искомого элемента не обнаружено. Такой метод называется линейным поиском. Условия окончания такого поиска имеют вид:

- Элемент найден, т. е. $a_i = x$.
- Весь массив просмотрен, и совпадения не обнаружено.

Можно показать, что среднее количество сравнений при работе данного алгоритма равно $N/2$, где N – размер таблицы.

2. Двоичный (логарифмический) поиск. Известно, что поиск можно сделать гораздо более эффективным, если данные будут упорядочены. Основная идея заключается в случайном выборе некоторого элемента a_m и сравнении его с искомым элементом x. Если он равен x, то поиск заканчивается. Если он меньше x, то из этого следует, что все элементы с индексами, меньшими или равными m можно исключить из дальнейшего поиска. Если он больше x, то из дальнейшего поиска исключаются индексы больше или равные m. Это соображение приводит к следующему алгоритму, известному как двоичный поиск, или поиск делением пополам. Поиск начинается с середины таблицы. Искомый элемент сравнивается с элементом, находящимся в середине таблицы. Он может быть больше, равным или меньше, чем проверяемая величина. Дальнейшие действия для каждого из элементов таблицы зависят от результата сравнения:

- Если равно, то элемент найден.
- Если больше, то поиск продолжается в верхней части таблицы.
- Если меньше, то поиск продолжается в нижней части таблицы.

Этот метод делит таблицу пополам при каждой проверке, систематически локализуя искомую величину. Поскольку каждый раз часть таблицы делится

пополам, при таком поиске требуется сделать максимально $\log_2(N)$ проверок. Сравнивая время линейного поиска с временем двоичного поиска и обозначая через A и B время одной проверки соответственно для каждого из методов, можно получить: $T_{lin} = A \times N/2$; $T_{bin} = B \times \log_2(N)$. Поскольку алгоритм двоичного поиска более сложный, можно ожидать, что константа B >> A. Кроме того, для успешной работы алгоритма двоичного поиска необходимо выполнять следующее условие для размера таблицы: $N = 2^k, k = 1,2,3,....$

Улучшенные методы поиска.

Алгоритмы двоичного поиска, несмотря на то, что они являются быстрыми, могут работать только с таблицами, которые упорядочены. Поэтому такие процедуры поиска могут использоваться только совместно с алгоритмом сортировки, который и упорядочивает данные. В действительности, для достижения хорошей скорости при поиске нет необходимости иметь упорядоченную таблицу. Можно добиться значительно лучших результатов при использовании неупакованной таблицы при условии ее избыточности. Это означает, что количество памяти, отведенной для такой таблицы, превышает число хранимых в ней величин. Улучшение может быть достигнуто путем заполнения таблицы случайным (или псевдослучайным) путем. Случайный порядковый номер K для данного элемента формируется из значения элемента. Если позиция с номером K пуста, то туда помещается новый элемент. В противном случае ищется другая позиция для размещения. Алгоритмы поиска такого сорта принято называть поиском методом прямого доступа. Основная проблема при заполнении таблицы таким способом заключается в генерации случайного числа, основываясь на значении данного элемента. Одной из хороших возможностей является простое деление размещаемой величины на длину таблицы N и использование остатка. Условием успешного применения такого метода является отсутствие общих множителей у размера таблицы и значения элемента. В этом случае для данной группы из M размещаемых величин остатки будут почти равномерно распределены в диапазоне от 0 до (N-1). Вторая проблема состоит в создании процедуры, которой следует придерживаться, когда подготовленный элемент попал на занятую позицию. Одним из методов решения этой задачи является рассматриваемый ниже метод открытой адресации. Он заключается в следующем.

Пусть первая проба дает позицию с номером K и эта позиция уже занята. В этом случае проверяется следующая, (K+1)-ая позиция и т. д., до тех пор, пока не будет найдена свободная позиция. Если поиск выходит за пределы таблицы, то он возобновляется с ее начала. (Считается, что таблица циклическая). Ниже приводится пример, иллюстрирующий применение данного метода. Пусть таблица имеет 17 позиций (N=17), в которых должны быть размещены следующие двенадцать чисел: 19,13,5,27,1,26,31,16,9,2,11,21. Эти числа должны быть занесены в таблицу в позиции, определяемые остатком от деления на 17. Если текущая позиция уже заполнена, то проверяется следующая позиция, и т. д.

Процесс заполнения таблицы иллюстрируется ниже.

Позиция	Величина	Количество проверок на обнаружение	Количество проверок на отсутствие.
0			1
1	1	1	6
2	19, 2*	1	5
3	2	2	4
4	21	1	3
5	5	1	2
6			1
7			1
8			1
9	26, 9*	1	7
10	27, 9*	1	6
11	9, 11*	3	5
12	11	2	4
13	13	1	3
14	31	1	2
15			1
16	16	1	1

Общее количество проверок на обнаружение – 16, общее количество проверок на отсутствие – 54. Колонка «количество проверок на обнаружение» дает число попыток для нахождения соответствующей величины в таблице, колонка «количество попыток на отсутствие» дает число попыток для того, чтобы определить, что данное число в таблице отсутствует. Очевидно, что число отсутствует, если при его поиске встретилась пустая позиция. Например, для поиска числа 54 в данном случае первоначальная позиция была бы равна 3 и потребовалось бы 4 попытки для определения его отсутствия. Символом «*» отмечены случаи разрешения конфликтных ситуаций. Для рассматриваемого примера:

Число попыток для заполнения таблицы = 16;

Среднее число проверок на обнаружение = 1,33;

Среднее число проверок на отсутствие = 3,37.

Соответствующие значения для упакованной таблицы, использующей метод быстрой сортировки и логарифмический поиск:

Число попыток для заполнения таблицы и ее сортировки: $M + M \times \log_2 M = 55$;

Среднее число проверок на обнаружение: $\log_2 M = 3{,}58$;

Среднее число проверок на отсутствие: $\log_2 M = 3{,}58$.

Таким образом, оказывается, что метод открытой адресации имеет преимущество в скорости, но платой за это является использование таблицы с избыточностью. Кроме того, очень трудно удалить какой – либо элемент из такой таблицы, так как простое удаление приведет к нарушению цепочки адресов.

Порядок выполнения лабораторной работы.

Для выполнения настоящей лабораторной работы необходимо разработать функции, реализующие рассмотренные выше алгоритмы поиска. Внутри указанных функций необходимо фиксировать действительное количество выполняемых сравнений.

Приложение, реализующее лабораторную работу, должно выполнять следующие действия:

1. Сформировать таблицу, состоящую из множества n произвольных слов по заданию преподавателя.
2. Некоторое слово, принадлежащее этому множеству, подвергнуть процедуре линейного поиска.
3. Некоторое слово, не принадлежащее этому множеству, также подвергнуть процедуре линейного поиска.
4. Отсортировать таблицу из п. 1 одним из методов сортировки.
5. Некоторое слово, принадлежащее этому множеству, подвергнуть процедуре двоичного поиска.
6. Некоторое слово, принадлежащее этому множеству, также подвергнуть процедуре двоичного поиска.
7. Множество слов из п. 1 разместить методом открытой адресации в соответствующей таблице. Число попыток и текущее значение коэффициента заполнения таблицы ρ при заполнении для каждого элемента таблицы должно фиксироваться. Значение ρ определяется по формуле: $\rho = \frac{K-1}{N}$; где (К-1) - количество заполненных к текущему шагу элементов.
8. По окончании заполнения таблицы необходимо подсчитать общее количество попыток при запоминании T_s.
9. Для слов, принадлежащих заданному множеству, л выполнить процедуру поиска методом прямого доступа, фиксируя число проверок на обнаружение для каждого i – го слова t_i. По окончании поиска всех слов необходимо подсчитать общее количество попыток при поиске $T_r = \sum_i t_i$, и среднее значение этой величины.
10. Для некоторого множества слов, не принадлежащих заданному множеству, выполнить процедуру поиска методом прямого доступа, фиксируя число проверок на отсутствие для каждого j – го слова t_j. По окончании проверки всех слов необходимо подсчитать общее количество попыток на отсутствие $T_n = \sum_j t_j$, и среднее значение этой величины.

Содержание отчета.

Отчет по лабораторной работе должен включать в себя:

1. Листинг разработанного приложения с комментариями;
2. Расчет среднего числа проверок на обнаружение величины $T_u^{aver} = T_u / K$ и среднего числа проверок на отсутствие величины $T_n^{aver} = T_n / K$, где К – количество проверяемых слов, для метода поиска прямым доступом;

3. Теоретический расчет количества проверок на обнаружение, в зависимости от значения коэффициента заполнения таблицы ρ по формуле:

$$T_S(\rho) = 1 + \frac{\rho}{2 \times (1 - \rho)}$$

4. Теоретический расчет количества проверок на отсутствие слова в таблице, в зависимости от значения коэффициента заполнения таблицы ρ по формуле:

$$T_n(\rho) = \frac{1}{1 - \rho}$$

4. Таблицы и графики по п. п. 3 и 4;
6. Краткие выводы по проделанной работе.

Литература:

1. Н. Вирт. «Алгоритмы и структуры данных». М.: «Мир», 1989 г.
2. Д. Кнут. « Искусство программирования для ЭВМ». Т. 3. М.: «Мир», 1978 г.