第15回 文字列処理（2）

アルゴリズムとデータ構造ならびに同演習

柏原　昭博

akihiro.kashihara@inf.uec.ac.jp

講義内容

- □　第11回目　ヒープ
- □　第12回目　ソート（整列）
- □　第13回目　クイックソート・その他のソート
- □　第14回目　文字列処理（基本）
- □　**第15回目　文字列処理（配列処理・文字探索）**

文字列処理

- 文字列の配列処理
- 文字列探索
  - 単純法
  - BM（Boyer-Moore）法
  - KMP（Knuth-Morris-Pratt）法

文字列の配列表現

- いくつかの文字列を処理する
  - 文字列の配列表現

演習15-1

- 2次元配列city[][10]に含まれる各都市名の文字列の長さと，都市名に’o’が含まれるかどうかを判定するプログラムを書きなさい．（sample15-1.c）

  ヒント：文字列の長さを求めるにはstrlen関数を用いる
  文字が含まれるかの判定はstrchr関数を用いる

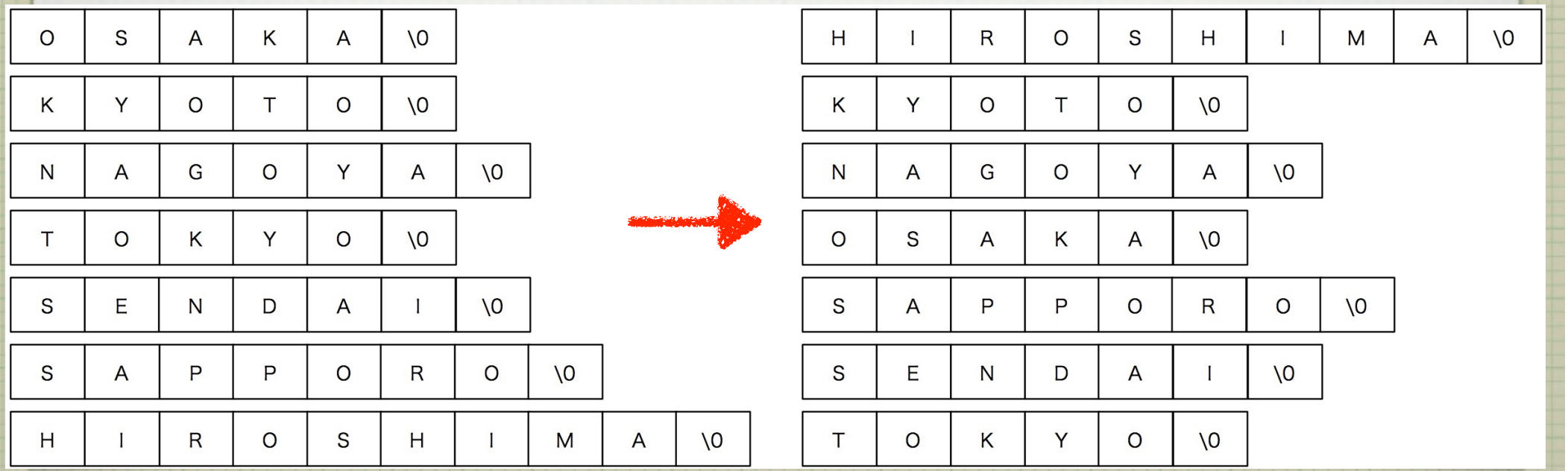
sample15-1.c

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

int main (void)
{
  char city[][10]={"osaka", "kyoto", "nagoya", "tokyo",
                   "sendai", "sapporo", "hiroshima"};

  int n=sizeof(city)/sizeof(city[0]);//文字列の個数
  char flag;
  int i;

  for (i=0;i<n;i++){//データ配列の表示
   if (strchr(city[i],'o')!=NULL)
      flag=1;//'o'が含まれる場合
   else flag=0;
   printf("city[%d]=%s lenght=%d o:%d \n",i,city[i],
                                    (int)strlen(city[i]),flag);
  }
  return 1;
}
```

演習15-2

- 2次元配列city[][10]に含まれる各都市名を昇順にソートするプログラムを作りなさい（sample15-2.c）．ソートには基本選択法を使いなさい．(sample12-2.c参考)

  ヒント：文字列の比較にはstrcmp関数を用いる
  文字列のコピーにはstrcpy関数を用いる



基本（直接）選択法

```
for (i=0; i < n-1; i++){
    min = i;
    for (j=i+1; j < n; j++)
        if (a[j] < a[min])
            min = j;
    a[i]をa[min]と交換;
}
```

...未ソート部分が無くなるまで

...未ソート部分の最小値を選択

...未ソート部分の先頭要素と交換

比較回数= (n-1)+(n-2)+(n-3)+...+2+1= n(n-1)/2

sample15-2.c
抜粋

```c
void selection_mojiretsusort(char moji[][10], int n)
{
  int i, j;
  char *toshi[n];
  char *p, *min;
  char tmp[10];
  for (i=0; i<n-1; i++){//未ソート部分が無くなるまで
   strcpy(tmp, (char *)moji[i]);
   min=moji[i];//先頭要素（最小値）へのポインタ
   for (j=i+1; j<n; j++){ //未ソート部分の最小値を選択
      if (strcmp(moji[i],moji[j])>0){
         strcpy(moji[i],moji[j]);
         min=moji[j];              } //最小値へのポインタ
      }
   strcpy( min , tmp);}
}
int main (void)
{
  char city[][10]={"osaka", "kyoto", "nagoya", "tokyo", "sendai",
                   "sapporo", "hiroshima"};
  int i, comp;
  int n=sizeof(city)/sizeof(city[0]);
  selection_mojiretsusort(city, n);//基本選択法
  for (i=0;i<n;i++)//データ配列の表示
   printf("sorted city[%d]=%s \n",i,city[i]);
  return 1;
}
```

演習15-2A

- sample15-2a.cに，各都市名を昇順にソートする関数の引数に，**文字型データへのポインタ**を用いる場合を示す．このプログラムの動作を理解しなさい．

sample15-2a.c
抜粋

```
int main (void)
{ ...
  int n=sizeof(city)/sizeof(city[0]);//文字列数
  int nn=sizeof(city[0])/sizeof(city[0][0]); //一文字列の長さ
  ...
  selection_mojiretsusort((char *)city, n, nn);//基本選択法
}
void selection_mojiretsusort(char *moji, int n, int nn)
{
  int i, j, k, mojinum;
  char *p, *ptr, *min;
  char tmp[10];
  mojinum=0;
  for (i=0; i<n-1; i++){//未ソート部分が無くなるまで
   p = moji+mojinum;//各文字列の先頭へのポインタ
   strcpy(tmp, p);
   ptr=p;//未ソート部分の文字列へのポインタ
   min=p;
   for (j=i+1; j<n; j++){ //未ソート部分の最小値を選択
         ptr+=nn;
     if (strcmp(p,ptr)>0){
        strcpy(p,ptr);
        min=ptr;
     }}
   strcpy(min, tmp);   mojinum+=nn;
}}
```

課題15-1

- 演習15-2で作成したプログラムにおいて，基本選択法によるソートをクイックソートに置き換えて都市名を昇順に並び替えるプログラムを作りなさい（ex15-1.c）なお，課題13-3で作成したように，文字列配列の先頭要素，中央要素，末尾要素の３つのうち中央値を持つ要素を軸とすること

文字列の探索

2次元配列

- ある文字列（テキスト）の中に別の文字列（パターン）が存在するかどうかを調べる
- 例

  テキスト1：ababdabeabcdabce

  パターン1：abc

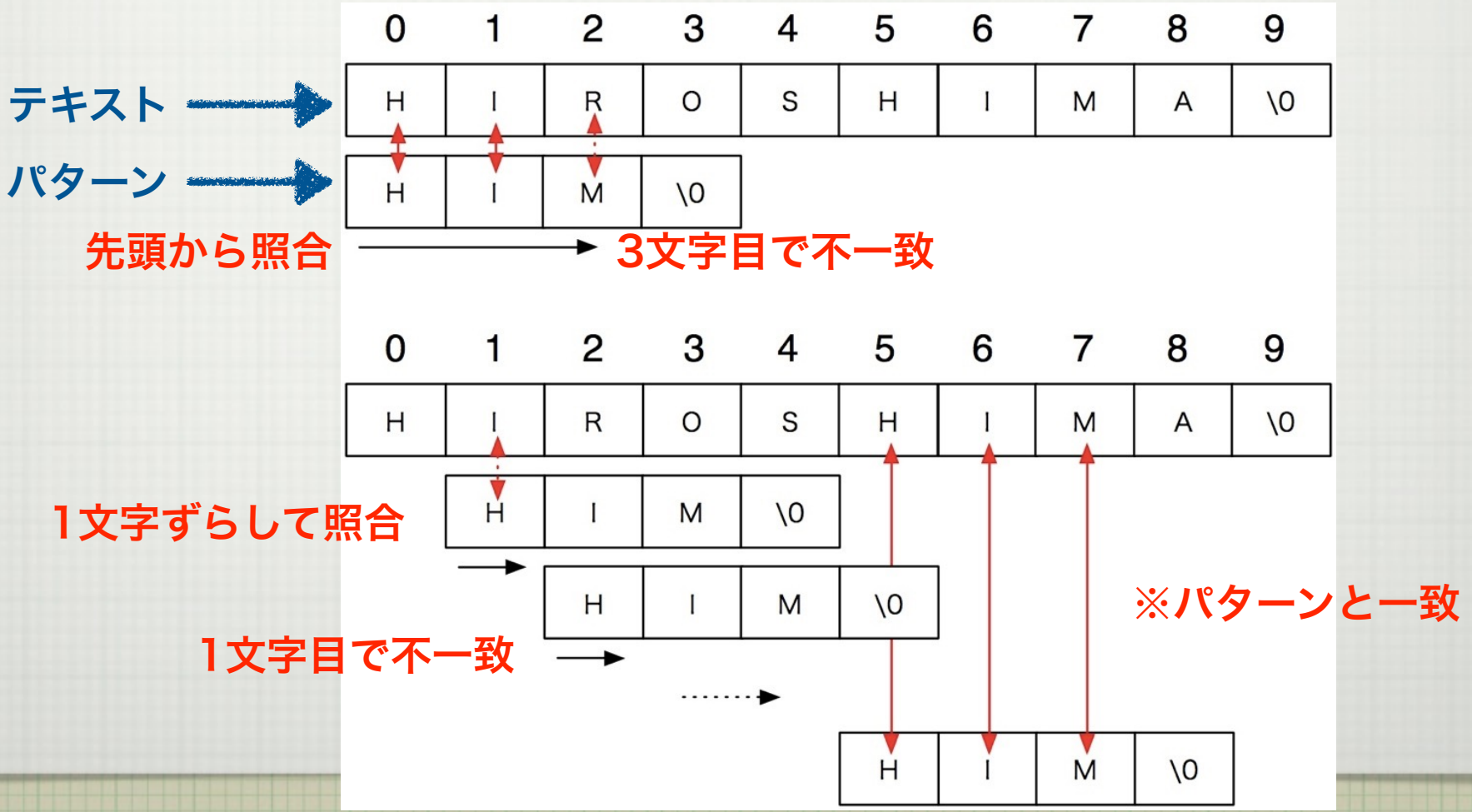
  テキスト2：This is a pen. That is a pencil.

  パターン2：pencil

文字列探索：単純法

2次元配列

- テキストの先頭から順にパターンと照合する．文字が一致しない場合は1文字ずつずらしながら照合を繰り返す



演習15-3

- 文字列探索の単純法によって，テキスト中にパターンが存在すれば，テキストの先頭から何文字目に現れるかを求めるプログラムを作りなさい（sample15-3.c）
- 例：
  テキスト：universityofelectrocommunications
  パターン：electro
  出力：パターンは13文字目に見つかりました.

sample15-3.c
抜粋

```
int p_match(char *txt, char *pattern)
{
  int ptrtxt=0;
  int ptrpat=0;


  while (txt[ptrtxt]!='\0' && pattern[ptrpat]!='\0')
   if (txt[ptrtxt]==pattern[ptrpat]) {
      ptrtxt++; ptrpat++;
   }
   else {
      ptrtxt=   ptrtxt-ptrpat+1
      ptrpat=   0
   }
  if (pattern[ptrpat]=='\0')
   return(ptrtxt-ptrpat);
  return -1;
}
```

演習15-4

- 文字列探索の単純法によって，テキスト中にパターンが何回現れるかを求めるプログラムを作りなさい（sample15-4.c）

- 例：
  テキスト：universityofelectrocommunisityoo
  パターン：sity
  出力：パターンは2個見つかりました.

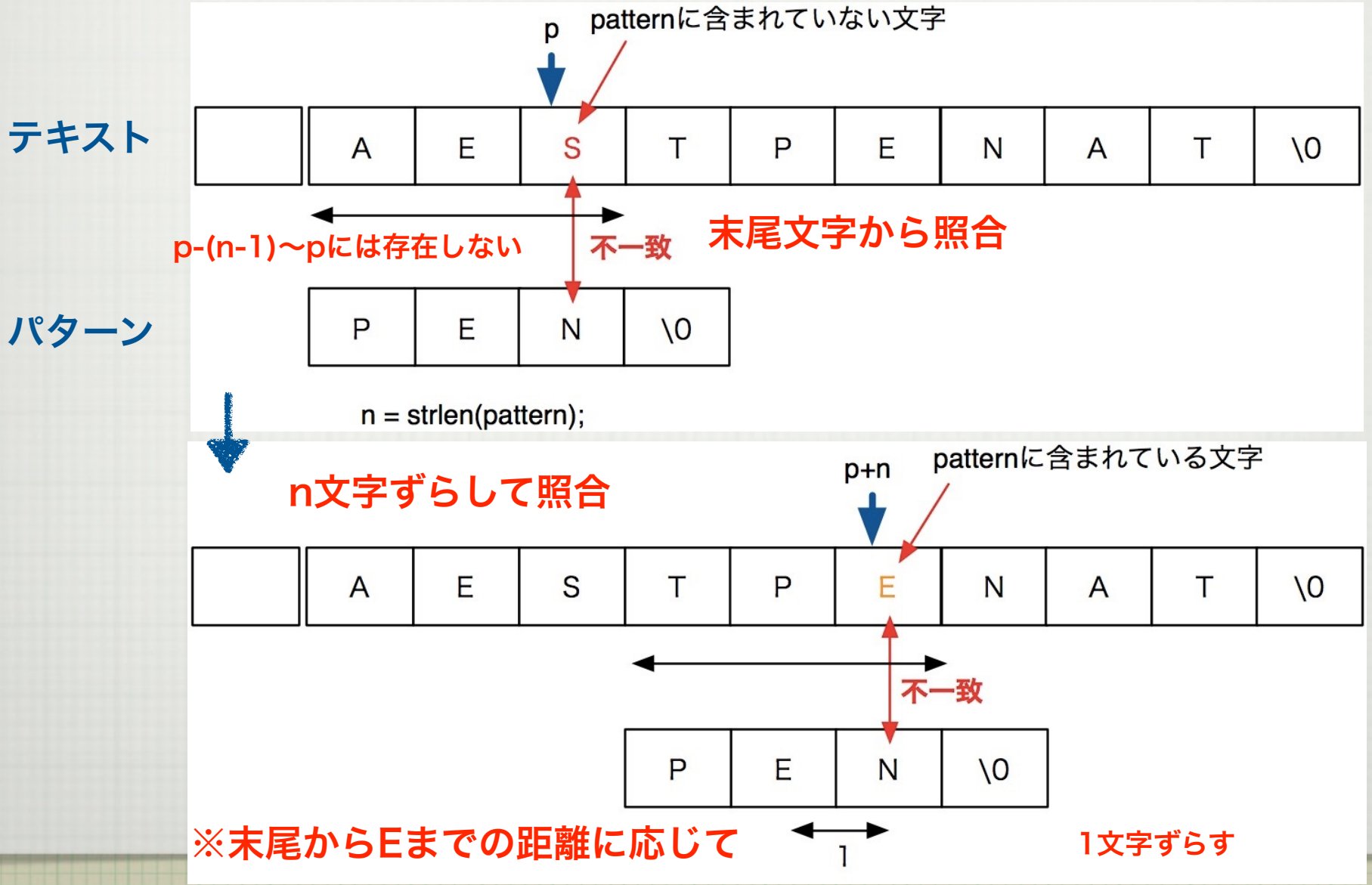
sample15-4.c
抜粋

```
int p_match(char *txt, char *pattern)
{
  int ptrtxt=0;
  int ptrpat=0;
  int num=0;


  while (txt[ptrtxt]!='\0'){
     while (pattern[ptrpat]!='\0' && txt[ptrtxt]==pattern[ptrpat]){
      ptrtxt++; ptrpat++;
   }
   if (pattern[ptrpat]=='\0')
      num++;
   ptrtxt=ptrtxt-ptrpat+1;
   ptrpat=0;
  }
  return num;
}
```

文字列探索：BM法

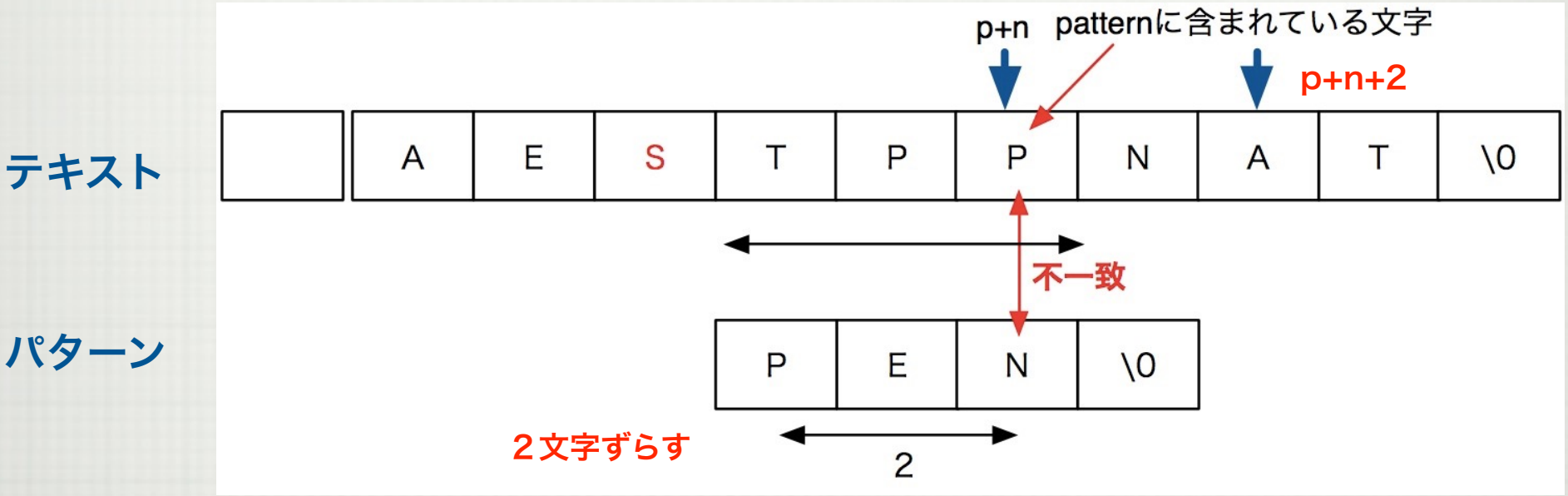
- **単純法では，**文字が一致しない場合1文字ずつずらしながら照合を繰り返す：**非効率**
- パターンの末尾文字から照合
- 照合しない場合，テキスト中の不一致文字に応じて，次の照合開始点を決める（ずらす量はあらかじめ用意する表によって決まる）

末尾文字で不一致の場合



末尾文字で不一致の場合 cont.

不一致の文字がパターンに含まれる場合



表の作成

どれだけずらすかを表す

skip[ ]

テキスト側の末尾文字	P	E	N	その他
ずらす数	2	1	3	3

strlen(pattern)

pattern中に同じ文字がある場合

テキスト側の末尾文字	A	R	R	A	Y	その他
ずらす数	1	2	2	1	5	5

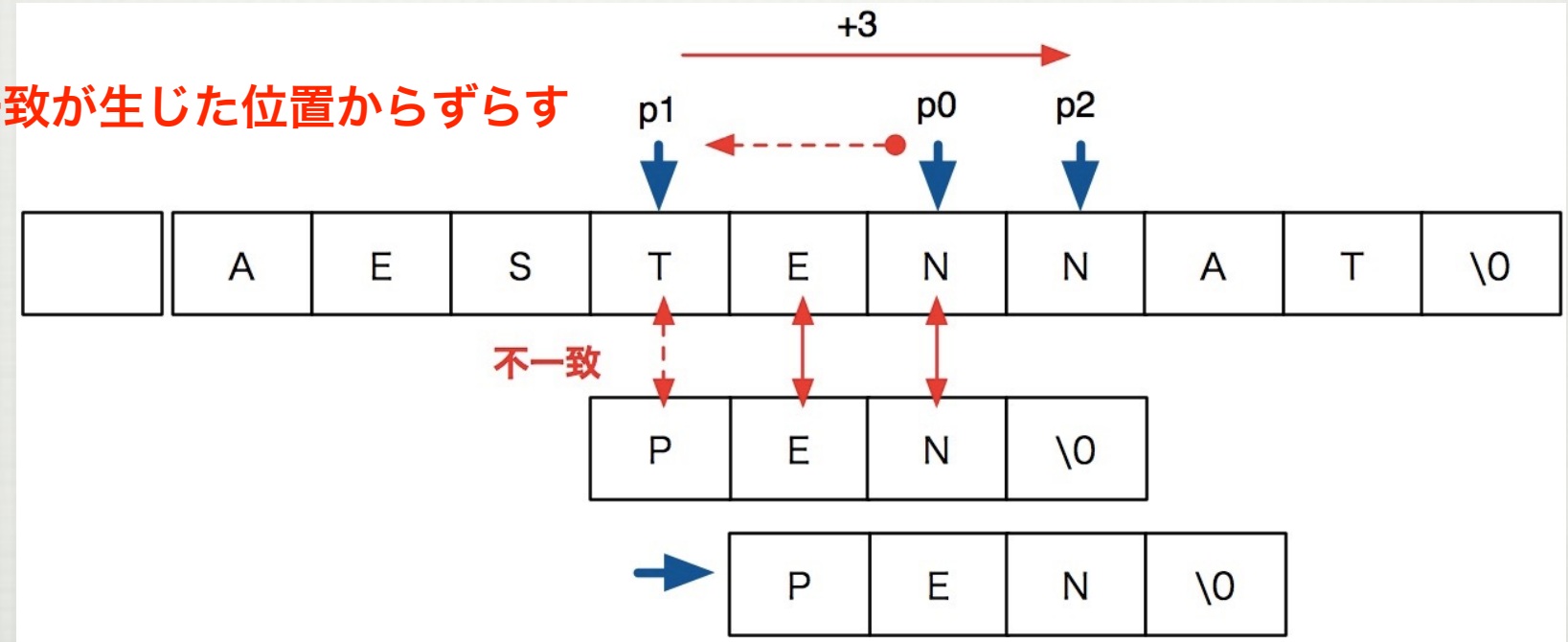
末尾側の値に合わせる

strlen(pattern)

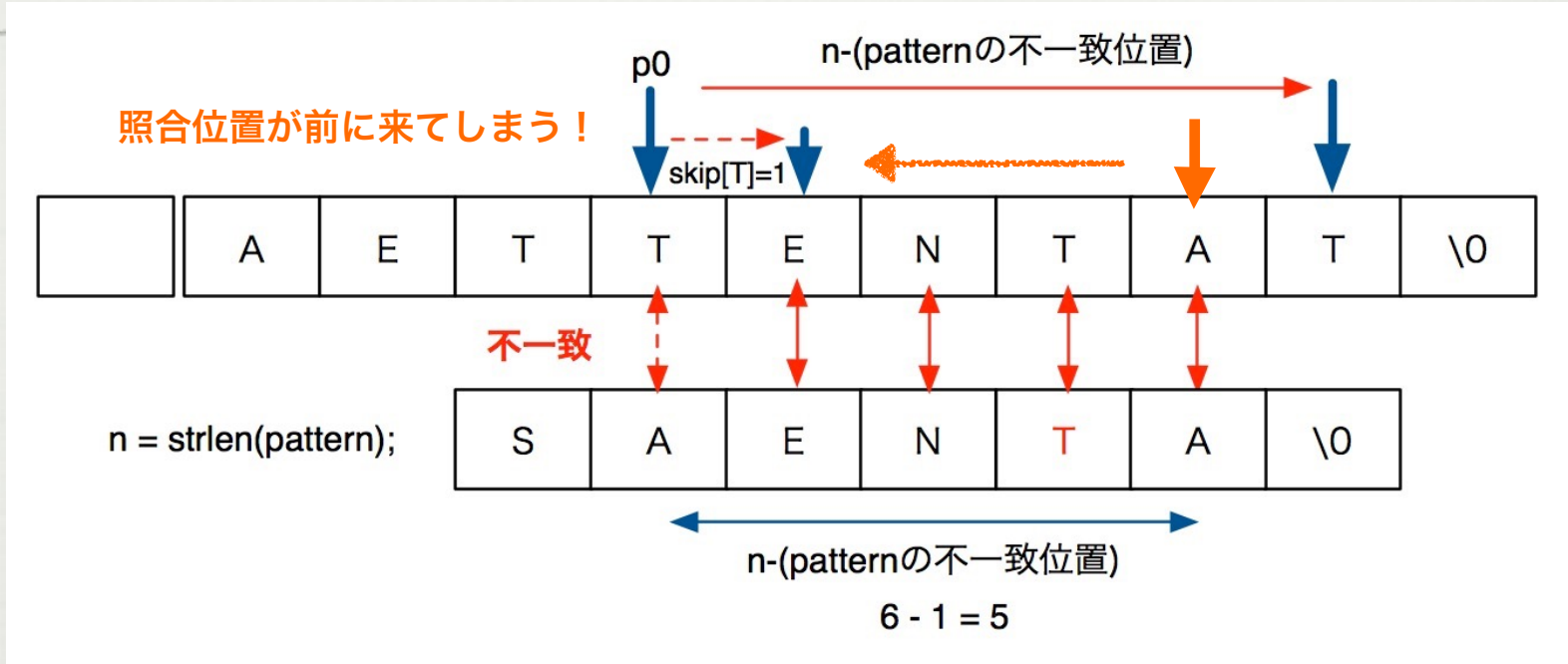
末尾以外で不一致が起こる場合

‘T’はpatternに含まれていないので３ずらす

不一致が生じた位置からずらす



末尾以外で不一致が起こる場合 cont.（注意！）



照合位置を一つ後方へ移動させるために，不一致位置から比較した回数（n-patternの不一致位置）だけ，ずらす．

※つまり，ずらす数(skip[T]) < (n-patternの不一致位置)の場合は，
(n-patternの不一致位置)をずらす回数とする
前回から右へ１つだけずらすことに相当

練習問題

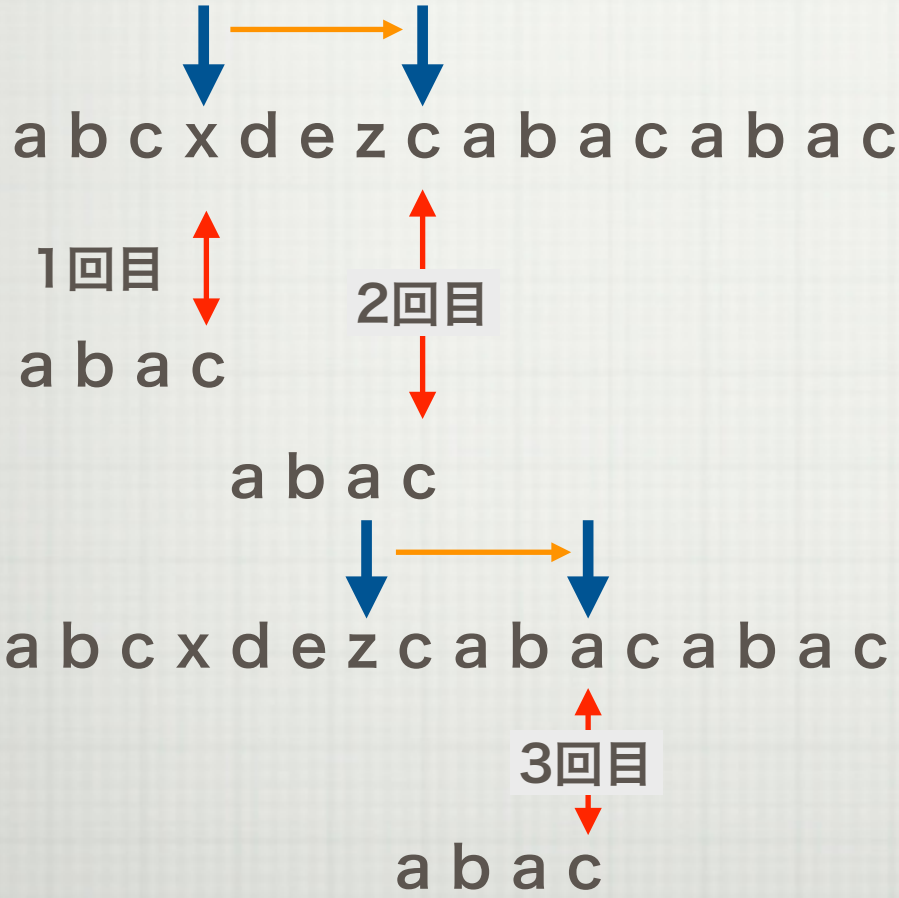
- 以下のテキストからパターン文字列（abac）をBM法で探索する様子をシミュレーションしなさい．まず，不一致の際にずらす数を示す表skipを作成し，何回目の走査でパターンを見つけることができるか考えなさい．

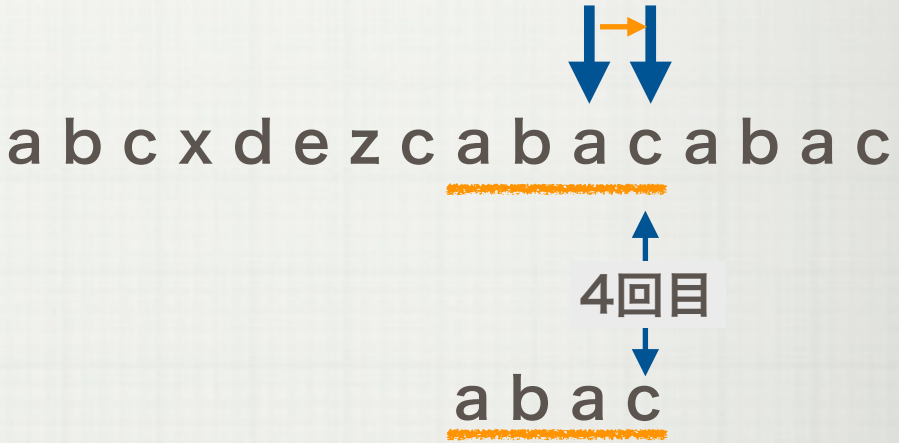
  テキスト：a b c x d e z c a b a c a b a c
  パターン：a b a c

解答

テキスト側の末尾文字	a	b	a	c	その他
ずらす数	1	2	1	4	4

strlen(pattern)





演習15-5

- アルファベットからなるテキストからパターン文字列をBM法で探索するプログラムを考える．まず，パターンから，パターンをずらすために用いる表を作成するプログラムを作りなさい（sample15-5.c）

sample15-5.c
抜粋

```
void table(char *key, int *skip)
{
  int n,k;

  n=strlen(key);
  for (k=0;k<255;k++) skip[k]=n;//パターン文字列の長さで初期化
  for (k=0; k<n-1; k++)
   skip[key[k]]=    n-1-k
                    //パターン文字列に含まれる文字ごとにずらす数を計算
}
```

演習15-6

- アルファベットからなるテキストの先頭からパターン文字列をBM法で探索し，最初にパターン文字列が出現する位置を出力するプログラムを作りなさい（sample15-6.c抜粋）．
  例：
  テキスト：abcxdezcabacabac
  パターン：abac
  出力：9文字目に見つかりました

sample15-6.c
抜粋

```c
int bm_match(char *txt, char *key)
{
  int ptrtxt=0;
  int ptrpat=0;
  int txtlen=strlen(txt);//テキストの長さ
  int keylen=strlen(key);//パターンの長さ
  int skip[256];//ずらす数を求める表
  char *p;
  int i;

  table(key, skip);//表の作成
  p=txtsearch(txt, key, skip);//文字列探索

  if (p!=NULL)//パターンが見つかった場合
   printf("%d文字目に見つかりました \n", (int)(p-txt+1));
  else printf("見つかりませんでした \n");

  return 1;
}
```

sample15-6.c
抜粋

```
char *txtsearch(char *txt, char *key, int *skip)
{
  int n,m,pkey;
  char *p;

  m=strlen(txt);//テキストの長さ
  n=strlen(key);//パターンの長さ
  p=txt+n-1;//照合位置の初期化
  while (p<txt+m) {
   pkey=n-1;//パターンの末尾から照合
   while (*p==key[pkey]) {
      if (pkey==0) {//一致した場合，一致した文字要素へのポインタを返す
         return p;
      }
      p--; pkey--;
   }
   p+=(skip[*p]>n-pkey)? skip[*p]:n-pkey;//一致しなかった場合，
                                          照合位置をずらす
  }
  return NULL;
}
```

課題15-2

- sample15-6.c抜粋を参考に，アルファベットからなるテキストについてパターン文字列をBM法で探索し，パターン文字列が出現する位置と出現回数を出力するプログラムを作りなさい（ex15-2.c）

レポート提出

- 課題15-1～15-2
- 提出期限：8月5日 0:00（8月4日 まで）
- 提出先：Webclass

- **試験：8月5日　４限**