C++プログラミングIII

第14回: 文字列探索

渡邉和宏

本日の講義内容

1. 文字列探索とは
2. 文字列探索アルゴリズム
3. 文字列探索の実装準備
4. 課題演習

1. 文字列探索とは

我々の生活に欠かせない文字列探索

Webページ内の検索:

https://1000logos.net/top-10-search-engines-logos-in-the-world/

コマンドラインでファイル内の文字列探索:

```
$ find . -name prac13_skel.cpp -type f -print0 | xargs -0 grep "heap"
./prac13_skel.cpp:void downheap(int a[], int left, int right);
./prac13_skel.cpp:void heap_sort(int a[], int n){
./prac13_skel.cpp:// downheap処理: a[left] から a[right] をヒープ化させる
./prac13_skel.cpp:void downheap(int a[], int left, int right){
./prac13_skel.cpp:    heap_sort(array, size);
```

文字列探索の概要 (1/2)

- 文字列探索: ある文字列中に別の文字列が含まれているかを探索すること.
- 文字列探索アルゴリズム: 部分文字列の位置を調べるアルゴリズム.
- 本講義ではchar型配列を用いた文字列の表現を考える.

  ＊NULL終端文字列を含むことに注意

```
char  str[7] = "string";
```

文字列終端を表すNULL文字

s	t	r	i	n	g	\0
str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]

文字列探索の概要 (2/2)

- 例: "string"に"ring"が含まれているか? 含まれているならば, 位置はどこか?

```
char  str[7] = "string";
```

s	t	r	i	n	g	\0
str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]

文字列探索アルゴリズムを用いて, テキスト内に現れるパターンの位置を調べるプログラムを考えたい.

【 使用単語の定義 】
探索する文字列: パターン (pattern) → "ring"
探索される側の文字列 : テキスト (text) → "string"

2. 文字列探索アルゴリズム

主な探索アルゴリズムの比較

＊探索テキストの文字数n, パターンの文字数m

- Brute-Force法
  - ✓ 単純なアルゴリズム.
  - ✕ 時間計算量が$O(nm)$.
  - △ただし作為的なパターンを処理しない限り実質は$O(n)$で実用的.
- Knuth-Moris-Pratt法
  - ✓ 時間計算量が最悪でも$O(n)$, 実際はそれより速い. 順ファイルを読み込みながらの探索に適している.
  - ✕ 処理が複雑にも関わらずBM法より遅い. パターン内に繰り返しがなければ効果がない.
- Boyer-Moore法
  - ✓ KMP法より理論的にも実践的にも優れている. 時間計算量が最悪$O(n)$,平均的には$O(n/m)$と高速.
  - ✕ 処理が複雑.

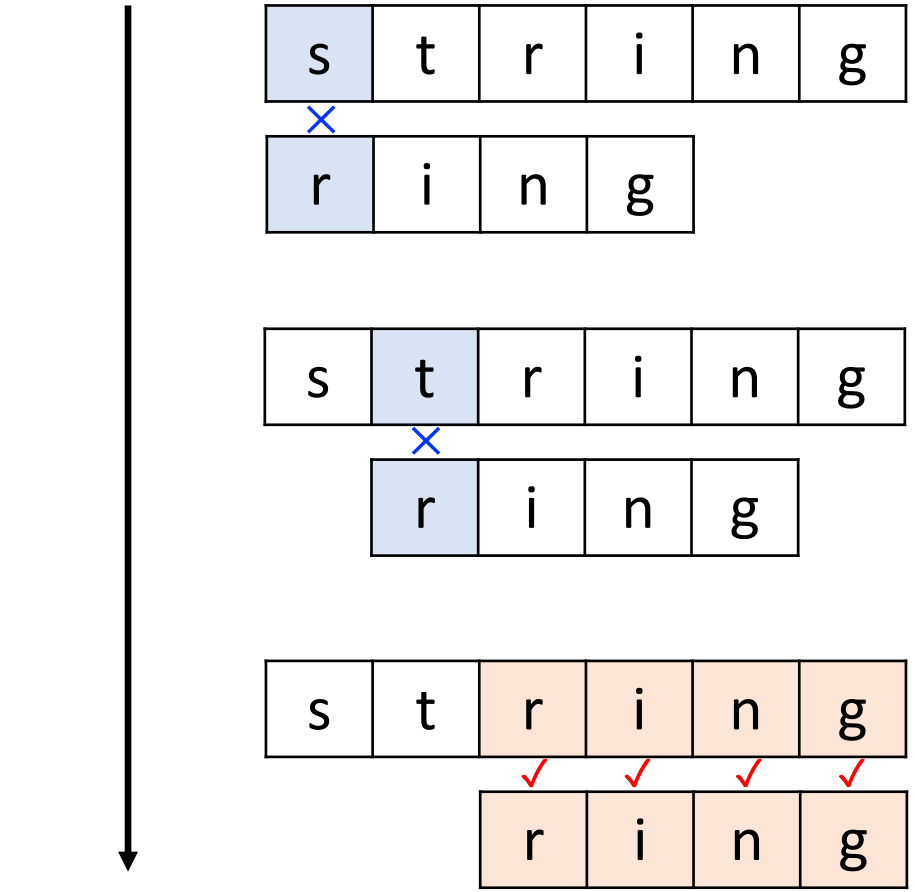
Brute-Force (BF)法 (1/3)

- 線形探索を拡張した簡単なアルゴリズムで, 単純法や力任せ法とも呼ばれる.
- 課題では下記の手順でBF法を実装する.

1) 先頭文字から順にテキストとパターンの照合を行う.
2) 文字が一致する間, テキストとパターンの添字を一つずつ増やして照合を続ける.
   i. パターン中の文字がすべて一致すれば, 探索成功.
   ii. 異なる文字に出会うと, それ以上の照合は不要のためステップ 3へ.
3) テキストの添字を1文字ずらし, パターンの添字を先頭に戻した後, ステップ1 -- 2を繰り返す. テキストの文字をすべて走査したら終了 (その場合は探索失敗).

Brute-Force (BF)法 (2/3)

例1: 探索テキスト "string", 探索パターン "ring"



探索テキストとパターンの先頭文字を重ねて,
先頭文字から比較 → 1文字目で不一致

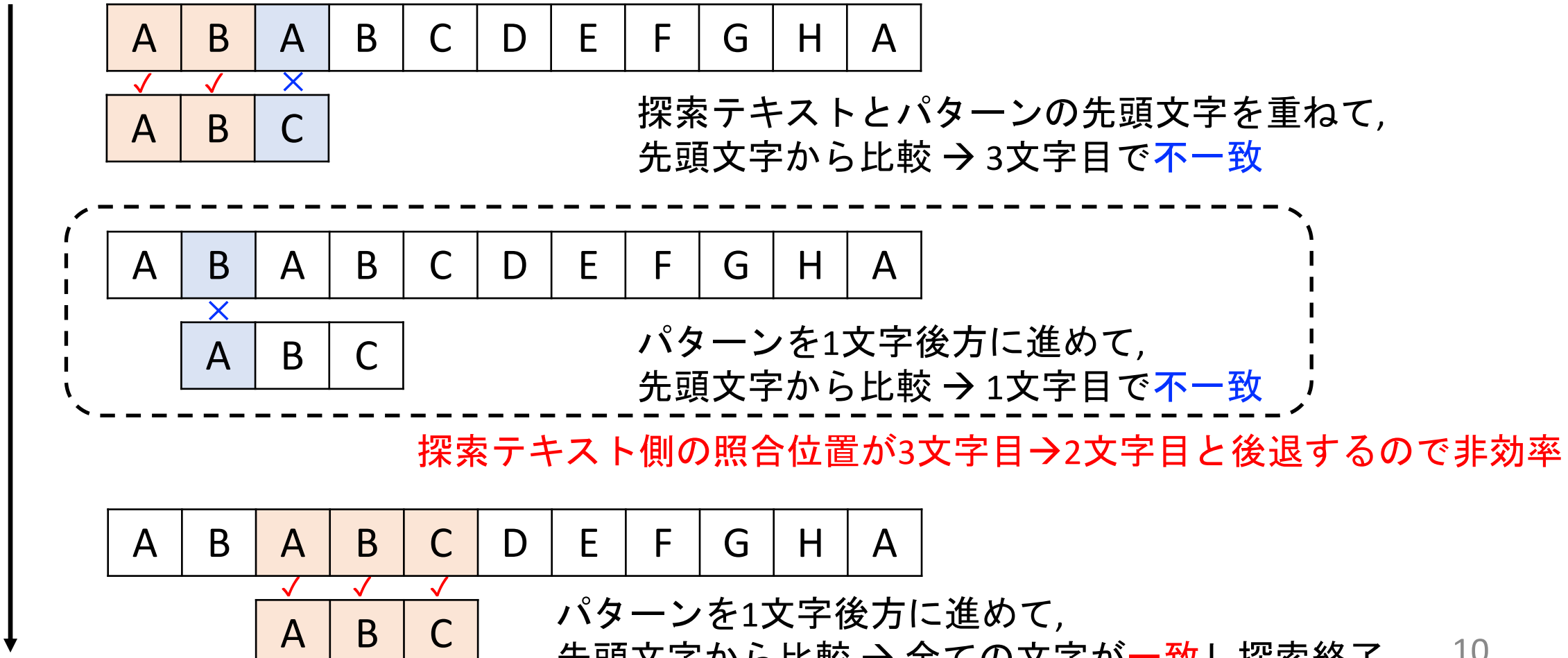
パターンを1文字後方に進めて,
先頭文字から比較 → 1文字目で不一致

パターンを1文字後方に進めて,
先頭文字から比較 → 全ての文字が一致

"ring" の探索に成功し, 探索処理の終了.

Brute-Force (BF)法 (3/3)

例2: 探索テキスト "ABABCDEFGHA", 探索パターン "ABC"

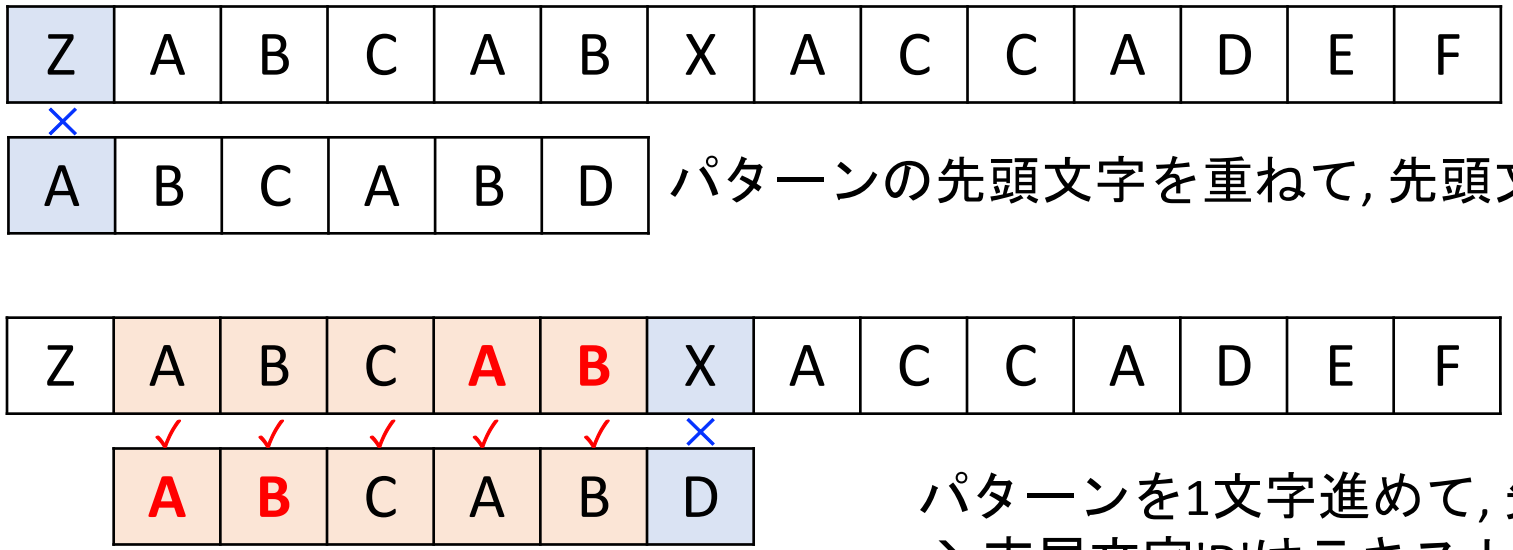


Knuth-Moris-Pratt (KMP)法 (概要のみ) (1/3)

- D.E.Knuth, V.R.Pratt, J.H.Morisによる考案.
- BF法では, 不一致文字と出会った段階で, それまでの照合結果を捨て去って, 探索対象文字を１つ進めて, パターンの先頭文字からの照合を行い直すため非効率. (前スライド参照)
- 一方, KMP法は照合結果を捨てることなく活用する.
- BF法と異なり, パターンの照合位置が前進するだけという利点がある.
- 注意) 課題ではKMP法は実装しない.

Knuth-Moris-Pratt (KMP)法 (概要のみ) (2/3)

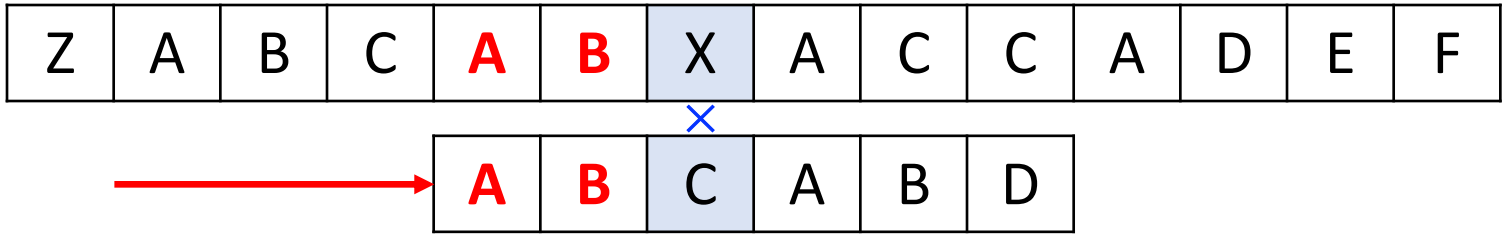
例: 探索テキスト "ZABCABXACCADEF"
探索パターン "ABCABD"



ここで, テキストとパターンの "**AB**" が照合済みとみなせれば, テキストの'X'以降の文字列が, パターンの "CABD" と一致するかどうかを調べれば良く, BF法のように照合位置を前方に戻す必要がない.

Knuth-Moris-Pratt (KMP)法 (概要のみ) (3/3)

例: 探索テキスト "ZABCABXACCADEF"
　　探索パターン "ABCABD"



テキストの "AB" と重なるように, パターンを大きく3文字進めて, パターンの3文字目'C' 先頭から順に比較 → 'X'-'C': 不一致

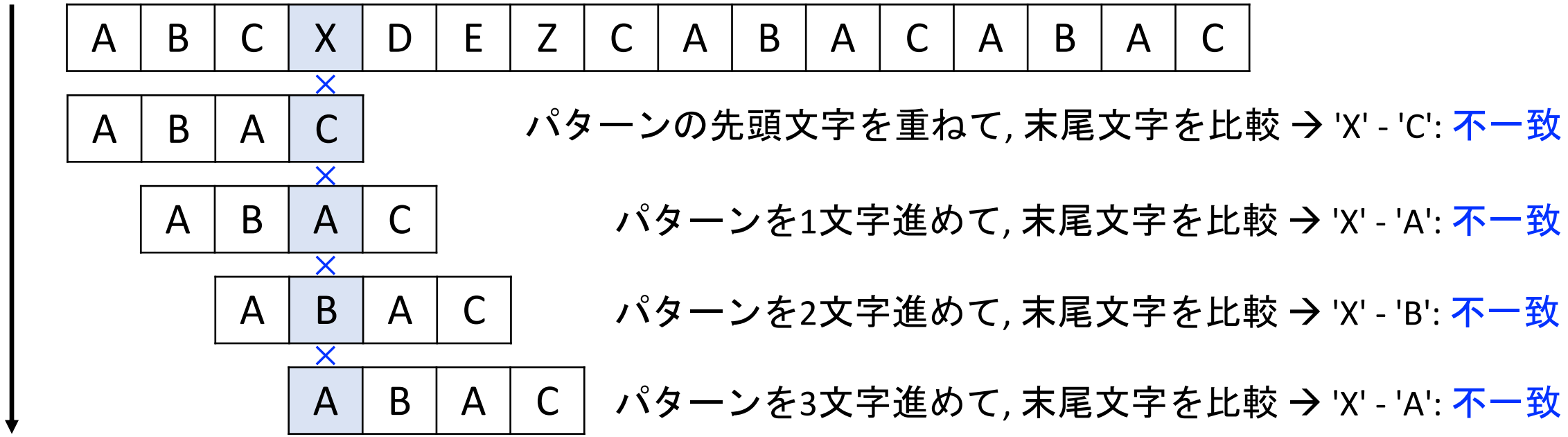
- KMP法はテキストとパターン中の重なっている部分を見つけて, 照合の再開位置を計算し, パターンを大きく移動させるアルゴリズム.
- 効率良く処理するために, KMP法では照合再開値を事前に計算し, 表(スキップテーブル)を準備しておく.

Boyer-Moore (BM)法 (1/5)

- R.S. Boyer, J.S.Mooreによる考案.
- KMP法同様に, 照合結果を捨てることなく活用する.
- KMP法よりも探索速度に優れるため, 実際の文字列探索で広く利用されている.
- BM法では, パターンの末尾文字から先頭側へと照合を行い, その過程で不一致文字を見つけた場合に, 事前に用意した表(スキップテーブル)に基づいてパターンの移動量を決定.
- 課題ではBM法を実装する. 手順は後述.

Boyer-Moore (BM)法 (2/5)

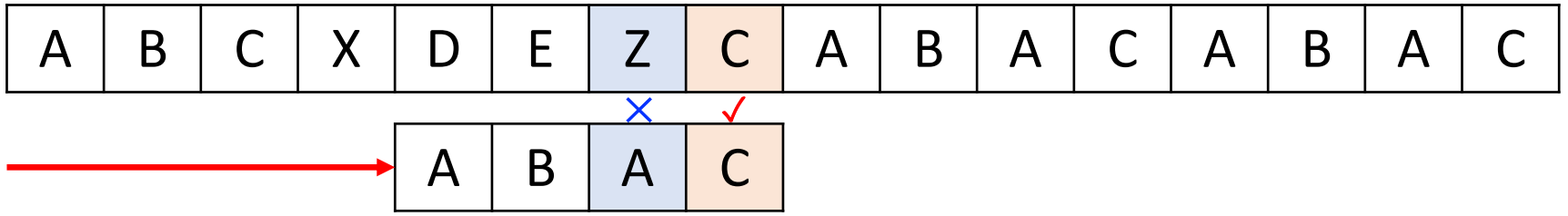
例: 探索テキスト "ABCXDEZCABACABAC"
　　探索パターン "ABAC"



パターンに含まれない文字を探索テキスト中に発見した場合, テキスト中のそこまでの文字("ABCX")を探索対象から除外できる.

Boyer-Moore (BM)法 (3/5)

例: 探索テキスト "ABCXDEZCABACABAC"
　　探索パターン "ABAC"



パターンを一気に不一致した文字まで飛ばす

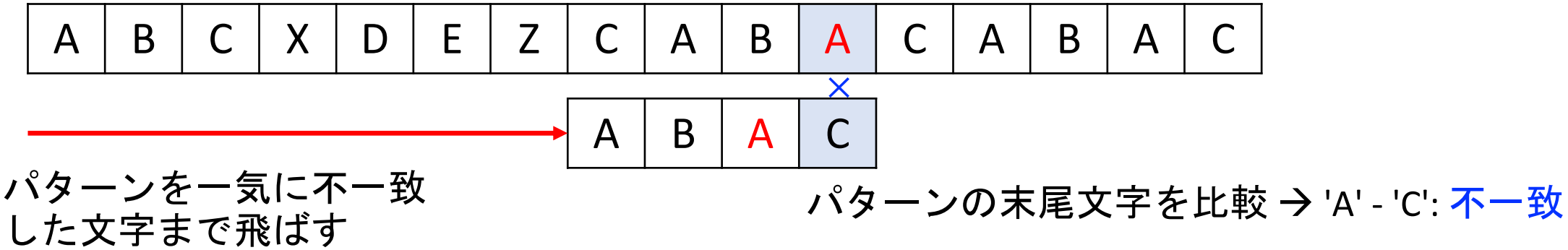
パターンの末尾文字を比較 → 'C' - 'C': 一致

照合位置を一文字先頭側に進めて比較 → 'Z' - 'A': 不一致

パターンに含まれない文字を探索テキスト中に発見したため, そこまでの文字('Z'まで)は探索対象から除外する.

Boyer-Moore (BM)法 (4/5)

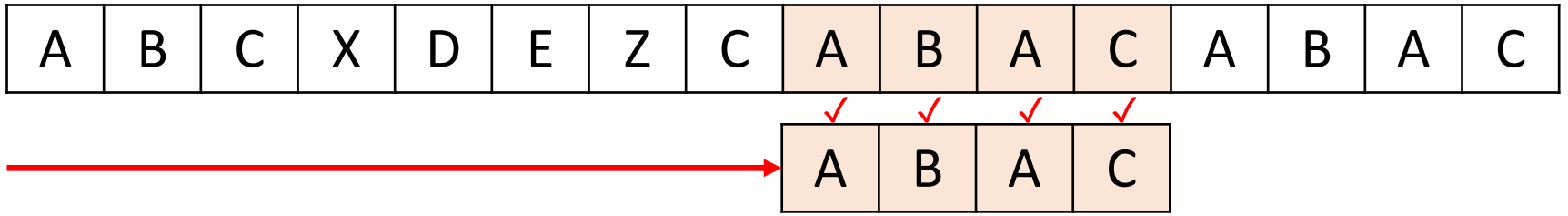
例: 探索テキスト "ABCXDEZCABACABAC"
　　探索パターン "ABAC"



- 探索テキスト内の 'A' とパターン内の 'C' は一致していないが, 'A' はパターン "ABAC" に含まれている.
- そこで, 探索テキスト内の 'A' とパターン内の後側の 'A' (3文字目の 'A') が重なるようにパターンを移動させる.
- 注意) 前側の'A'に重ねると上手くいかない.

Boyer-Moore (BM)法 (5/5)

例: 探索テキスト "ABCXDEZCABACABAC"
　　探索パターン "ABAC"



パターンを一気に不一致した文字まで飛ばす

パターンの末尾文字を比較 → 'C' - 'C': 一致

照合位置を一文字先頭側に進めて比較 → 'A' - 'A': 一致

照合位置を一文字先頭側に進めて比較 → 'B' - 'B': 一致

照合位置を一文字先頭側に進めて比較 → 'A' - 'A': 一致

"ABAC"の探索に成功し，探索処理の終了．

3. 文字列探索の実装準備

BM法でのスキップテーブルの作成

- BM法を実装するために, 各文字に出会ったときの移動量 (照合中の文字を何文字ずらして照合を再開するか) を格納した表 (スキップテーブル) を事前に作成すべき.
- スキップテーブルを用意していない場合, パターンを移動させるたびに照合再開値を計算し直すので, 非効率である.
- 探索パターンが n 文字であるときの移動量は以下の通り.
  - パターンに含まれない照合文字: 移動量＝ n
  - パターンに含まれる照合文字:　移動量＝ n－k－1
    (k ＝ 探索パターンにおいて最後に出現する位置の添え字)
  - 文字が重複する場合は右端の文字が優先.

BM法でのスキップテーブルの例

例: 探索テキスト → "ABCXDEZCABACABAC"
　　探索パターン → "ABAC"

A	B	C	D	E	F	G	H	I
1	2	0	4	4	4	4	4	4

J	K	L	M	N	O	P	Q	R
4	4	4	4	4	4	4	4	4

S	T	U	V	W	X	Y	Z
4	4	4	4	4	4	4	4

- Aはパターンの3文字目(添字2):
  移動量 = 4 – 2 – 1 = 1
- Bはパターンの2文字目(添字1):
  移動量 = 4 – 1 – 1 = 2
- Cはパターンの4文字目(添字3):
  移動量 = 4 – 3 – 1 = 0
- Dはパターンに含まれていない:
  移動量 = 4
- ….
- 表に載っていない文字 (数字や小文字など)については, 移動量は全て4.

Boyer-Moore (BM)法の実装手順

1) スキップテーブルをパターンのサイズで初期化した後, パターンに対してテーブルの値を計算.
2) テキストの照合開始位置の添字をパターンの末尾の位置に設定し探索開始.
3) 文字が一致する間, テキストとパターンの添字を一つずつ先頭側にずらして照合を続ける.
    i. パターン中の文字がすべて一致すれば, 探索成功.
    ii. 異なる文字に出会うと, スキップテーブルの値に応じて探索テキストの照合開始位置をずらす.
    iii. パターン添字を末尾に戻し, ステップ 4へ.
4) 探索テキストの照合開始位置がテキストの末尾に達していない限り, ステップ3の i--iiiを繰り返す. テキストの文字をすべて走査したら終了 (その場合は探索失敗).

課題演習

課題内容 (課題ファイル参照)

- CoursePowerで配布されるスケルトンファイルを完成させ, 文字列探索を行うプログラムを作成する.
- 本課題ではBF法とBM法の両方を実装し, 探索効率の違いを確認する. KMP法はBM法と比べると実用的ではないため, 今回は実装しない.
- 与えられた入力ファイルを読み込み, パターンに合う文字列が含まれているか探索する. パターンの探索が成功した場合は, 探索に要した照合回数を出力する. 入力ファイルはCoursePowerで配布.

input14.txt (探索対象テキスト)

It should have been so simple. It was relatively easy to get the project done, and it was simple enough that even a child should have been able to complete it on time, but that wasn't the case. The deadline had arrived, and the project remained unfinished.

課題内容 (課題ファイル参照)

❑ コマンドラインでの実行例1
$ ./a.out (もしくは./a.exe)
Search pattern: exam
Search algorithm: f

With the BF algorithm:
The search pattern, exam, is not found.

❑ コマンドラインでの実行例2
$ ./a.out (もしくは./a.exe)
Search pattern: deadline
Search algorithm: f

With the BF algorithm:
The search pattern, deadline, is found.
The word lies at 198.
The number of times searched: 199

❑ コマンドラインでの実行例3
$ ./a.out (もしくは./a.exe)
Search pattern: deadline
Search algorithm: b

With the BM algorithm:
The search pattern, deadline, is found.
The word lies at 198.
The number of times searched: 32

- パターンと使用アルゴリズムを入力(下線)
- fはBF法の指定, bはBM法の指定
- テキスト中の空白も1文字としてカウント
- 大文字と小文字は区別する

講義内容のまとめ

- 文字列探索の主要アルゴリズムの一部であるBF法, KMP法, BM法を紹介した.
- BF法は単純だが, パターン探索過程で照合位置が前後するため非効率である.
- 一方, BM法やKMP法は, スキップテーブルの値に応じて, パターン探索過程で照合位置を大きく動かすという特徴を持つ.

質問・コメントを歓迎します.
対面での対応は, 12号館1階2104室.
メールもしくはteamsのチャットでも対応します.