

## アルゴリズムとデータ構造III 12回目:1月5日(木)

全文検索アルゴリズム  
(Aho-Corasick)  
暗号: 符号化: テキスト圧縮

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

## 授業の予定(中間試験まで)

1	10/06	スタック(後置記法で書かれた式の計算)
2	10/13	チューリング機械, 文脈自由文法
3	10/20	構文解析 CYK法
4	11/10	構文解析 CYK法
5	11/17	構文解析(チャート法), グラフ(ダイクストラ法)
6	12/01	構文解析(チャート法), グラフ(ダイクストラ法, DPマッチング)
7	12/08	グラフ(DPマッチング, A*アルゴリズム)
8	12/09	グラフ(A*アルゴリズム), 前半のまとめ
9	12/15	中間試験

## 授業の予定(中間試験以降)

10	12/16	全文検索アルゴリズム(simple search, KMP)
11	12/22	全文検索アルゴリズム(BM, Aho-Corasick)
12	01/05	全文検索アルゴリズム(Aho-Corasick), データ圧縮
13	01/12	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	01/19	テキスト圧縮(zip), 音声圧縮(ADPCM, MP3, CELP), 画像圧縮(JPEG)
15	01/26	期末試験

## レポート

### 全文検索アルゴリズム(BM)

- Boyer-Moore法のプログラムを作成
  - プログラミング言語は何でも良い
  - プログラムの説明
- データ
  - text: 2種類
    - 1: ABCDABABCDEABCD
    - 2: ZYXWVUTSABCDEFG
  - Key: 2種類
    - 1: AB
    - 2: ABCD
- 結果表示(4種類の実験に対して)
  - キーワード出現位置(あれば複数)
  - 照合回数
- 締め切り: 2月9日(木) 17:00
- 提出場所: 鈴木の居室(A3-K514)前のレポート入れ

## 本日のメニュー

- 全文検索アルゴリズム
  - Aho-Corasickの続き
- 世の中は不公平
  - Zipfの法則
- 不公平を生かす
  - 暗号
  - 符号化
    - モールス信号
    - ハフマン符号
- テキスト圧縮

## 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







## BM法による文字列照合

Method BM

```
begin
  pos:=n;
  while pos<=m do
    begin
      if text[pos]=key[n] then
        begin
          k:=pos-1;
          j:=n-1;
          while j>0 and text[k]=key[j] do
            begin
              k:=k-1;
              j:=j-1;
            end
          if j=0 then
            print k+1;
          end
          pos:=pos+skip(text[pos]);
        end
      end
    end
end
```

m: textの長さ  
n: keywordの長さ  
j: keywordの照合位置  
pos: text中の照合位置

## BM法による文字列照合

### skip関数

入力: キーワード key  
出力: skip関数

文字種: p~q  
n: keyの長さ

Method skip

```
begin
  for i:=p to q do
    skip(i):=n;
  for i:=1 to n-1 do
    skip(key[i]):=n-i;
  end
end
```

初期設定(全ての文字種でkeyの長さだけskip)  
Keyに含まれる文字種の場合keyの先頭から末尾まで調べて最後に見つかった位置をkeyの長さから引いた数だけskipする

## BM法の評価

- 最良の場合 m/n回の文字照合  
textの文字  $\cap$  keyの文字 =  $\phi$
- 最悪の場合 m\*n回の文字照合  
textの文字 = keyの文字 = {a}
- キーワードが長いほど高速
  - keyに含まれない文字がtextに出現したときにkeyの長さだけスキップできる
- 文字種類数が少ないほど遅くなる
  - text中の文字がkey中に現れる確率が高くなる  $\rightarrow$  遅くなる

## Aho-Corasick法

- マシンAC
- AC法の文字列照合手順
- AC法の文字列照合アルゴリズム
- AC法の評価
- マシンACの構成方法

## Aho-Corasick法

- 文書中から**複数**のキーワードを検索するための手法
- テキストをバックトラックすることなく**1回走査する**だけで、複数のキーワードを同時に検出することができる
- goto関数, failure関数, output関数により構成される

## goto関数, failure関数, output関数

- goto関数
  - ある状態で文字xが入力されたときに遷移する状態
- failure関数
  - goto関数からfailが返された際の照合ポインタの移動先
- output関数
  - ある状態に遷移したときに検出できるキーワード

### マシンAC goto関数

キーワード {"ab", "bc", "bab", "d", "abcde"}

ある状態で文字xが入力されたときに遷移する状態

failure関数

### マシンAC failure関数

goto関数からfailが返された際の照合ポインタの移動先

failure関数

s	f(s)
1	0
2	3
3	0
4	0
5	1
6	2
7	0
8	4
9	7
10	0

goto関数

### マシンAC output関数

ある状態に遷移したときに検出できるキーワード

output関数

s	output(s)
2	{"ab"}
4	{"bc"}
6	{"bab", "ab"}
7	{"d"}
8	{"bc"}
9	{"d"}
10	{"abcde"}

goto関数

### 照合ポインタの遷移

テキストストリング "xabcdex"

keyword "ab", "bc", "bab", "d", "abcde"

s	f(s)
1	0
2	3
3	0
4	0
5	1
6	2
7	0
8	4
9	7
10	0

s	output(s)
2	{"ab"}
4	{"bc"}
6	{"bab", "ab"}
7	{"d"}
8	{"bc"}
9	{"d"}
10	{"abcde"}

### 照合ポインタの遷移

テキストストリング "xabcdex"

keyword "ab", "bc", "bab", "d", "abcde"

s	f(s)	s	output(s)
1	0	2	{"ab"}
2	3	4	{"bc"}
3	0	6	{"bab", "ab"}
4	0	7	{"d"}
5	1	8	{"bc"}
6	2	9	{"d"}
7	0	10	{"abcde"}
8	4		
9	7		
10	0		

入力文字 x b a b c d e x

goto関数による遷移 0 → 0 → 3 → 5 → 6 → 8 → 9 → 10 → 0

failure関数による遷移

### 練習問題 照合ポインタの遷移

テキストストリング "abcdcbca"

keyword "ab", "bc", "bab", "d", "abcde"

s	f(s)	s	output(s)
1	0	2	{"ab"}
2	3	4	{"bc"}
3	0	6	{"bab", "ab"}
4	0	7	{"d"}
5	1	8	{"bc"}
6	2	9	{"d"}
7	0	10	{"abcde"}
8	4		
9	7		
10	0		

入力文字 a b c d b c b a

goto関数による遷移

failure関数による遷移

### 練習問題 照合ポインタの遷移 テキストストリング "abcdcbca"

s	output(s)
2	{"ab"}
4	{"bc"}
6	{"bab", "ab"}
7	{"d"}
8	{"bc"}
9	{"d"}
10	{"abcde"}

keyword  
"ab", "bc", "bab", "d", "abcde"

入力文字 a b c d b c b a

goto関数による遷移 0 → 1 → 2 → 8 → 9 3 → 4 3 → 5

Failure関数による遷移  
9 → 7  
4 → 0

### マシンACの構成方法

- goto関数とoutput関数の構成方法
- failure関数の構成方法

### goto関数とoutput関数の構成方法 1/2

abを追加 0-a-1-b-2 output(2)={"ab"}

bcを追加 0-a-1-b-2 output(2)={"ab"}, 0-b-3-c-4 output(4)={"bc"}

babを追加 0-a-1-b-2 output(2)={"ab"}, 0-b-3-c-4 output(4)={"bc"}, 0-a-5-b-6 output(6)={"bab"}

dを追加 0-a-1-b-2 output(2)={"ab"}, 0-b-3-c-4 output(4)={"bc"}, 0-a-5-b-6 output(6)={"bab"}, 0-d-7 output(7)={"d"}

keyword  
"ab" (2)  
"bc" (4)  
"bab" (6)  
"d" (7)  
"abcde" (10)

### goto関数とoutput関数の構成方法 2/2

abcdeを追加 0-a-1-b-2-c-8-d-9-e-10 output(10)={"abcde"}, 0-a-1-b-2-c-8-d-9-e-10 output(2)={"ab"}, 0-b-3-c-4 output(4)={"bc"}, 0-a-5-b-6 output(6)={"bab"}, 0-d-7 output(7)={"d"}

キーワード以外の時の処理を追加  
9 → 7  
4 → 0

keyword  
"ab"  
"bc"  
"bab"  
"d"  
"abcde"

### failure関数の構成方法

状態sのfailure関数  
f(s)=q | ACstring[q]がACstring[s]の最長の接尾辞になる状態q

keyword  
"ab"  
"bc"  
"bab"  
"d"  
"abcde"

1: a → 0  
2: ab → 3  
3: b → 0  
4: bc → 0  
5: ba → 1  
6: bab → 2  
7: d → 0  
8: abc → 4  
9: abcd → 7  
10: abcde → 0

babの最長の接尾辞

s	f(s)
1	0
2	3
3	0
4	0
5	1
6	2
7	0
8	4
9	7
10	0

### データ圧縮

- 対象データ
  - テキスト
  - 音声
    - 音楽
    - 話し声
  - 画像
  - 動画
- 圧縮方式
  - 可逆圧縮 (ロスレス圧縮)
  - 非可逆圧縮 (ロシシー圧縮)

## 本日後半のメニュー

- 世の中は不公平
  - Zipfの法則
- 不公平を生かす
  - 暗号
  - 符号化
    - モールス信号
    - ハフマン符号
- テキスト圧縮

## 世の中は不公平...

### だからおもしろい

- 頻度分布の偏り
  - 例: 株取引, FX, 麻雀, ブラックジャック
- 自分だけが知っている(つもりの)頻度の偏りを利用して得をする
  - 株価チャートを解読
  - 麻雀で山読みして勝つ
  - ブラックジャックでカードカウンティングする
  - 試験で山を掛ける(張る)
- 確率を無理やり変える
  - 偽情報を流して株価操作(犯罪行為)
  - スティング(映画)のポーカー(DVDで確認)
  - 試験範囲を満遍なく勉強する(効果絶大)
  - 授業中, 指名されないように下を向く(逆効果)

## ジップの法則(Zipf's law)

「あるタイプの現象が生起する確率はその現象の生起する順位に反比例する」: 経験則

$$\text{生起確率} = \frac{\text{定数}C}{\text{順位}}$$

- Zipfの法則が当てはまる事象
  - 文字毎の出現頻度
  - コンピュータにおけるコマンドの使用頻度
  - Webページのアクセス頻度
  - 都市の人口
  - 文献の参照回数
  - 会社でのランク(役職)と給料など
  - ケータイのシェア(docomo, au, softbank, e-mobile)

## 携帯電話: 各グループ毎の加入者数累計 (2009年12月 ケータイWatchより)

順位	事業者	累計	割合(確率)	Zipf's law C=0.51
1	NTTドコモ	55,297,200	50.2%	51.0%
2	KDDI	31,329,400	28.4%	25.5%
3	ソフトバンク	21,501,900	19.5%	17.0%
4	イー・モバイル	2,048,200	1.8%	12.8%

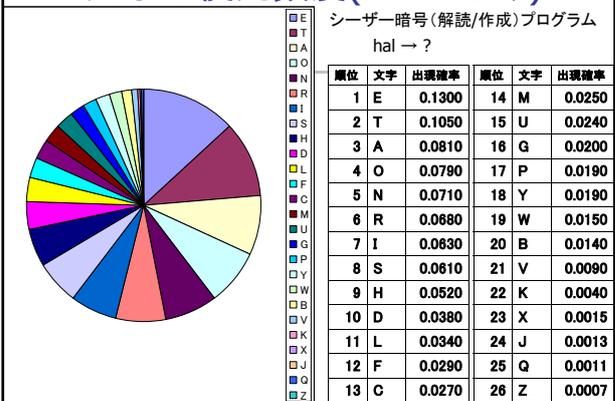
$$\text{生起確率} = \frac{\text{定数}C}{\text{順位}}$$

## 自然言語の統計的性質

- 文字の使用頻度(英語) \_はスペース

順位	文字	%	2	%	3	%	4	%
1	_	17.4	e_	3.0	_th	1.6	_the	1.2
2	e	9.7	_t	2.4	the	1.3	the_	1.0
3	t	7.0	th	2.0	he_	1.3	_of_	0.6
4	a	6.1	he	1.9	_of	0.6	and_	0.4
5	o	5.9	_a	1.7	of_	0.6	_and	0.4
6	i	5.5	s_	1.7	ed_	0.5	_to_	0.4
7	n	5.5	d_	1.5	_an	0.5	ing_	0.3

## 文字の使用頻度(caesarより)



## 単語の使用頻度

順位	単語	%	2	%	3	%
1	the	6.1	of the	0.9	one of the	0.03
2	of	3.5	in the	0.5	as well as	0.02
3	and	2.7	to the	0.3	the United States	0.02
4	to	2.5	on the	0.2	out of the	0.02
5	a	2.1	and the	0.2	some of the	0.01
6	in	1.9	for the	0.1	the end of	0.01
7	that	0.9	to be	0.1	the fact that	0.01

## 単語の出現頻度分布

### ジップの法則(Zipf's law):

- 単語の出現順位 ( $r$ ) と出現頻度 ( $f$ ) は反比例の関係にある

$$r = \frac{C}{f} \quad f = \frac{C}{r}$$

$n$  番目の単語の出現確率  $P_n$

$$P_n = \frac{C}{n}$$

順位	文字	出現確率	0.065/順位
1	the	0.061	0.065
2	of	0.035	0.0325
3	and	0.027	0.0108333
4	to	0.025	0.0027083
5	a	0.021	0.0005417
6	in	0.019	0.00009028
7	that	0.009	0.0000129

$C$  は定数  
低頻度の語には当てはまらない

## データの頻度分布の偏りを利用した技術

- 暗号(換字式)の解読
  - 小説(ポー, ドイルなど)
  - シーザー暗号
- データ圧縮(ロスレス)
  - キー入力時の打鍵回数の削減
  - モールス符号
  - ハフマン符号(情報理論 2年前期 宮本先生)

## 小説中での暗号解読の解説

- 黄金虫(The gold bug)
  - 著者: エドガー・アラン・ポー
  - 作品: 翻訳版
    - <http://www.aozora.gr.jp/cards/000094/card2525.html>
  - 作品: 原文
    - [http://en.wikisource.org/wiki/The\\_Gold-Bug](http://en.wikisource.org/wiki/The_Gold-Bug)
- 踊る人形(The Adventure of the Dancing Men)
  - 著者: アーサー・コナン・ドイル
  - 作品: 翻訳版 題: 暗号舞踏人の謎
    - <http://www.aozora.gr.jp/cards/000009/card45340.html>
  - 作品: 原文
    - [http://en.wikisource.org/wiki/The\\_Adventure\\_of\\_the\\_Dancing\\_Men](http://en.wikisource.org/wiki/The_Adventure_of_the_Dancing_Men)