**Тема 2. ПОИСК ОБРАЗА В СТРОКЕ**

Одно из наиболее часто встречающихся в программировании действий - поиск. Он же представляет собой идеальную задачу, на которой можно ис­пытывать различные структуры данных по мере их появления. Существует несколько основных «вариаций этой темы», и для них создано много различ­ных алгоритмов. В данной теме будет рассмотрен специфический поиск, так называемый поиск строки. Его можно определить следующим образом.

Пусть задан массив *str* из *N* элементов и массив *img* из *М* элементов, причем *0 ≤ М < N*. Описаны они так:

*char str[N];*

*char img[M];*

Поиск строки обнаруживает первое вхождение *img* в *str*. Оба массива содержат символы, так что *str* можно считать некоторым текстом или строкой, а *img* - образом, первое вхождение в строку которого необходимо найти. Это действие типично для любых систем обработки текстов, отсюда и очевид­ная заинтересованность в поиске эффективного алгоритма для этой задачи.

1. **Прямой поиск строки**

Прежде чем обратить внимание на эффективность, разберем некий «прямолинейный» алгоритм поиска, который называется прямым поиском строки. При прямом поиске строки сравнивается первый символ образа и со­ответствующий ему символ из строки. В начале работы алгоритма этим сим­волом будет первый символ строки. Если сравниваемые символы совпадают, то рассматриваются следующие, вторые, символы образа и строки. Таким образом происходит перемещение по образу от его начала к концу. Если образ закончился, значит, все символы образа совпали с символами рассматри­ваемой части строки, т. е. поиск образа в строке выполнен успешно.

В том случае, если произошло несовпадение символов образа и строки, образ сдвигается по строке на один символ вправо, и снова происходит срав­нение образа, начиная с первого символа. Теперь уже первый символ образа сравнивается со вторым символом строки, второй символ образа - с третьим символом строки и т. д. Поиск происходит до тех пор, пока не будет найдено вхождение образа в строке или же не закончится строка, что значит, что вхо­ждение образа не было найдено. На рис. 1 приведен пример прямого поиска строки, сравниваемые символы образа подчеркнуты.

Hoola-Hoola girls like Hooligans.

Hooligan

Hooligan

Hooligan

Hooligan

Рис. 1. Прямой поиск образа в строке

При программной реализации можно использовать два цикла. Внешний цикл отвечает за продвижение образа по строке, во внутреннем цикле проис­ходит продвижение по образу. Листинг данной программы выглядит сле­дующим образом:

*i = -\_1;*

*do{*

*i++;*

*j = 0;*

*while((j < m) && (str[i + j] == img[j]))*

*j++;*

*}while((j != m) && (i < n - m));*

В действительности член (i < n - m) в условии окончания определяет, не закончилась ли строка и в случае отрицательного ответа свидетельствует, что нигде в строке совпадения с образом не произошло. В то же время невы­полнение условия *(j* != m) соответствует ситуации, когда соответствие образа некоторой части строки найдено.

**Эффективность прямого поиска в строке.** Данный алгоритм работает достаточно эффективно, если допустить, что несовпадение пары символов происходит, по крайней мере, после всего лишь нескольких сравнений во внутреннем цикле. Можно предполагать, что для текстов, составленных из 128 печатных символов, несовпадение будет обнаруживаться после одной или двух проверок. Тем не менее, в худшем случае производительность будет внушать опасение. Например, для строки, состоящей из *N* - 1 символов *А* и единственного *В,* а образа, содержащего *М - 1* символов *А* и опять *В*, чтобы обнаружить совпадение в конце строки, требуется произвести порядка *N* \* *M* сравнений. Далее рассматриваются два алгоритма, которые намного эффек­тивнее решают задачу поиска образа в строке.

1. **Алгоритм Кнута, Морриса и Пратта**

В 1970 г. Д. Кнут, Д. Моррис и В. Пратт изобрели алгоритм, фактически требующий только *N* сравнений даже в самом плохом случае. Новый ал­горитм основывается на том соображении, что, начиная каждый раз сравне­ние образа с самого начала, после частичного совпадения начальной части образа с соответствующими символами строки пройденная часть строки фак­тически известна и можно «вычислить» некоторые сведения (на основе само­го образа), с помощью которых можно продвинуться по строке дальше, чем при прямом поиске.

Алгоритм Кнута, Морриса и Пратта или КМП-алгоритм использует при сдвиге образа таблицу *d*, которая формируется еще до начала поиска об­раза в строке. Можно выделить два этапа работы КМП-алгоритма:

1. формирование таблицы *d*, используемой при сдвиге образа по строке;
2. поиск образа в строке.

Таблица *d* формируется на основе образа и содержит значения, которые в дальнейшем будут использованы при вычислении величины сдвига образа. Размер данной таблицы равен длине образа, которую можно определить при помощи функции *int strlen(char* \*) библиотеки *<string.h>.* Таким образом, таблица *d* фактически является одномерным массивом, состоящим из числа элементов равного количеству символов в образе.

Первый элемент массива *d* всегда равен -1. Все элементы массива *d* соответствующие символам одинаковым первому символу образа также прирав­ниваются -1. Как показано на рис. 2, элемент *d*[0], соответствующий первому символу образа - *a -* равен -1, также и четвертый элемент d[3], также соответ­ствующий символу *a* равен -1.

образ: a b c a b c

значения элементов массива *d*: -1 -1

Рис. 2. Образ и значения некоторых элементов массива *d*

Для остальных символов образа значение элементов массива *d* вычис­ляется следующим образом: значение *d[j],* соответствующее *j*-му символу образа, равно максимальному числу символов непосредственно предшест­вующих данному символу, совпадающих с началом образа. При этом если рассматриваемому символу предшествует *k* символов, то во внимание при­нимаются только *k-1* предшествующих символов.

образ: a b c a b c

значения элементов массива *d*: -1 0 0 -1 1 2

Рис. 3. Образ и значения элементов массива *d*

На рис. 3 показаны значения элементов массива *d*. Отметим, что пято­му символу образа, символу *b*, предшествует символ *a,* совпадающий с пер­вым символом образа, поэтому *d[4],* соответствующий символу *b*, равен 1. Аналогично, *d[5]* равен 2, поскольку символу *с* предшествуют два символа *a* и *b,* совпадающие с двумя первыми символами образа.

Поскольку *d[j]* принимает значение равное максимальному числу сим­волов предшествующих *j*-му символу, то, как показано на рис. 4, элемент *d[6],* соответствующий символу с, равен четырем, а не двум.

образ: a b a b a b c

значения элементов массива *d*: 4

Рис. 4. Образ и значения элемента *d*[6]

Можно сделать следующий вывод: значения массива *d* определяется одним лишь образом и не зависит от строки текста. Для определения значенияэлементов массива *d* необходимо найти самую длинную последовательность символов образа, непосредственно предшествующих позиции j, которая сов­падает полностью с началом образа. Так как эти значения зависят только от образа, то перед началом фактического поиска необходимо вычислить эле­менты массива *d.* Эти вычисления будут оправданными, если размер строки или текста значительно превышает размер образа.

Вторым этапом работы КМП-алгоритма является сравнение символов образа и строки и вычисления сдвига образа в случае их несовпадения. Символы образа рассматриваются слева направо, т. е. от начала к концу образа. При несовпадении символов образа и строки образ сдвигается вправо по строке. Величина сдвига вычисляется следующим образом: если при пе­реборе символов образа используется индекс j, то сдвиг образа происходит на *j - d[j]* символов. На рис. 5 приведен ряд примеров иллюстрирующих сдвиг образа при работе КМП-алгоритма.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Строка | **. . .** | ***a*** | ***a*** | ***a*** | ***a*** | ***a*** | ***c*** | **. . .** |
| Образ | | ***a*** | ***a*** | ***a*** | ***a*** | ***a*** | ***b*** |  |
| *j* | | 0 | 1 | 2 | 3 | 4 | 5 |  |
| *d*[*j*] | | -1 | -1 | -1 | -1 | -1 | 4 |  |
| Сдвинутый | |  | ***a*** | ***a*** | ***a*** | ***a*** | ***a*** | ***b*** |

образ

*j* = 5, d[5] = 4,

смещение = *j –* d[j] = 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **. . .** | ***a*** | ***b*** | ***c*** | ***a*** | ***b*** | ***d*** | **. . .** |  |
|  | ***a*** | ***b*** | ***c*** | ***a*** | ***b*** | ***c*** |  | *j* = 5, d[5] = 2, |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | смещение = *j - d[j]=3* |
| -1 | 0 | 0 | -1 | 1 | 2 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | ***a*** | ***b*** | ***c*** | ***a*** | ***b c*** |

Строка

Образ

|  |
| --- |
| *j* |
| *d*[*j*] |

Сдвинутый

образ

Рис. 5. Частичное совпадение и смещение образа

Приведем пример поиска в строке образа «*Hooligan*». На рис. 6 приве­дены значения элементов массива *d.* На рис. 7 показан принцип работы КМП-алгоритма, сравниваемые символы подчеркнуты.

образ: H o o l i g a n

значения элементов массива d: -1 0 0 0 0 0 0 0

Рис. 6. Образ и значения элементов массива *d*

Hoola-Hoola girls like Hooligans.

Hooligan

Hooligan

Hooligan

Hooligan

Hooligan

. . . .

Hooligan

Рис. 7. Поиск образа в строке по методу Кнута, Морриса и Пратта

Обратите внимание: при каждом несовпадении символов образ сдвига­ется на все пройденное расстояние, поскольку меньшие сдвиги не могут при­вести к полному совпадению.

**Эффективность КМП-алгоритма.** Точный анализ КМП-поиска, как и сам его алгоритм, весьма сложен. Его разработчики доказывают, что требует­ся порядка *М* + *N* сравнений символов, что значительно лучше, чем *M* \* *N* сравнений из прямого поиска. Они так же отмечают то приятное свойство, что указатель сканирования строки *i* никогда не возвращается назад, в то время как при прямом поиске после несовпадения просмотр всегда начинает­ся с первого символа образа и поэтому может включать символы строки, ко­торые уже просматривались ранее. Это может привести к затруднениям, если строка читается из вторичной памяти, ведь в этом случае возврат обходится дорого. Даже при буферизованном вводе может встретиться столь большой образ, что возврат превысит емкость буфера.

* 1. **Алгоритм Боуера и Мура**

Поиск образа в строке по методу Кнута, Морриса, Пратта дает выигрыш только в том случае, когда несовпадению символов из образа и строки предшествовало некоторое число совпадений. Если при сравнении образа и анализируемой части строки сопоставляемые символы различны, то образ сдвигается только на один символ. Продвижение образа более чем на едини­цу происходит лишь в этом случае, когда происходит совпадение нескольких символов образа и строки. К несчастью, это скорее исключение, чем правило: совпадения встречаются значительно реже, чем несовпадения. Поэтому вы­игрыш от использования КМП-стратегии в большинстве случаев поиска в обычных текстах весьма незначителен.

В 1975 г. Р. Боуер и Д. Мур предложили метод, который не только улучшает обработку самого плохого случая, но дает выигрыш в промежуточ­ных ситуациях. Скорость в алгоритме Бойера-Мура достигается за счет того, что удается пропускать те части текста, которые заведомо не участвуют в ус­пешном сопоставлении. Данный алгоритм, называемый БМ-поиском, осно­вывается на необычном соображении - сравнение символов начинается с конца образа, а не с начала.

Как и при работе КМП-алгоритма, перед началом поиска образу сопос­тавляется таблица *d*, используемая в дальнейшем при смещении образа по строке. При создании матрицы *d* используется таблица кодов символов ASCII. Любой печатный или служебный символ имеет свой код. Например, код символа *n* - 110, *g* - 103, код пробела - 32. Коды ASCII печатных симво­лов приведены в табл. 1. Таблица ASCII содержит 256 символов. Поэтому матрицу *d* можно объявить, как одномерный целочисленный массив, состоя­щий из 256 элементов: *int* d[256].

Таблица 1

Таблица ASCII. Печатные символы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Код | Символ | Код | Символ | Код | Символ | Код | Символ | Код | Символ | Код | Символ | Код | Символ |
| 32 | **пробел** | 55 | 7 | 78 | **N** | 101 | **e** | 124 | **1** | 208 | **Р** | 231 | **з** |
| 33 | **!** | 56 | **8** | 79 | **O** | 102 | **f** | 125 | **}** | 209 | **С** | 232 | **и** |
| 34 | “ | 57 | **9** | 80 | **P** | 103 | **g** | 126 | **~** | 210 | **Т** | 233 | **й** |
| 35 | **#** | 58 | **:** | 81 | **Q** | 104 | **h** |  |  | 211 | **У** | 234 | **к** |
| 36 | **$** | 59 | **;** | 82 | **R** | 105 | **i** | 168 | **Ё** | 212 | **Ф** | 235 | **л** |
| 37 | **%** | 60 | **<** | 83 | **S** | 106 |  | 184 | **ё** | 213 | **Х** | 236 | **м** |
| 38 | **&** | 61 | **=** | 84 | **T** | 107 | **k** |  |  | 214 | **Ц** | 237 | **н** |
| 39 | **'** | 62 | **>** | 85 | **U** | 108 | **1** | 192 | **А** | 215 | **Ч** | 238 | **о** |
| 40 | **(** | 63 | **?** | 86 | **V** | 109 | **m** | 193 | **Б** | 216 | **Ш** | 239 | **п** |
| 41 | **)** | 64 | **@** | 87 | **W** | 110 | **n** | 194 | **В** | 217 | **Щ** | 240 | **р** |
| 42 | **\*** | 65 | **A** | 88 | **X** | 111 | **o** | 195 | **Г** | 218 | **Ъ** | 241 | **с** |
| 43 | **+** | 66 | **B** | 89 | **Y** | 112 | **P** | 196 | **Д** | 219 | **Ы** | 242 | **т** |
| 44 | **,** | 67 | **C** | 90 | **Z** | 113 | **q** | 197 | **Е** | 220 | **Ь** | 243 | **у** |
| 45 | **-** | 68 | **D** | 91 | **[** | 114 | **r** | 198 | **Ж** | 221 | **Э** | 244 | **ф** |
| 46 | **.** | 69 | **E** | 92 | **\** | 115 | **s** | 199 | **З** | 222 | **Ю** | 245 | **х** |
| 47 | **/** | 70 | **F** | 93 | **]** | 116 | **t** | 200 | **И** | 223 | **Я** | 246 | **ц** |
| 48 | **0** | 71 | **G** | 94 | **А** | 117 | **u** | 201 | **Й** | 224 | **а** | 247 | **ч** |
| 49 | **1** | 72 | **H** | 95 | \_ | 118 | **v** | 202 | **К** | 225 | **б** | 248 | **ш** |
| 50 | **2** | 73 | **I** | 96 | ` | 119 | **w** | 203 | **Л** | 226 | **в** | 251 | **ы** |
| 51 | **3** | 74 | **J** | 97 | **a** | 120 | **x** | 204 | **М** | 227 | **г** | 252 | **ь** |
| 52 | **4** | 75 | **K** | 98 | **b** | 121 | **y** | 205 | **Н** | 228 | **д** | 253 | **э** |
| 53 | **5** | 76 | **L** | 99 | **c** | 122 | **z** | 206 | **О** | 229 | **е** | 254 | **ю** |
| 54 | **6** | 77 | **M** | 100 | **d** | 123 | **{** | 207 | **П** | 230 | **ж** | 255 | **я** |

Первоначально всем элементам матрицы *d* присваивается значение равное длине образа. Длину образа можно получить, используя функцию *int strlen(char* \*) из библиотеки *<string.h>.*

Следующим шагом является присвоение каждому элементу таблицы d, индекс которого равен коду ASCII текущего рассматриваемого символа об­раза, значения равного удаленности текущего символа от конца образа.

Рассмотрим формирование таблицы *d* на примере образа «*Hooligan*». Поскольку данный образ содержит 8 символов, то его длина равна 8.

Соответственно, на начальном этапе все элементы массива *d* инициализиру­ются числом 8:

d[0] = d[1] = ... = d[254] = d[255] = 8

Далее происходит присваивание элементам массива *d* соответствую­щих значений, равных расстоянию от рассматриваемого символа до конца образа. При этом индекс элемента массива d, который получает новое значе­ние, определяется кодом ASCII рассматриваемого символа. Так, код ASCII последнего символа образа *«Hooligan»* - *n* - равен 110. Поскольку данный символ является последним, то удаленность от конца образа равна нулю. Таким образом:

d[110] = 0;

также на языке программирования Си можно записать:

d[‘n’] = 0;

Код ASCII предпоследнего символа образа *«Hooligan»* - *a* - равен 97. Удаленность данного символа от конца образа равна 1 и следовательно:

d[97] = 1;

или

d[‘a’] = 1;

Аналогичным образом изменяются значения элементов таблицы d, соответствующие символам образа g, i, l, o:

d[‘g’] = 2; или d[103] = 2;

d[‘i’] = 3; или d[105] = 3;

d[‘l’] = 4; или d[108] = 4;

d[‘o’] = 5; или d[111] = 5;

В том случае, когда образ содержит несколько одинаковых символов, элементу таблицы d, соответствующему данному символу, присваивается значение равное удаленности от конца образа самого правого из одинаковых символов. Так, образ *«Hooligan»* содержит два символа o, удаленность от конца образа первого из них равно шести, удаленность второго - пять. В этом случае d[‘o’] будет равно пяти.

В рассматриваемом примере присваивание значений элементам масси­ва *d* происходит при продвижении по образу справа налево. Таким образом, легко определить, был ли уже рассмотрен тот или иной символ, выполнив проверку на содержимое соответствующего элемента массива d: если эле­мент содержит значение равное длине образа, значит данный символ еще не рассматривался. Поэтому, когда будет рассматриваться второй символ образа *«Hooligan»,* символ o, удаленность которого от конца образа равна шести, проверка содержимого соответствующего элемента массива *d* покажет, что d[‘o’] не равно длине образа - 8, что свидетельствует о том, что символ *o* уже встречался, и d[‘o’] изменяться не должно.

Следует отметить также, что элементу массива d, соответствующего символу H, должно быть присвоено значение семь, т. е. фактическая удален­ность символа *H* от конца образа:

d[‘H’] = 7; или d[72] = 7;

На рис. 8 показан образ и значения элементов массива d, соответст­вующие символам данного образа.

образ: H o o l i g a n

значения элементов массива d: 7 5 5 4 3 2 1 0

Рис. 8. Образ и значения элементов массива d, соответствующие символам образа

Вторым этапом работы алгоритма БМ-поиска после построения табли­цы *d* является собственно сам поиск образа в строке. При сравнении образа со строкой образ продвигается по строке слева направо, однако посимволь­ные сравнения образа и строки выполняются справа налево.

Сравнение образа со строкой происходит до тех пор, 1) пока не будет рассмотрен весь образ, что говорит о том, что соответствие между образом и некоторой частью строки найдено, 2) пока не закончится строка, что значит, что вхождений, соответствующих образу в строке нет, 3) либо пока не про­изойдет несовпадения символов образа и строки, что вызывает сдвиг образа на несколько символов вправо и продолжение процесса поиска.

В том случае, если произошло несовпадение символов, смещение об­раза по строке определяется значением элемента таблицы d, причем индек­сом данного элемента является код ASCII символа *строки.* Подчеркнем, что, несмотря на то, что массив *d* формируется на основе образа, при смещении индексом служит символ из строки. На рис. 9. показан пример работы алго­ритма БМ-поиска, сравниваемые символы подчеркнуты.

Hoola-Hoola girls like Hooligans.

Hooligan

Hooligan

Hooligan

Hooligan

Hooligan

Рис. 9. Поиск образа в строке по методу Боуера и Мура

На первой итерации произошло несовпадение символа образа *n* и сим­вола строки o. Значение элемента d[‘o’] равно пяти. Поэтому образ смещает­ся вправо на пять символов. Если бы произошло совпадение этих двух сим­волов, то далее рассматривался бы предпоследний символ образа и соответ­ствующий ему символ строки и т. д.

При очередном сравнивании образа и строки происходит несовпадение символов *n* и *g.* Опять же используя в качестве индекса элемента массива *d* код ASCII символа строки g, получаем значение два: d[‘g’] = 2. Образ сдвига­ется на два символа вправо. Таким образом, образ постепенно сдвигается по строке до тех пор, пока образ в строке не будет найден или не кончится строка.

**Эффективность БМ-алгоритма.** Замечательным свойством БМ- поиска является то, что почти всегда, кроме специально построенных приме­ров, он требует значительно меньше *N* сравнений. В самых же благоприят­ных обстоятельствах, когда последний символ образа всегда попадает на не­совпадающий символ строки, число сравнений равно *N* / M.

Авторы приводят и несколько соображений по поводу дальнейших усовершенствований алгоритма. Одно из них - объединить приведенную только что стратегию, обеспечивающую большие сдвиги в случае несовпаде­ния, со стратегией Кнута, Морриса и Пратта, допускающей «ощутимые» сдвиги при обнаружении совпадения (частичного). Такой метод требует двух таблиц, получаемых при предтрансляции: d1 - только что упомянутая табли­ца, a d2 - таблица, соответствующая КМП-алгоритму. Из двух сдвигов выби­рается больший, причем и тот и другой «говорят», что никакой меньший сдвиг не может привести к совпадению.