問2　文字列照合処理に関する次の記述を読んで，設問1～3に答えよ。

ある文字列（テキスト）中で特定の文字列（パターン）が最初に一致する位置を求めることを文字列照合という。そのための方法として，次の二つを考える。ここで，テキストとパターンの長さは，どちらも1文字以上とする。

〔方法1　単純な照合方法〕

テキストを先頭から 1 文字ずつパターンと比較して，不一致の文字が現れたら，比較するテキストの位置（比較位置）を 1 文字分進める。この方法による処理例を図 1 に示す。

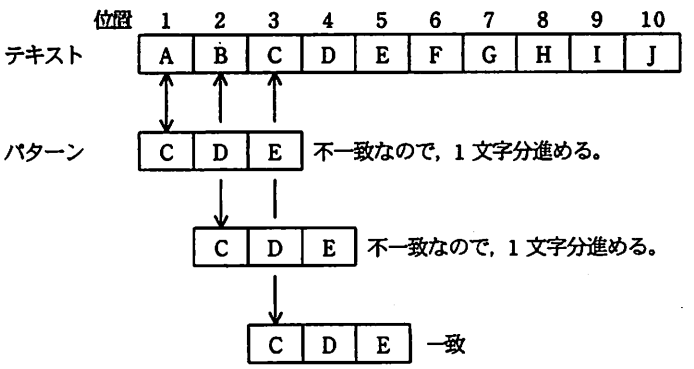


**図1　単純な照合方法**

この手順を基に，テキストとパターンを与えると，最初に一致した文字位置を返すアルゴリズムを作成した。このアルゴリズムを図2に示す。

なお，テキストは配列 T，パターンは配列 P に格納されており，T[n] 及び P[n] はそれぞれの n 番目の文字を，T.length 及び P.length はそれぞれの長さを示す。

```
function search1(T,P)
  for i を 1 から [ ア ] まで 1 ずつ増やす
    for j を 1 から [ イ ] まで 1 ずつ増やす
      if T[i+j−1]と P[j]が等しい        ← α
        if j と [ イ ] が等しい
          return i
        endif
      else
        break
      endif
    endfor
  endfor
  return −1        // パターンが見つからない場合は−1 を返す
endfunction
```

**図 2　単純な照合方法のアルゴリズム**

〔方法 2　効率的な照合方法〕

方法 1 では，パターンとテキストの文字が不一致となった場合，比較位置を 1 文字分進めている。ここでは，比較位置をなるべく多くの文字数分進めることで，照合における比較回数を減らすことを考える。

パターンの末尾に対応する位置にあるテキストの文字（判定文字）に着目すると，次のように比較位置を進める文字数（スキップ数）が決定できる。

(1)　判定文字がパターンに含まれていない場合は，判定文字とパターン内の文字の比較は常に不一致となるので，パターンの文字数分だけ比較位置を進める。また，判定文字がパターンの末尾だけに含まれている場合も，同様にパターンの文字数分だけ比較位置を進める。

(2)　判定文字がパターンの末尾以外に含まれている場合は，判定文字と一致するパターン内の文字が，テキストの判定文字に対応する位置に来るように比較位置を進める。ただし，パターン内に判定文字と一致する文字が複数ある場合は，パターンの末尾から最も近く（ただし，末尾を除く）にある判定文字と一致するパターン内の文字が，判定文字に対応する位置に来るように比較位置を進める。

この方法による処理例を図 3 に示す。

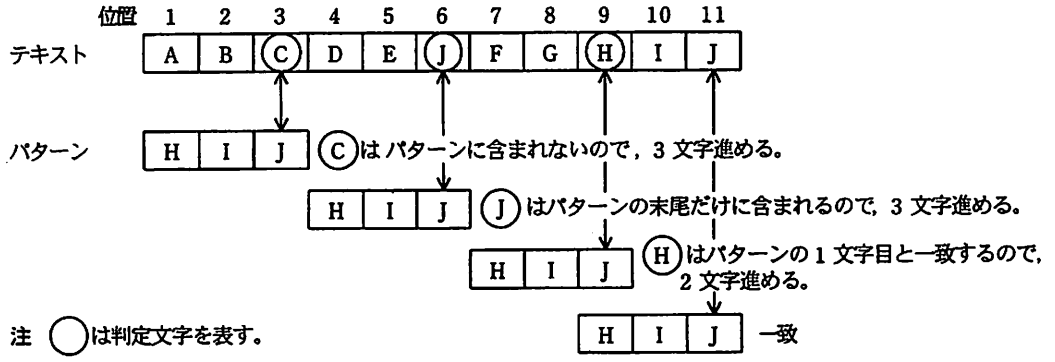


図３　効率的な照合方法

パターンが与えられると，判定文字に対応するスキップ数を一意に決定することができる。例えば，パターン HIJ に対して判定文字とスキップ数の対応表を作成すると表１のようになる。

表１　パターン HIJ に対するスキップ数

| 判定文字 | H | I | J | その他の文字 |
|---|---|---|---|---|
| スキップ数 | 2 | 1 | 3 | 3 |

この考え方を基に作成したアルゴリズムを図４に示す。

なお，テキストは配列 T，パターンは配列 P，スキップ数を決める表の判定文字は配列 C，スキップ数は配列 D に格納されており，T[n]，P[n]，C[n]，D[n]はそれぞれの n 番目の文字又は数値を，T.length 及び P.length はテキストとパターンの長さを示す。

このアルゴリズムは，三つの主要部分から成っている。一つ目は，与えられたパターンについて判定文字とスキップ数の対応表を作成する処理である。ここでは，配列 C に格納される空白文字は表 1 の“その他の文字”及びパターンの末尾の文字“J”を表現するために使用している。テキストとパターンは空白文字を含まないものとする。二つ目は，文字を比較する処理である。ここでは，現在の比較位置に対応したテキストとパターンを方法 1 と同様に 1 文字ずつ比較して，パターンに含まれる文字と対応するテキストの文字がすべて一致するか，不一致となる文字が見つかるまで繰り返す。三つ目は，判定文字とスキップ数の対応表を引いてスキップ数を決定する処理である。

```
function search2(T,P)
  for x を 1 から P.length まで 1 ずつ増やす
    C[x] ← " "                          // " "は空白文字を表す
    D[x] ← P.length
  endfor
//  (1) : 表の作成
  for j を 1 から P.length−1 まで 1 ずつ増やす
    for k を 1 から j まで 1 ずつ増やす
      if C[k]と P[j]が等しい or C[k]と" "が等しい
        C[k] ← P[j]
        D[k] ← [ ウ ]
        break
      endif
    endfor
  endfor
//  文字列を照合する
  i ← 1
  while i が [ ア ] 以下である
//  (2) : 文字の比較
    for j を 1 から [ イ ] まで 1 ずつ増やす
      if T[i+j−1]と P[j]が等しい          ← β
        if j と [ イ ] が等しい
          return i
        endif
      else
        break
      endif
    endfor
//  (3) : スキップ数の決定
    for k を 1 から P.length まで 1 ずつ増やす
      if C[k]と T[i+P.length−1]が等しい or C[k]と" "が等しい
        d ← D[k]
        break
      endif
    endfor
    i ← i+d
  endwhile
  return −1
endfunction
```

**図 4　効率的な照合方法のアルゴリズム**

設問1　図2及び図4中の［ア］～［ウ］に入れる適切な字句を答えよ。

設問2　パターン HIPOPOTAMUS に対するスキップ数を表 2 に示す。［エ］，［オ］に入れる適切な数値を答えよ。

表2　パターン HIPOPOTAMUS に対するスキップ数

| 判定文字 | H | I | P | O | T | A | M | U | S | その他の文字 |
|---|---|---|---|---|---|---|---|---|---|---|
| スキップ数 | エ | 9 | オ | 5 | 4 | 3 | 2 | 1 | 11 | 11 |

設問3　次のテキストとパターンに対して，図 2 での$\alpha$と図 4 での$\beta$の実行回数をそれぞれ答えよ。

テキスト：PICKLED_PEPPER

パターン：PEP