



# アルゴリズムとデータ構造III

## 9回目: 12月4日(金) 4時限

---

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(中間期評価)



---

- CNSの授業のコミュニティに以下の項目について記入してください(匿名での記入が可能).
1. この授業の良いところはどこですか？
  2. この授業の改善してほしいところはどこですか？

# 授業の予定(中間試験まで)

1	10/01	スタック(後置記法で書かれた式の計算)
2	10/15	文脈自由文法, 構文解析, CYK法
3	10/22	構文解析 CYK法
4	10/29	構文解析 CYK法
5	11/12	構文解析 CYK法, 動的計画法
6	11/19	構文解析(チャート法), グラフ(ダイクストラ法)
7	11/26	グラフ(ダイクストラ法, DPマッチング, A*アルゴリズム)
8	12/03	グラフ(A*アルゴリズム), 前半のまとめ
9	12/04 4時限	<b>教室:A1-41</b> 全文検索アルゴリズム (simple search, KMP)

# 授業の予定(中間試験以降)

10	12/10	中間試験(8回目までの範囲)
11	12/11 4時限	教室:A1-41 全文検索アルゴリズム(BM, Aho-Corasick)
12	12/17	全文検索アルゴリズム(Aho-Corasick), データ圧縮
13	01/07	暗号(黄金虫, 踊る人形) 符号化(モールス信号, Zipfの法則, ハフマン符号)テキスト圧縮
14	01/14	テキスト圧縮(zip), 音声圧縮(ADPCM, MP3, CELP), 画像圧縮(JPEG)
15	01/21	期末試験

# 中間試験

- 中間試験日
  - 12月10日(木)
- 範囲
  - スタック
  - 文脈自由文法
  - 構文解析
  - CYK法
  - (トップダウンチャート法)
  - 動的計画法
  - ダイクストラ法
  - DPマッチング
  - A\*アルゴリズム



# 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

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
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						

答え

キーワードは含まれているか: YES

出現位置: 4文字目から始まる文字列と9文字目から始まる文字列





# 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

- テキスト文字列の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. (n:キーワードの文字数)

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
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					1文字目からの照合→6回目の照合で失敗									
	a	b	c	a	b	a				2文字目からの照合→1回目の照合で失敗									
		a	b	c	a	b	a			3文字目からの照合→1回目の照合で失敗									
			a	b	c	a	b	a		4文字目からの照合→照合成功！！									
				a	b	c	a	b	a	5文字目からの照合→1回目の照合で失敗									

 は照合失敗箇所

 は文字列照合に成功

# Simple Search

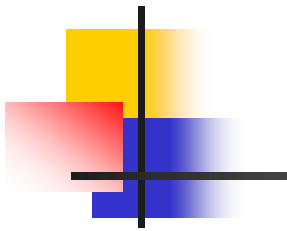
同じ部分を何度も照合しなければならない

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																
							a	b	c													
								a														
									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								

照合失敗

文字列照合成功

# Simple Searchのアルゴリズム

- 
- 入力: テキスト文字列 text, キーワード key
  - 出力: テキスト文字列中のキーワードの位置
  - m: テキスト文字列の長さ
  - 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

if text[i+j-1] ≠ key[j] then キーワードと1字ずつ照合

goto 1; 照合に失敗したらループを抜ける

print i;

1:

end

end

# Simple Search 最も効率の悪い場合

■ key = aaa

文字照合回数  $(7-3+1)*3=15$

■ text = aaaaaaa

$(m-n+1)*n$ 回  
一般に  $m \gg n$  なので  $O(mn)$

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

- テキスト文字列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, **バックトラックの必要がない**

# Knuth-Morris-Pratt法

Key: a b c a b a

1 2 3 4 5 6

次にキーワードの何文字目から照合すればよいか → next 0 1 1 0 1 3 2

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 (keyの6文字目で照合失敗)

1 2 ← キーワードの2文字目に対応している

a b c a b a (照合成功)

1 2 1

Keyの3文字目から

a b c a b a (照合成功)

1 2 1

Keyの2文字目から

a b c (keyの3文字目で照合失敗)

Keyの2文字目から

a b c a b a

Keyの1文字目から

1 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 照合

                j:=next(j);

            if j=n then

                次の照合位置

                print i-n+1:

                j:=j+1;

                照合成功

        end

end

# キーワードの接頭辞文字列の出現位置

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

テキスト文字列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を調べる

位置	1	2	3	4	5	6	7				
キーワード	a	b	c	a	b	a					
				a	b	c	a	b	a		

6文字目で照合失敗した場合: 直前文字列がabなので3文字目から照合開始

キーワード	a	b	c	a	b	a	b	c	a	b	a
						a	b	c	a	b	a

照合に成功した場合: 直前文字がaなので2文字目から照合開始

next関数値	0	1	1	0	1	3	2				
---------	---	---	---	---	---	---	---	--	--	--	--

# next関数

Keyword: abcabaのとき    a:1 : keywordの一文字目のa  
123456                            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

# next関数

Keyword: abcabaのとき a:1 : keywordの一文字目のa

123456

a : a以外の文字

4文字目のaで照合失敗（直前の文字がabca）

→ 照合失敗箇所（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abcabb）

→ 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

Method next

begin

t:=0;

next(1):=0;

for j:=1 to n do      keyの各文字に対してnext関数値を計算

begin

while t ≠ 0 and key[j] ≠ key[t] do

t:=next(t);      keyのj文字目までの文字列がkeyの

t:=t+1;

接頭辞と一致しているか調べる

if key[j+1]=key[t] then

next(j+1):=next(t);

keyの

j+1文字目の

else

next関数値を

next(j+1):=t;

決定

end

end

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の接頭辞になっているか



# KMP法の評価

---

## ■ KMP法

- 漸近的時間計算量  $O(m)$

- next関数が必要

テキスト文字列の各文字に対して1回照合

## ■ Simple Search法

- 漸近的時間計算量  $O(mn)$

テキスト文字列の各文字に対して  
キーワード文字数回照合

m: テキストの文字数

n: キーワードの文字数