デジタルサーチャー講座

検索を科学する

塩田 紳二

第2回 検索の基本動作

インターネット検索や文字列検索、世の中には、いろいろな検索があるが、その基本になるのはコンピュータのメモリー上での検索処理である。つまり、メモリーの中にあるデータから特定のパターンを見つけるという処理である。今回は、簡単な検索を例にして、コンピュータ内部での検索処理を見ていくことにする。

検索パターンの指定

検索とは、検索の対象となるデータと、検索したい情報を使って行う処理のことだ。簡単にいえば、対象となるデータの中から指定されたパターンと一致するものだけを見つけるというのが検索である。実際の検索では、その対象は、文字や文字列だけでなく、複雑なものになることがある。しかし、コンピュータのメモリーの中では、どれも、2進法で表現されたビットパターンにすぎない。以後の検索例では、文字列を使って対象となるデータや検索パターンを指定しているが、これは、図や文章による表現が簡単だからで、文字列の場合だけを解説しているわけではない。

さて、検索の一番簡単なパターンは、見つけたい文字列そのものを指定することだ。たとえば、次のように複数の英単語からなるデータがあったとする。

```
the
rain
in
spain
stays
mainly
in
the
plain
```

検索の対象となるデータは、通常、複数の要素に分かれていることが多い。たとえば、上の例では、データは単語ごとに分かれている。データベースでは、個々の要素をレコードと呼ぶ。このためこうした検索の場合でも個々の要素をレコードと呼ぶこともある。しかし、データベースの場合、レコードは同じ形式であることを仮定しているため、一般的な検索では、レコードと呼ばないこともある。ここでは、単に「要素」と呼ぶことにしよう。

この単語データから、“ rain ”、“ the ”といった単語を探すのが、「完全一致」による検索である。

完全一致の場合、検索パターンとデータの各要素（この場合は個々の単語）の一致をデータの先頭から調べていく。すべてのデータを調べても一致したものがなければ、検索は失敗したことになる。たとえば、上記の例で、“ cat ”という検索パターンを探しても見つからない。

単語が検索パターンと一致するかどう

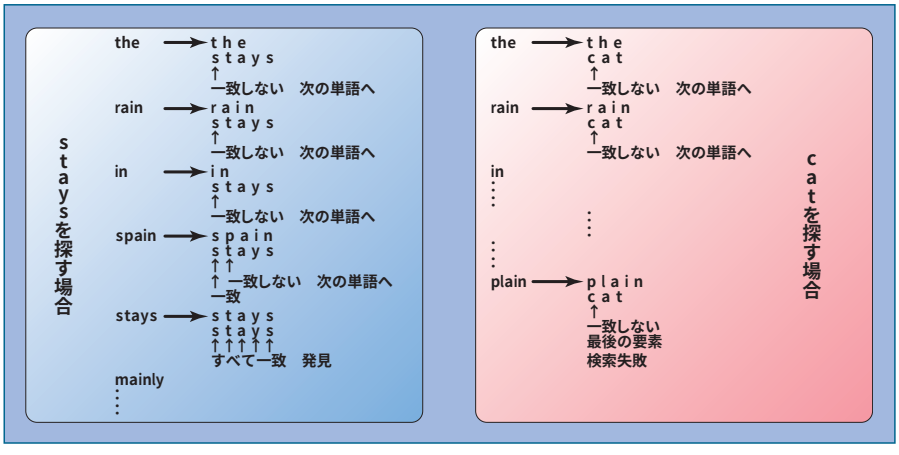

図1 完全一致での検索パターン

かは、単語の各文字と検索パターンの各文字を順に調べていく方法で行われる(図1)。

前方一致、後方一致、部分一致

完全一致は、検索の基本だが、人間が使うには、ちょっと不便なこともある。なぜなら、人間は必ずしも検索対象を明確に記述できるとは限らないからである。このような単語のデータの場合、たとえば、最初が“ S ”の単語とか、最後が“ in ”だったというあいまいな記憶だけが残っていて、それを見つけたいという場合がある。

最初が“ sp ”で始まるもののように、見つける要素の前半だけを検索パターンに指定することを「前方一致」による検索という。先ほどの例では、“ spain ”が該当する。

具体的にはどうやって検索しているのかというと、基本は完全一致の場合と同じだが、単語全体を調べないだけである。検索パターンは、上記の例では2文字しかないので、各単語の先頭の2文字が一致するかどうかを調べていく(図2)。

逆に、最後が“ in ”になる単語というような検索を行うことを「後方一致」と呼ぶ。この場合は、各単語の最後の2文字が一致するかどうかを調べる(図3)。前方一致と違う点は、単語によって長さが違うために、最後の2文字を見つけるのには手間がかかるという点である。

もう1つのパターンとして、途中に“ ai ”を含む単語という検索方法も考えられる。このような検索を「部分一致」検索と呼ぶ。

さらなる検索の分解

前方一致と完全一致による検索をより細かく見ていこう。前述のように、これらは、対象となるデータと、検索パターンの双方を先頭から順に一致していくかどうかを調べることで、一致するかどうかを判定している。

この動作は、メモリー内の連続した一定の領域に置かれたデータ同士(ただし、双方のサイズは必ずしも一致しない)を順次比較する。

通常コンピュータのメモリーは、8bitや16bitといった単位でアドレスが割り当てられており、CPUはメモリーのアドレスを指定してその内容を読み書きする。パソコンなどで使われているCPUでは、アドレスは8bitをひとまとまりにしたバイト単位で割り当てられる(図4)。

前の単語検索の例でいえば、個々の単語は、それを構成する文字を連続したメモリーの領域に置けば表現できる(図5)。ただし、単語を単にメモリーに書き込んでしまうと、単語と単語がつながってしまい、区別ができなくなる。そこで、通常は、要素の終わりを意味する特殊な文字やビットパターンを使って、個々の要素を分離する。このようなものを「終端記号」や「終端文字」などと呼ぶ。どのようなビットパターンを使えばいいかは、対象となるデータによって違ってくる。たとえば文字列だけからなるデータでは、改行コードやヌルコードなどが使われる。また、対象データ自体の最後を判定するために、さらに別の終端記号が置かれることもある。

検索の基本動作とは、検索されるデータ、検索パターンの双方がメモリー上にあり、CPUは、その一致を調べるという動作である(図6)。CPUは、アドレスを1つずつ、同じ値かどうかを比較することを検索パターンすべてについて繰り返す。途中で不一致が見つかればその要素の検索は失敗で、次の要素へ移る。比較するときには、終端記号に注意し、これを越えて比較を行わないようにする。CPUによっては、これを1つの命令として実行することはできるが、動作としては、丹念にメモリーのアドレス1つ1つを比較していくしかない。

このとき、完全一致と前方一致では、どこまで比較するかの違いでしかない。終端記号の直前まで比較を行うのが完全一致検索で、検索パターンが短く、それより手前で比較が終わるのが前方一致である。この動作から見ると、完全一致による検索とは、前方一致検索の特殊な

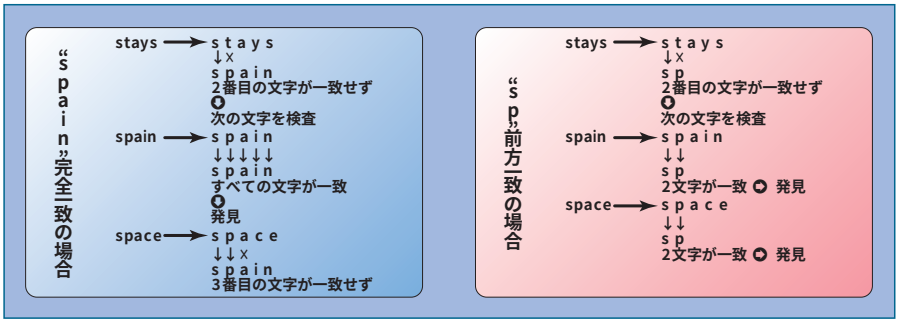

図2　前方一致での検索パターン

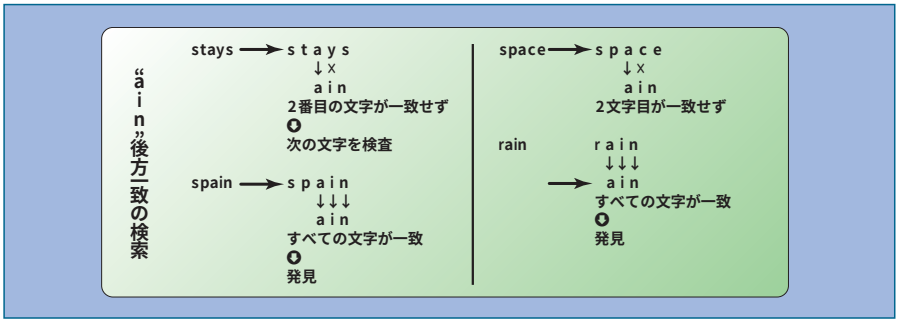

図3　後方一致での検索パターン

場合、といえるわけだ。

さらに部分一致の検索を考えてみよう。この場合、メモリー内のデータに対して、一致するところが見つかるまで、比較を始める位置をずらしながら一致を見ていく(図7)。

検索を始めるところを先頭に固定すると、前方一致や完全一致と同じことになる。つまり、部分一致検索の特殊な場合と考えることができる。

同じように、開始位置を対象データの最後と検索パターンの最後が一致するところまでずらしたのが、後方一致検索である。

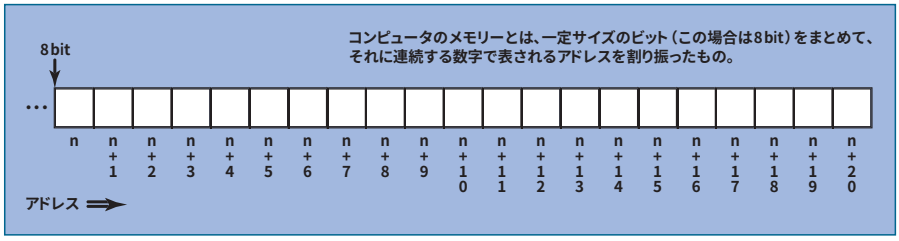

図4　アドレスの割り振り

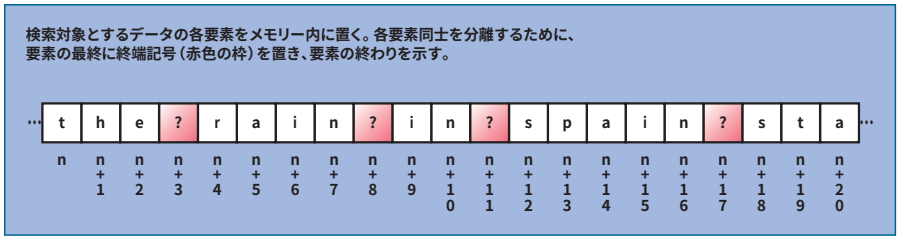

図5　検索対象のデータは終端記号で区切られる

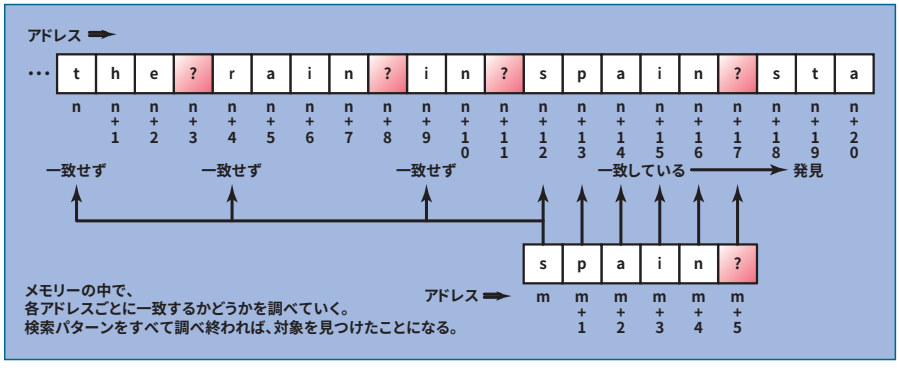

図6　検索の基本動作

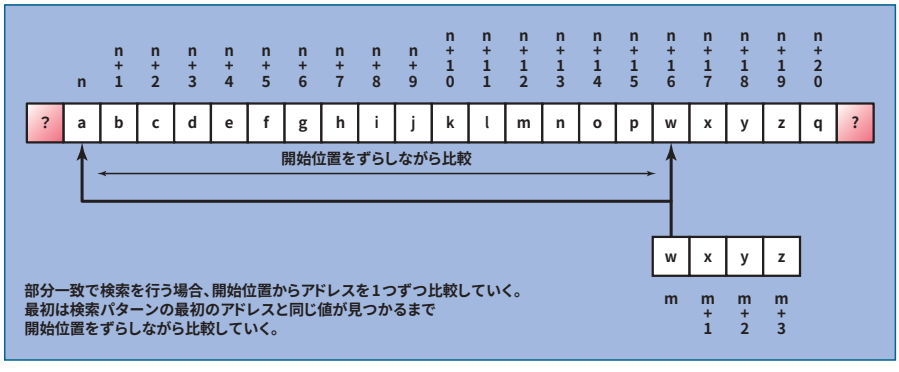

図7　部分一致検索の動作

こう考えると、あるメモリー範囲の中から、特定のパターンが連続したものを見つけるという部分一致検索の動作が、メモリー内のデータに対する検索の基本的な動作であるといえる。

検索対象データの状態

実は、検索は、対象となるデータの状態により、探し方やその効率が変わってくる。先ほどの単語データの例では、“ the ”が2つ入っている。つまり、「完全一致」の場合であっても、1つだけ見つけたからといって、必ずしもそれで終わりというわけではない。このため、どんな場合でも、最後の単語まで調べる必要がある。エディターで、“ the ”という単語をすべて他の単語に置き換えるような場合を考えてみよう。最初の1つだけが見つかっても処理が終わりとは限らないわけだ。

となると、このようなデータでは、単語の位置や検索パターンに関係なく、どんな場合でも、対象データすべてを調べることになる。対象データが少ない場合にはいいが、大量になると、検索速度が問題になる。

もし、対象データに単語が重複して登録されていることはないと仮定できるなら、検索の効率は上がる。1つでもデータが見つかれば、そこでやめることができるからだ。前述のエディターの例ではこれは無理だが、たとえば、名簿のようなそれぞれの要素が重複しないデータというのは世の中に少なくない。

これでも、前方一致の場合には、すべてのデータを探す必要がある。単語自体は重複していなくても、先頭の数文字が同じという単語がある可能性があるからだ。

では、さらに、単語がアルファベット順に並んでいると仮定できると、前方一致でも、すべてのデータを調べる必要がなくなる。たとえば、先頭が“ AB ”という単語を見つける場合、2文字目がBより後の“ C ”や“ D ”で始まる単語に遭遇すれ

ば、もうそれ以上先に、先頭が“ AB ”になる単語がないからだ(図8)。これを実現するには、データを辞書順に並べ替えてしまえばよい。

後方一致を効率的に行う方法はあるだろうか？　少なくとも、要素の並べ方といった方法では、後方一致検索を効率化することはできない。なぜなら、辞書順に並べたとしても、同じ語尾を持つ単語は全体に分散してしまうからだ。したがって、すべての要素を調べることは避けられそうにない。

では、各単語の先頭に単語の文字数が入っていたらどうだろうか？　後方一致検索は、前述したように部分一致検索の特殊な場合に相当する。このとき、後方一致となるように検索を開始する場所をすばやく決めることができれば、検索効率は多少ではあるが上がることになる。単語の先頭に単語の文字数が書かれている場合、後方一致検索を行うために比較を開始する場所は、単語の長さをn、検索パターンの長さmをとすれば、単語の先頭から、n－m＋1文字目から比較を始めればよい(図9)。

もし、単語の長さが不明の場合、終端記号を見つけるまで、単語の先頭からデータの大きさを数える必要がある。しかし、単語の先頭にその大きさが記述してあれば、この手間を省けるわけだ。

さらに、単語の長さがわかれば、前方一致検索で、比較が成功しなかったとき、次の単語へ進む処理を同じように簡単にできる。

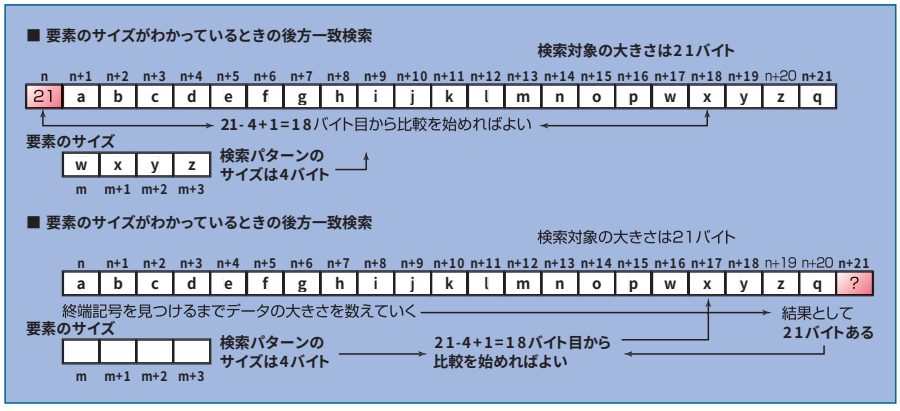

図9　効率の良い後方一致検索の方法

さらに複雑な処理

検索などでは、もう少し複雑なパターンを表現する「ワイルドカード」が使われることがある。ワイルドカードとは、どんな文字にでも、何文字でも一致する記号のことなどをいう。また、これを使った検索をワイルドカード検索ということがある。記号としてアスタリスク(＊)を使うことが多い。

これを使うと、前方一致、後方一致、部分一致は、

前方一致　pattern＊
後方一致　＊pattern
部分一致　＊pattern＊

と表現できる。たとえば、パターンを“ ABC ”としたとき、“ ＊ABC＊ ”は、

AAABCCCC
ABC
XABC
ABCX

などに一致する。注意するのは、文字数が0、つまり存在しない文字にも一致することである。たとえば、この例では、ABCなどがその例になる。“ ＊ABC＊ ”は部分一致検索であるが、その特殊な場合が前方一致検索や後方一致検索であることを考えると、存在しない文字にも一致することは妥当な解釈方法である。

さらに“ ＊ABC＊XYZ＊ ”といった複雑な検索も、“ ABC ”、“ XYZ ”の2つの部分一致検索を行うことで処理できる。つまり、最初に“ ABC ”で部分一致検索を行い、発見したら、“ ABC ”の直後から“ XYZ ”による部分一致検索が成功するかどうかで、複雑な検索を処理できる(図10)。

さまざまな技法はあるが、1つ1つ比較していくのがコンピュータの検索の基本なのである。

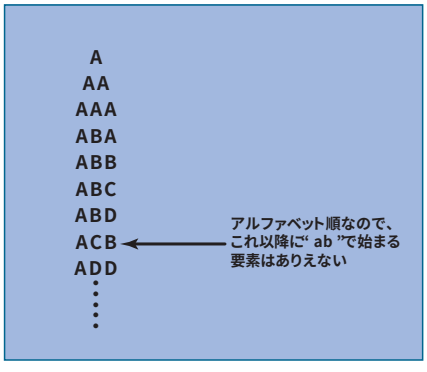

図8　“ AB ”を前方一致検索する場合

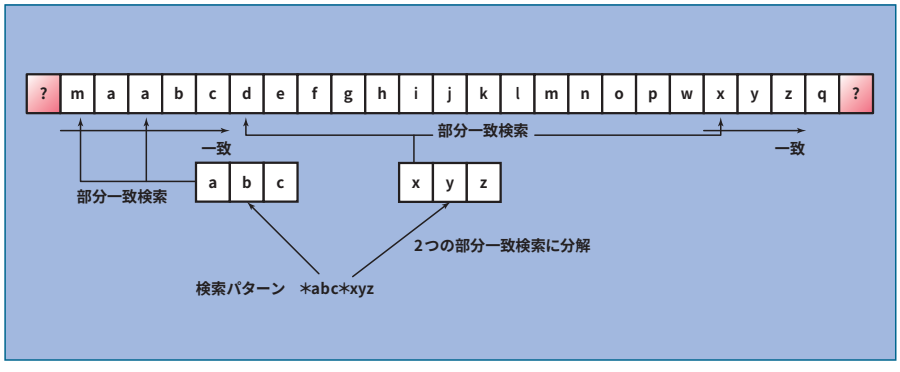

図10　ワイルドカードを使った複雑な検索であっても、これを複数の部分一致検索に分解して実行できる

［インターネットマガジン　バックナンバーアーカイブ］ ご利用上の注意

このPDFファイルは、株式会社インプレスR&D（株式会社インプレスから分割）が1994年～2006年まで発行した月刊誌『インターネットマガジン』の誌面をPDF化し、「インターネットマガジン　バックナンバーアーカイブ」として以下のウェブサイト「All-in-One INTERNET magazine 2.0」で公開しているものです。

http://i.impressRD.jp/bn

このファイルをご利用いただくにあたり、下記の注意事項を必ずお読みください。

- 記載されている内容（技術解説、URL、団体・企業名、商品名、価格、プレゼント募集、アンケートなど）は発行当時のものです。
- 収録されている内容は著作権法上の保護を受けています。著作権はそれぞれの記事の著作者（執筆者、写真の撮影者、イラストの作成者、編集部など）が保持しています。
- 著作者から許諾が得られなかった著作物は収録されていない場合があります。
- このファイルやその内容を改変したり、商用を目的として再利用することはできません。あくまで個人や企業の非商用利用での閲覧、複製、送信に限られます。
- 収録されている内容を何らかの媒体に引用としてご利用する際は、出典として媒体名および月号、該当ページ番号、発行元（株式会社インプレスR&D）、コピーライトなどの情報をご明記ください。
- オリジナルの雑誌の発行時点では、株式会社インプレス R&D（当時は株式会社インプレス）と著作権者は内容が正確なものであるように最大限に努めましたが、すべての情報が完全に正確であることは保証できません。このファイルの内容に起因する直接的および間接的な損害に対して、一切の責任を負いません。お客様個人の責任においてご利用ください。

このファイルに関するお問い合わせ先
株式会社インプレスR&D
All-in-One INTERNET magazine 編集部
im-info@impress.co.jp