

アルゴリズム論 12

文字列パターン照合

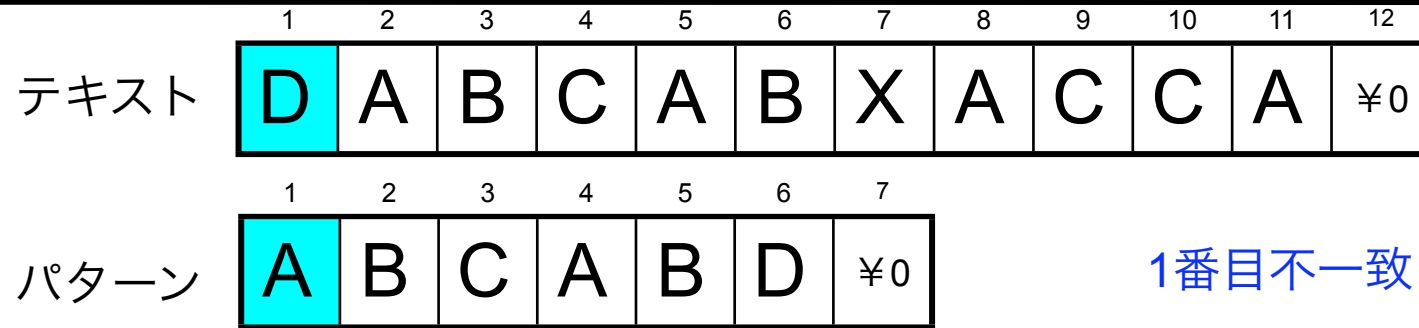
- 単純法(素朴法,力まかせ法)
- KMP法
- BM法

KMP法

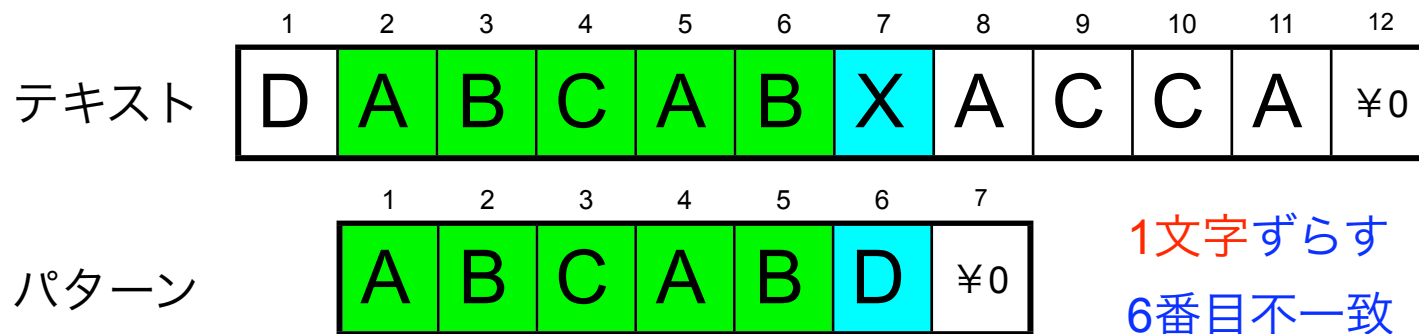
- ・単純法の欠点と想定される改良点
 - ・不一致が出ると、パターンを移動して先頭から照合する
 - ・前回の照合結果に関係なくパターンの始めから照合
 - ・前回の照合結果を有効に利用する
- ・KMP法
 - ・D.E.Knuth, J.H.Morris, V.R.Prattの3名にちなむ
 - ・KMP法の特徴
 - ・テキストとパターン中の照合している部分を記録
 - ・不一致になった場合の照合しなおす位置をスキップ表にまとめる
 - ・スキップ表に基づきパターン移動を決定する

KMP法の具体例

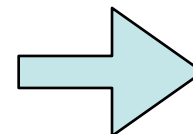
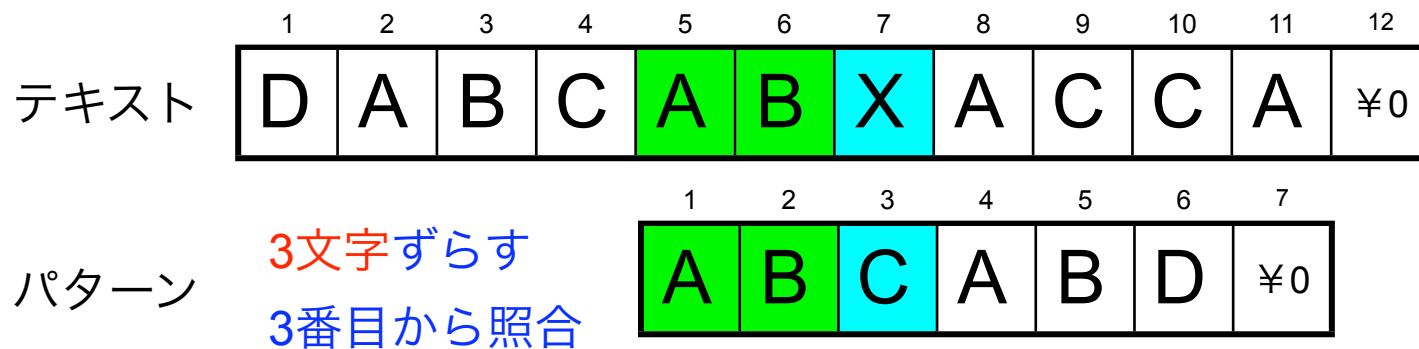
Step 1



Step 2



Step 3



パターンの照合開始場所の
表を事前に作成

KMP法のスキップ表作成1

Step 1

パターン

1	2	3	4	5	6	7
A	B	C	A	B	D	¥0

2文字目以降について

一致しない

2 再開値=0

1	2	3	4	5	6	7
A	B	C	A	B	D	¥0

Step 2

パターン

1	2	3	4	5	6	7
A	B	C	A	B	D	¥0

一致しない

3 再開値=0

1	2	3	4	5	6	7
A	B	C	A	B	D	¥0

Step 3

パターン

1	2	3	4	5	6	7
A	B	C	A	B	D	¥0

2文字一致

4 再開値=1

5 再開値=2

1	2	3	4	5	6	7
A	B	C	A	B	D	¥0

パターン4の'A'が不一致 > パターン移動後'A'をスキップして'B'から照合開始 (1スキップ)

パターン5の'B'が不一致 > パターン移動後'B'をスキップして'C'から照合開始 (2スキップ)

KMP法のスキップ表作成2

Step 4

	1	2	3	4	5	6	7	
パターン	A	B	C	A	B	D	¥0	一致しない 6 再開値=0

	1	2	3	4	5	6	7
	A	B	C	A	B	D	¥0

再開値表
(スキップ数)

	1	2	3	4	5	6	7
	A	B	C	A	B	D	¥0
	0	0	0	1	2	0	

表作成終了

パターンの中の類似の文字列を見つけ、照合時にスキップさせる

KMP法1(メイン)

```
#include <stdio.h>

int count0=0,count1=0;    /* 比較回数カウンタ count0:表作成 count1:照合 */

int kmp_match(char txt[], char pat[]); /* 関数プロトタイプ */

int main(void)
{
    int      idx;          /* 照合位置 */
    char      s1[80];       /* テキスト */
    char      s2[80];       /* パターン */

    printf(" Input text :");    /* テキスト入力 */
    scanf("%s",s1);

    printf(" Input pattern :"); /* パターン入力 */
    scanf("%s",s2);

    idx=kmp_match(s1,s2);      /* KMP法関数 */

    if (idx==-1)                /* 結果表示 */
        printf(" No pattern found in the text ¥n");
    else
        printf(" Pattern was found at %d ¥n",idx+1);

    /* 比較回数表示 */
    printf(" Number of comparison=%d+%d¥n",count0,count1);

    return(0);
}
```

KMP法2(関数1)

```
int kmp_match(char txt[], char pat[])
{
    int    pt=1; /* テキスト カーソル */
    int    pp=0; /* パターン カーソル */
    int skip[80];

    skip[pt]=0; /* スキップテーブル作成 */
    while (pat[pt] != '\0') {
        if (pat[pt]==pat[pp]) {
            skip[++pt]=++pp;
        }
        else if (pp==0) {
            skip[++pt]=pp;
        }
        else {
            pp=skip[pp];
        }
    }
}
```

KMP法2(関数1)

```

skip[pt]=0;    /* スキップテーブル作成 */           ①
while (pat[pt] !='¥0') {
    if (pat[pt]==pat[pp]) {                           ④ ⑤
        skip[++pt]=++pp;

    }
    else if (pp==0) {                                  ② ③
        skip[++pt]=pp;

    }
    else {                                             ⑥
        pp=skip[pp];
    }
}

```

	pt	pp	skip[pt]	
ABCABD				
ABCABD	1	0	0	①
ABCABD	2	0	0	②
ABCABD	3	0	0	③
ABCABD	4	1	1	④
ABCABD	5	2	2	⑤
ABCABD	6	0	0	⑥

KMP法3(関数2)

```
pt=pp=0;
while (txt[pt] !='¥0' && pat[pp] != '¥0' ) { /* 照合 */
    if (txt[pt]==pat[pp]) {
        pt++;
        pp++;

    } else if (pp==0) {
        pt++;

    } else {
        pp=skip[pp];
    }
}

if (pat[pp]=='¥0')
    return (pt-pp); /* 戻り値：照合結果 */
return(-1);
}
```

KMP法実行結果

case 2: 教科書p.109のケース

- case 1
- Input text :ABABCFGHGA
Input pattern :ABC
Pattern found at 3
Number of comparison=2+6
- case 2
- Input text :ababdababccbdcabcadb
Input pattern :ababc
Pattern was found at 6
Number of comparison=5+12
- case 3
- Input text :ABCABCABCABCABCDABC
Input pattern :ABCABCD
Pattern found at 10
Number of comparison=7+19
- case 4
- Input text :ABABCFGHGA
Input pattern :ZZ
No pattern found in the text
Number of comparison=1+11

演習問題(講義時間内で実施)

- ☑ 文字列パターン照合を行うプログラムのソースコードを入力し実行する
 - ☑ メイン(素朴法を改修)
 - ☑ KMP法
- ☑ テキストおよびパターンの文字列を入力し、実行結果を確認する

KMP 法 の特徴と計算量

KMP 法の特徴

テキストをスキャンする際は前進あるのみ

単純法では前進および後退があった

計算量：テキスト n 文字、パターン m 文字の文字列照合

- 文字の比較回数

- スキップテーブル作成の比較回数： m 回
- テキストとパターンの比較回数： n 回
- 最悪の場合 $m + n$ 回の比較

オーダ $O(m+n)$