1 Поиск

1.1 Задача поиска

Задача поиска состоит в том, чтобы найти определенный объект в некоторой совокупности объектов, т.е. установить факт его существования или отсутствия и, возможно, местоположение.

Рассматриваемая совокупность считается либо изначально линейно упорядоченной, либо линейно упорядочиваемой в процессе просмотра. Различаются структуры *прямого доступа* (массивы), любой элемент которых может быть выбран непосредственно, по заданному адресу (номеру, индексу), и структуры *последовательного доступа* (файлы, списки), элементы которых доступны после доступа к предыдущим элементам. Методы поиска (и сортировки) для этих структур будут различны. В дальнейшем, если не оговорено иное, рассматриваемая совокупность будет считаться линейным массивом.

Объекты в общем случае рассматриваются как записи произвольной природы, однако имеющие в своей структуре один и тот же *ключ* — поле, содержащее значение, которое сранивается в процессе поиска с искомым ключом. В более общем случае ключ можно рассматривать как функцию, которая строит значение ключа на основании сколь угодно сложного анализа всех данных, представленных в записи. Однако при рассмотрении методов поиска и сортировки мы для простоты будем отождествлять записи с их ключами.

Следующие описания структур данных будут использоваться в дальнейших алгоритмах: /описания глобальных констант, типов и данных на C:

```
#define N 100     /* размер входного массива */
typedef int key;   /* тип ключа */
static key a[N];    /* выходной массив */
```

1.2 Прямой поиск подстроки

Пусть заданы строка s из N элементов и строка q из M элементов, где $0 < M \leq N$. Требуется определить, содержится ли строка q в строке s и в случае положительного результата выдать позицию k, с которой начинается вхождение q в s, т. е.

$$q[1] = s[k], q[2] = s[k+1], ..., q[M] = s[k+M-1].$$

Будем называть строку q *шаблоном* поиска. Работа алгоритма прямого поиска заключается в поиске номера k, указывающего на первое с начала строки s совпадение с шаблоном q.

Алгоритм поиска сводится к повторяющимся сравнениям отдельных символов из строки и шаблона. Сначала сравниваются их первые символы. Если они совпадают, сравниваются вторые и т. д. Можно сказать, что шаблон q «накладывается» на строку s начиная с первого символа строки. Если до i-й позиции символы строки и шаблона совпадали, а на i-м $q[i] \neq s[i]$, то надо вернуться к первому символу шаблона и сравнить его с вторым символом из строки s. Т. е. в этом случае шаблон следует «наложить» на строку со второго символа строки, повторить сравнения и т. д.

Процесс повторяется до тех пор, пока смещение q по s не достигнет $N - M + 1$ или пока не будет найдено k такое, что $q[1..M] = s[k..k + M - 1]$. Очевидно, что данный алгоритм реализуется с помощью двух вложенных циклов, и в худшем случае требуется произвести $(N-M)\cdot M$ сравнений.

1.3 Алгоритм Рабина - Карпа поиска подстроки

Пусть строка и шаблон состоят из символов алфавита A. Каждый символ этого алфавита будем считать d-ичной цифрой, где $d = |A|$. Тогда строку из k символов можно рассматривать как d-ичную запись k-значного числа. Алгоритм поиска шаблона сводится к повторяющимся сравнениям чисел, представляющих подстроки строки s и шаблон.

Для начала предположим, что $A = \{0, 1, .., 9\}$. Число, десятичной записью которого является шаблон q, обозначим через t_q. Аналогично, обозначим через t_k число, десятичной записью которого является подстрока $s[k..k+M-1]$. Очевидно, что подстрока $s[k..k+M-1]$ совпадает с шаблоном q тогда и только тогда, когда $t_q = t_k$.

По схеме Горнера, описанной в первой части этого учебного пособия, значения t_q и t_1 можно вычислить за время, пропорциональное M. Временно забудем о том, что вычисления могут привести к очень большим числам. Из t_k $(k < 1 \leq N - M)$ за константное время можно вычислить t_{k+1}. В самом деле, по схеме Горнера

$$t_k = s[k+M-1]+10\cdot(s[k+M-2]+10\cdot(s[k+M-3]+..+10\cdot(s[k])..) \quad (1)$$

Чтобы получить t_{k+1} из t_k надо удалить последнее слагаемое из формулы (1) (т.е. вычесть $10^{M-1}\cdot(s[k])$), результат умножить на 10 и добавить к нему $s[k+M]$. В результате получим следующее рекурентное соотношение:

$$t_{k+1} = 10 \cdot (t_k - 10^{M-1} \cdot s[k]) + s[k+M] \tag{2}$$

Таким образом, вычислив все t_k, мы можем поочереди сравнить их с t_q, определив тем самым совпадение или несовпадение шаблона q с подстроками $s[k..k+M-1]$. Быстродействие этого алгоритма пропорционально $N+M$.

До сих пор мы не учитывали того, что числа могут быть слишком велики, и предположение о том, что t_{k+1} можно вычислить за константное время окажется неверным. С этой трудностью можно справиться следующим образом. Надо проводить вычисления чисел t_q и t_1 и вычисления по формуле (2) по модулю фиксированного числа p. Тогда все числа не превосходят p и действительно могут быть вычислены за время $\sim (N+M)$. Обычно в качестве p выбирают простое число, для которого $d \cdot p$ помещается в машинное слово, все вычисления в этом случае упрощаются. Рекурентная формула (2) приобретает вид

$$\begin{array}{l} t_{k+1} = (d \cdot (t_k - h \cdot s[k]) + s[k+M]) mod\ p \\ \text{где} h \equiv d^{M-1} (mod\ p) \end{array} \tag{3}$$

Обратим внимание на то, что из равенства $t_q \equiv t_k (mod\ p)$ еще не следует, что $q = s[k..k+M-1]$, в этом случае надо проверить совпадение шаблона и подстроки.

Алгоритм и функция, описанные ниже, определяют все вхождения шаблона q в строку s.

Алгоритм А4:

- вход: q – шаблон, s – строка, M – длина шаблона, N – длина строки;

  По схеме Горнера вычислить числа t_q и t_1 по модулю p;

  цикл по k от 2 до $M-N$

      по формуле (3) вычислить t_k

      если $t_q = t_k$ то сравнить шаблон q с подстрокой $s[k..k+M-1]$

          если они совпадают, то вывести число k;
- выход: все числа k – позиции начал вхождения шаблона в строку.