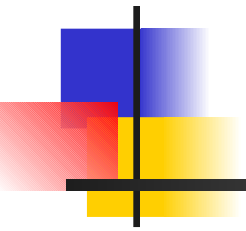


アルゴリズムとデータ構造III
10回目：12月17日（月）（補講）

全文検索アルゴリズム
（Simple Search, KMP）

授業資料　http://ir.cs.yamanashi.ac.jp/~ysuzuki/algorithm3/index.html

授業評価アンケートに関して

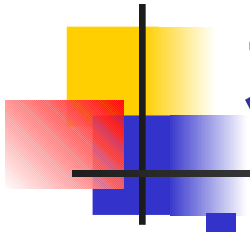
- 教科書があった方がよい
  - 授業の範囲を網羅している教科書が見つからなかったので教科書は指定しませんでした.
    来年はなるべく教科書を指定できるように今から教科書を探しておきます.

授業の予定（中間試験まで）

1	10/11	スタック（後置記法で書かれた式の計算）
2	10/18	文脈自由文法
3	10/25	構文解析　CKY法
4	11/01	構文解析　CKY法，チャート法
5	11/08	構文解析　CKY法，チャート法
6	11/15	構文解析　チャート法
7	11/29	グラフ（動的計画法，ダイクストラ法，DPマッチング）
8	12/06	グラフ（DPマッチング，ビームサーチ，A＊アルゴリズム）
9	12/13	中間試験

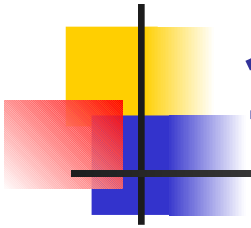
授業の予定(中間試験以降)

10	12/17	全文検索アルゴリズム(simple search, KMP)
11	12/20	全文検索アルゴリズム(BM, Aho-Corasick)
12	01/10	テキスト圧縮　暗号(例:モールス信号，黄金虫，踊る人形，ハフマン符号，Zipfの法
13	01/17	音声圧縮，zip，ADPCM，MP3
14	01/24?	音声圧縮(CELP)，画像圧縮(JPEG)
15	02/07	期末試験



本日のメニュー

- 全文検索アルゴリズム
  - 全文検索とは
  - simple search
    - 動作の説明
    - アルゴリズム
  - KMP
    - 動作の説明
    - アルゴリズム



全文検索

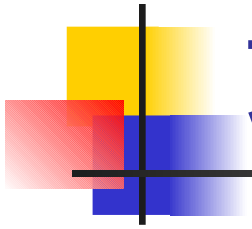
- 文書中から，与えられた文字列と完全に一致する部分を探し出す．

- 全文検索の種類
  - **文字列照合による全文検索**
  - 索引を用いた全文検索

文字列照合タスク

- テキスト処理には不可欠
- テキスト文字列からキーワードとその出現位置を見つける
- 例
  - テキスト文字列：aabcdabdabbabcdabacade
  - キーワード：abcaba

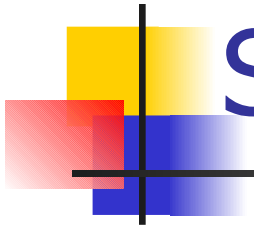
a	b	c	a	b	c	a	b	a	b	c	a	b	a	b	x	a	b	c	a
			a	b	c	a	b	a											
								a	b	c	a	b	a						



文字列照合アルゴリズム

- Simple Search
- Knuth-Morris-Pratt法
- Boyer-Moore法
- Aho-Corasick法

文字列照合問題の単純な解決法 Simple Search



- Simple Searchの文字列照合手順
- Simple Searchのアルゴリズム
- Simple Searchの評価

単純な文字列照合アルゴリズム Simple Search

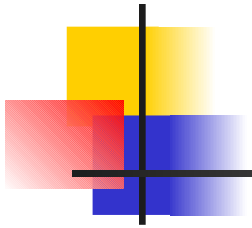
- テキストストリングの1文字目からn文字目まで，2文字目からn+1文字目まで，・・・がキーワードと一致するかどうかをチェックする.

a	b	c	a	b	c	a	b	a	b	c	a	b	a	b	x	a	b	c	a
a	b	c	a	b	a														
	a	b	c	a	b	a													
		a	b	c	a	b	a												
			a	b	c	a	b	a											
				a	b	c	a	b	a										

Simple Search 同じ部分を何度も照合しなければならない

位置	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
text	a	b	c	a	b	c	a	b	a	b	c	a	b	a	b	x	a	b	c	a	b	x
	a	b	c	a	b	a																
		a																				
			a																			
				a	b	c	a	b	a													
					a																	
						a												照合失敗				
							a	b	c													
								a										文字列照合成功				
									a	b	c	a	b	a								
										a												
											a											
												a	b	c								
照合回数	1	2	2	2	3	3	2	3	3	2	2	2	2	2								

Simple Searchのアルゴリズム



- 入力：テキストストリング　text, キーワード key
- 出力：テキストストリング中のキーワードの位置
- m: テキストストリングの長さ
- n: キーワードの長さ

```
Method
begin
        for i:=1 to m-n+1 do                    起点を決めて
        begin
                for j:=1 to n do                キーワードと1字ずつ照合
                        if text[i+j-1]≠key[j] then
                                goto 1;
                print i;
                1:
        end
end
```

Simple Search 最も効率の悪い場合

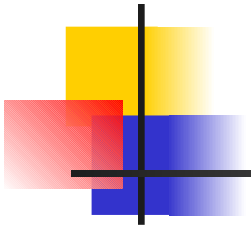
文字照合回数 (7-3+1)*3=15

(m-n+1)*n回

一般にm≫nなので O(mn)

- key = aaa
- text = aaaaaaa

位置	1	2	3	4	5	6	7
text	a	a	a	a	a	a	a
	a	a	a				
		a	a	a			
			a	a	a		
				a	a	a	
					a	a	a
照合回数	1	2	3	3	3	2	1

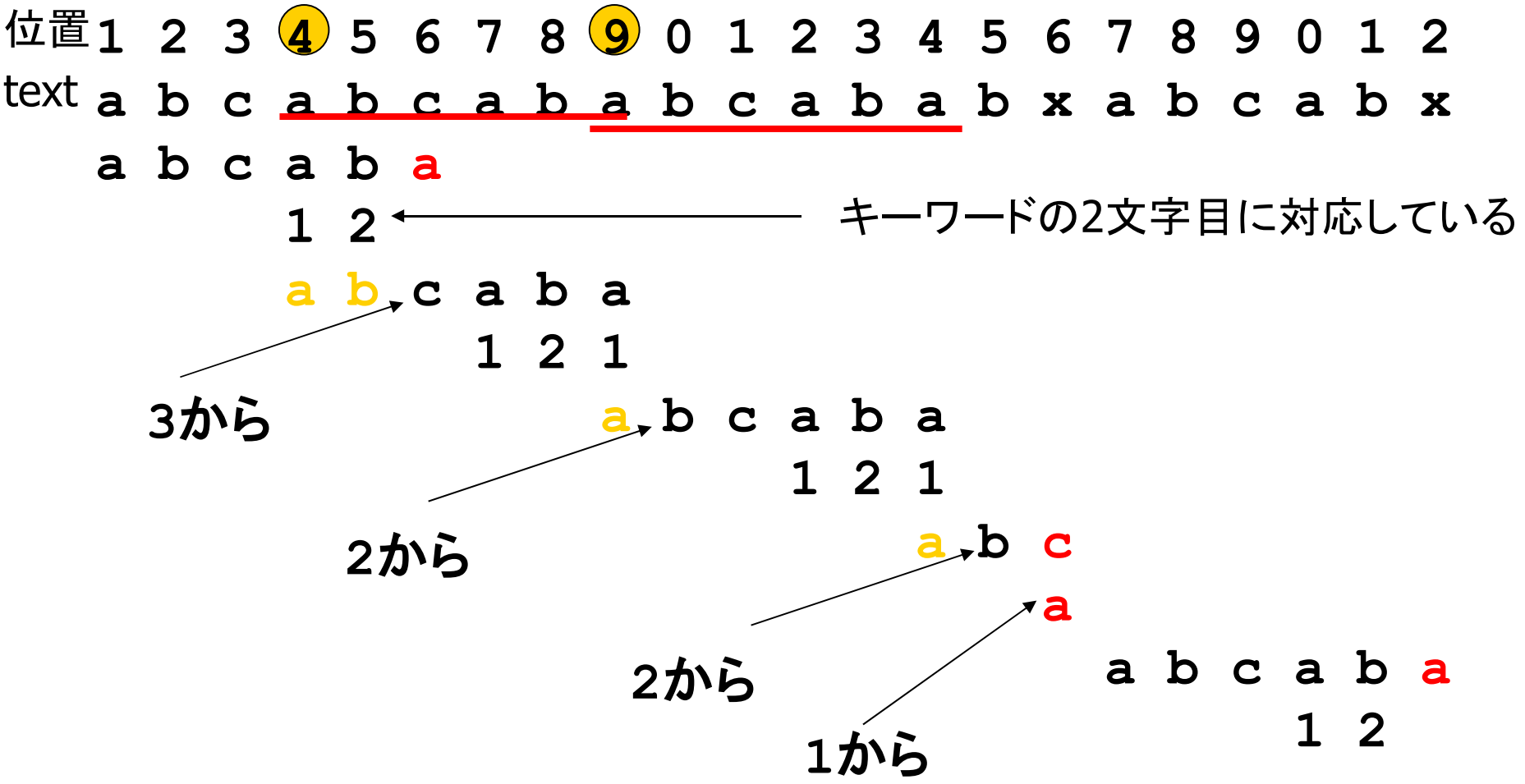


Knuth-Morris-Pratt法 （KMP法）

- Simple Search
  - テキストストリング中の各文字がキーワードと複数回照合される　→　冗長
- KMP法
  - 文字照合の実行中に次回の文字照合を考慮しつつ処理を進める
  - 文字照合中，バックトラックが必要ない

Knuth-Morris-Pratt法

```
Key:  a b c a b a
      1 2 3 4 5 6
next  0 1 1 0 1 3 2
```



KMP法　アルゴリズム

m :textの長さ
n :keywordの長さ
i: textの照合位置
J: keywordの照合位置

```
Method kmp
begin
      j:=1;
      for i:=1 to m do
      begin
            while j>0 and key[j] ≠text[i] do
                  j:=next(j);
            if j=n then
                  print i-n+1:
            j:=j+1;
      end
end
```

照合

つぎの照合位置

照合成功

キーワードの接頭辞文字列の出現位置

位置	1	2	3	4	5	6	7				
キーワード	a	b	c	a a	b b	a c a	a b	b c	a a	b	a
next関数値	0	1	1	0	1	3	2				

関数next：次回の照合でキーワードの何文字目を照合すべきか

テキストストリング中の照合に失敗した文字の直前の何文字がキーワードの接頭辞になっているか

next関数

Keyword: abcabaのとき
123456

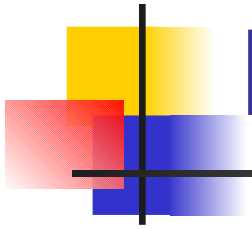
a:1：keywordの一文字目のa
a：a以外の文字

1文字目のaで照合失敗 （直前の文字がa）
→ 照合失敗箇所の右隣とa:1を照合
→ 照合失敗箇所はキーワードの0文字目と照合→ next(1)=0
2文字目のbで照合失敗 （直前の文字がab）
→ 照合失敗箇所とa:1を照合 → next(2)=1
3文字目のcで照合失敗 （直前の文字がabc）
→ 照合失敗箇所とa:1を照合 → next(3)=1
4文字目のaで照合失敗 （直前の文字がabca）
→ 照合失敗箇所の右隣とa:1を照合
→ 照合失敗箇所はキーワードの0文字目と照合→ next(4)=0
5文字目のbで照合失敗 （直前の文字がabcab）
→ 照合失敗箇所とa:1を照合 → next(5)=1
6文字目のaで照合失敗 （直前の文字がabcaba）
→ 照合失敗箇所とc:3を照合 → next(6)=3
6文字目のaで照合成功 （直前の文字がabcaba）
→ 照合失敗箇所（照合成功末尾の右隣）とb:2を照合 → next(7)=2

KMP法　アルゴリズム next関数
入力：キーワード key, 出力：next関数

n：keyの長さ
J：keyの照合位置
t：keyのj文字目の直前の何文字がkeyの接頭語になっているか

```
Method next
begin
        t:=0;
        next(1):=0;
        for j:=1 to n do
        begin
                while t ≠ 0 and key[j] ≠ key[t] do
                        t:=next(t);
                t:=t+1;
                if key[j+1]=key[t] then
                        next(j+1):=next(t);
                else
                        next(j+1):=t;
        end
end
```



KMP法の評価

- KMP法
  - 漸近的時間計算量　$O(m)$
  - next関数が必要
- Simple Search法
  - 漸近的時間計算量　$O(mn)$