アルゴリズム論 13

文字列パターン照合

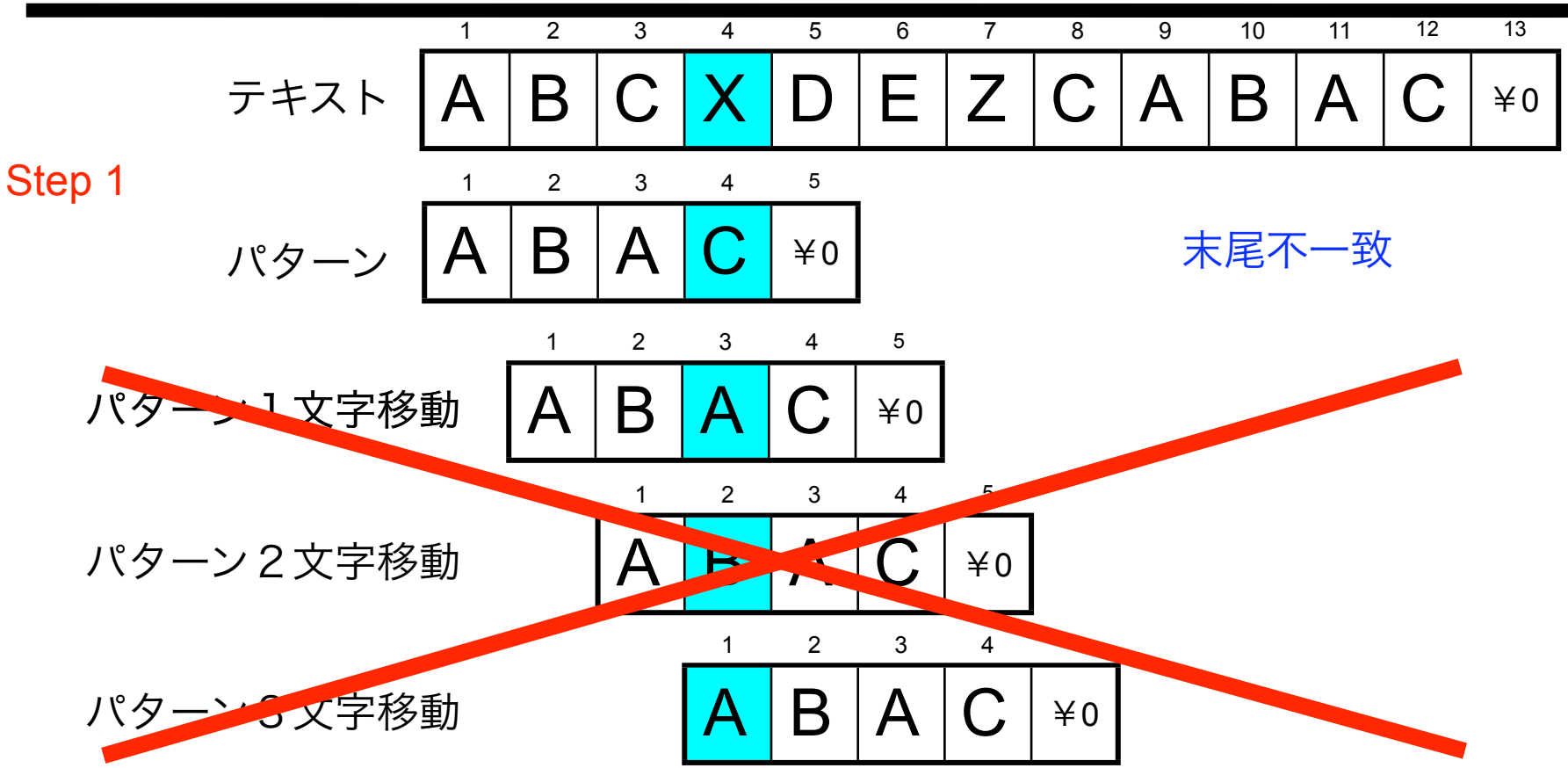
- 単純法(素朴法)
- KMP法
- BM法

BM法

- BM法
  - R.S.Boyer, J.S.Mooreの2名にちなむ
  - 平均的にKMP法よりも高速
  - BM法の特徴
    - テキストとパターン中の照合を末尾文字から開始
    - 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

26

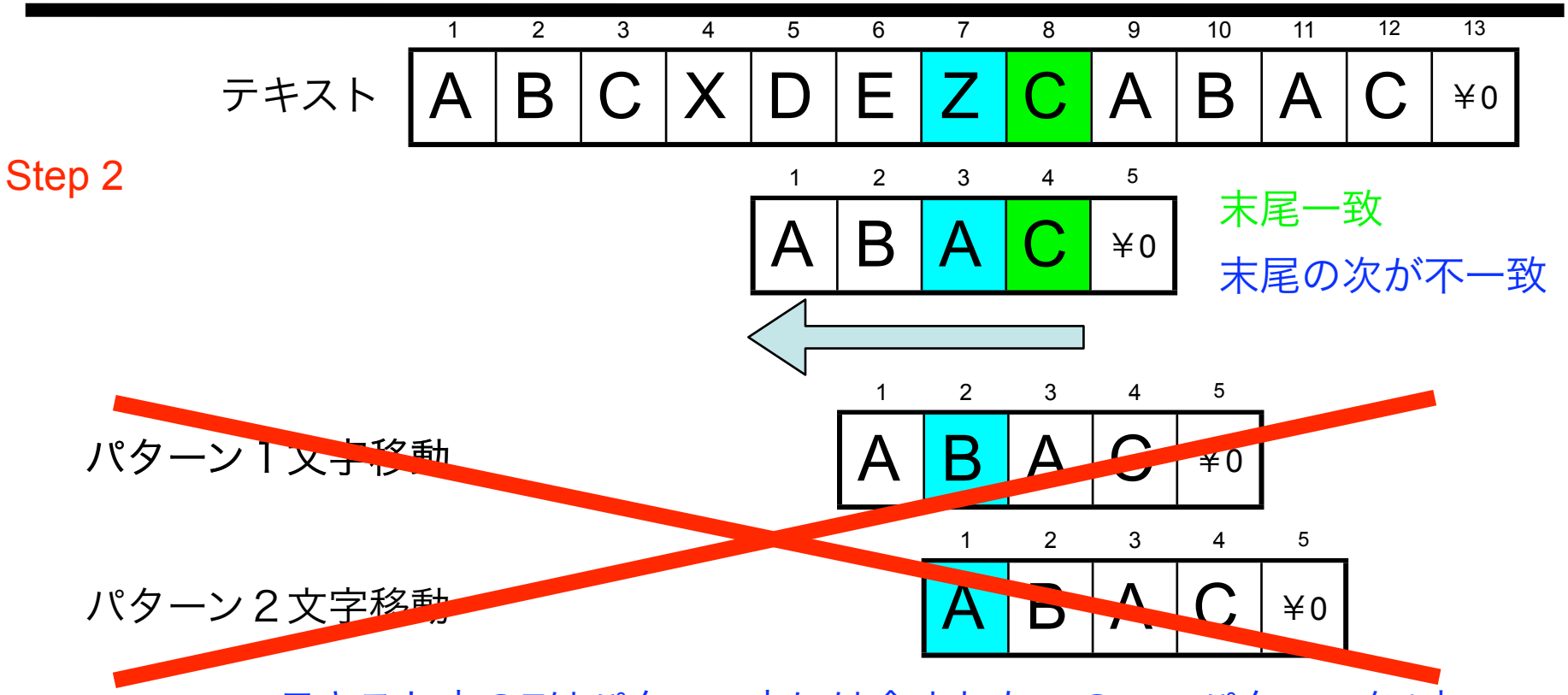
BM法の具体例1



テキスト中のXはパターン中には含まれないので、パターン1文字から3文字を移動させても一致しないのは明白

パターンの移動をスキップし、パターンを一気に4文字移動

BM法の具体例2

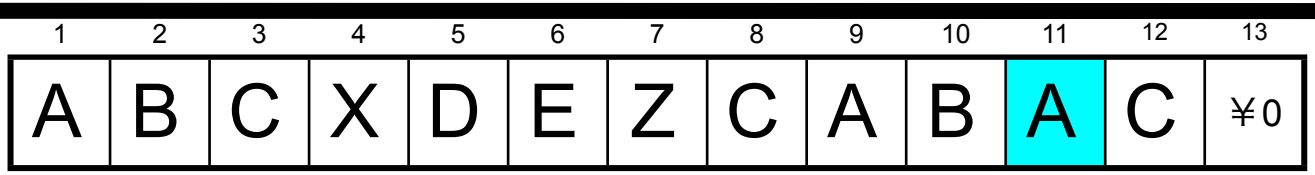


テキスト中のZはパターン中には含まれないので、パターンを1文字から2文字移動させても一致しないのは明白

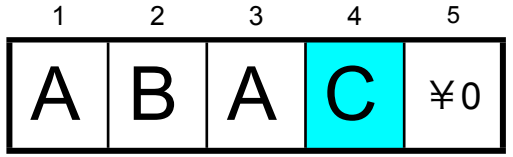
パターンの移動をスキップし、パターンを一気に3文字移動

(不一致した文字に注目する：不一致したテキスト中の文字に対して４文字移動)

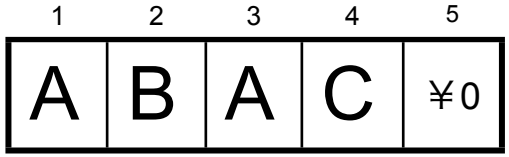
BM法の具体例3



Step 3



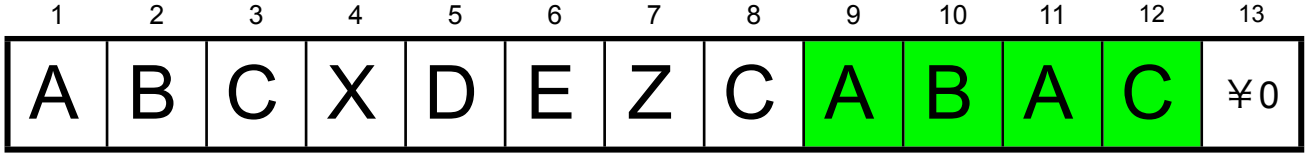
パターン3文字移動　　末尾が不一致



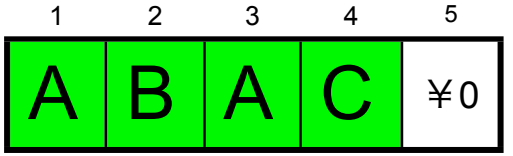
パターン１文字移動

テキスト中のAはパターン中に含まれるので、パターン移動で照合する可能性がある

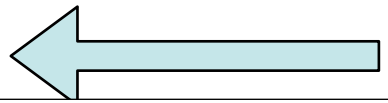
パターン中の末尾に近いAが一致するように、パターンを1文字移動



Step 4



全ての文字が照合　探索成功



BM法のスキップ表作成

事前にパターン中の文字列を調べ、移動する文字数をスキップ表を作成

スキップ表の例

A	B	その他
1	2	4

作成方法

パターン中の文字列の長さをmとする

- パターンに含まれない文字：m
- パターンに含まれる文字
  - ★末尾: m
  - ★末尾以外: m-k-1
    - k: パターン中の最後に出現した位置(配列の添字)
      - A: 3番目（添字2）　4-2-1=1
      - B: 2番目（添字1）　4-1-1=2

BM法1（メイン）

```
#include <stdio.h>
#include <string.h>
#include <limits.h>

int count=0;    /* 比較回数カウンタ  */

int bm_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=bm_match(s1,s2);  /* BM法関数  */

        if (idx==-1)            /* 結果表示  */
                printf(" No pattern found in the text ¥n");
        else
                printf(" Pattern found at %d¥n",idx+1);
        printf(" Number of comparison=%d¥n",count); /* 比較回数表示  */

        return(0);
}
```


31

BM法2(関数1)

```
int bm_match(char txt[], char pat[])
{
        int     pt;     /* テキスト  */
        int     pp;     /* パターン  */
        int txt_len=strlen(txt);  /* テキスト文字数  */
        int pat_len=strlen(pat);  /* パターン文字数  */
        int skip[UCHAR_MAX+1];    /* スキップ表の配列  */

        for (pt=0;pt<=UCHAR_MAX;pt++)        /* スキップ表初期化  */
                skip[pt]=pat_len;
        for (pt=0;pt<pat_len-1;pt++)         /* スキップ表作成  */
                skip[pat[pt]]=pat_len-pt-1;

        while (pt < txt_len) {               /* 照合 */
                pp=pat_len-1;                /* パターンの末尾  */

                while(txt[pt]==pat[pp])        {
                        if (pp==0)
                                return(pt);  /* 戻り値：照合結果  */

                        pp--;
                        pt--;

                }
                pt+=skip[txt[pt]];     /* スキップ値 パターン移動数  */
        }
        return(-1);
}
```


32

BM法実行結果

case 2: 教科書p. 111　のケース

case 1

```
Input text :ABABCDEFGHA
Input pattern :ABC
Pattern found at 3
Number of comparison=4
```

case 2

```
Input text :ababdababccbdcabcadb
Input pattern :ababc
Pattern found at 6
Number of comparison=6
```

case 3

```
Input text :ABCABCABCABCABCDABC
Input pattern :ABCABCD
Pattern found at 10
Number of comparison=10
```

case 4

```
Input text :ABABCDEFGHA
Input pattern :ZZ
No pattern found in the text
Number of comparison=5
```


33

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(素朴法またはKMP法の改修)
  - BM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

34

BM法 の特徴と計算量

BM法の特徴

テキストをスキャンする際は末尾から先頭へ

考え方

★実際の文書処理ではパターンに現れる文字より現れない文字の方が多い

-テキストとパターンの不一致の確率が高い

-不一致となったテキストの文字がパターン中に含まれない確率が高い

★パターンはテキストの文字と重なりを持たない位置まで右に移動することができる可能性が高い

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
  - パターンの長さに近い文字分だけ、パターン移動ができる確率が高い：n/m 回でテキストをスキャンできる

オーダ　O(n/m)

35