

アルゴリズムとデータ構造III 12回目:1月8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/algorithm3/index.html>

1

授業の予定(中間試験まで)

| | | |
|---|-------|------------------------------------|
| 1 | 10/02 | スタック(後置記法で書かれた式の計算) |
| 2 | 10/09 | チューリング機械, 文脈自由文法 |
| 3 | 10/16 | 構文解析 CYK法 |
| 4 | 10/23 | 構文解析 CYK法 |
| 5 | 10/30 | 構文解析(チャート法), グラフ(ダイクストラ法) |
| 6 | 11/06 | 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 | 11/13 | グラフ(DPマッチング, A*アルゴリズム) |
| 8 | 11/20 | グラフ(A*アルゴリズム), 前半のまとめ |
| 9 | 11/27 | 中間試験 |

2

授業の予定(中間試験以降)

| | | |
|----|-------|---|
| 10 | 12/04 | 全文検索アルゴリズム(simple search, KMP) |
| 11 | 12/18 | 全文検索アルゴリズム(BM, Aho-Corasick) |
| 12 | 01/08 | 全文検索アルゴリズム(Aho-Corasick), データ圧縮 |
| 13 | 01/15 | 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 | 01/29 | テキスト圧縮(zip), 音声圧縮(ADPCM, MP3, CELP), 画像圧縮(JPEG) |
| 15 | 02/05 | 期末試験 |

3

本日のメニュー

- 全文検索アルゴリズム
 - Aho-Corasickの続き
- 暗号
 - 黄金虫(The gold bug)
 - 踊る人形(The Adventure of the Dancing Men)
- 符号化
- テキスト圧縮

4

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

5

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: aabcdabdabbabcabacade
 - 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 | | | | | | | |

6

文字列照合アルゴリズム

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

7

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

8

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 | | | | | | | | | | | | |

9

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 | b | c | a | b | a | | | | | | | | | | | | | | | | |
| | | | a | | | | | | | | | | | | | | | | | | | | |
| | | | | a | b | c | a | | | | | | | | | | | | | | | | |
| | | | | | a | | | | | | | | | | | | | | | | | | |
| | | | | | | a | b | c | a | b | a | | | | | | | | | | | | |
| | | | | | | | a | | | | | | | | | | | | | | | | |
| | | | | | | | | a | b | c | | | | | | | | | | | | | |
| | | | | | | | | | 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
    begin
      for j:=1 to n do
        if text[i+j-1]≠key[j] then
          goto 1;
      print i;
    end
  end
end
  
```

11

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 $m \gg n$ 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 |
| 照合回数 | 1 | 2 | 3 | 3 | 3 | 2 | 1 |

12

Knuth-Morris-Pratt法 (KMP法)

Simple Search

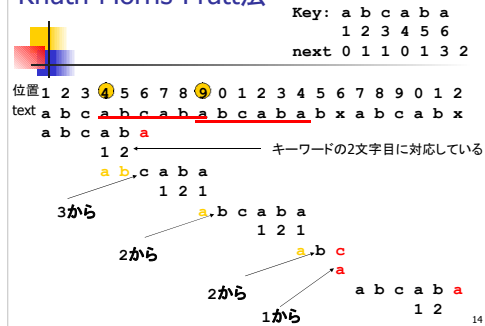
- テキストストリング中の各文字がキーワードと複数回照合される → 冗長

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

13

Knuth-Morris-Pratt法



KMP法 アルゴリズム

```
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);  つぎの照合位置
      if j=n then
        print i-n+1: 照合成功
      j:=j+1;
    end
  end
end
```

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

15

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 | |
| | | | | | | a | b | c | a | b |
| next関数値 | 0 | 1 | 1 | 0 | 1 | 3 | 2 | | | |

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
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

16

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

17

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

```

Method next
begin
  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
  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の接頭辞と一致しているか調べる
    t:=t+1;
    if key[j+1]=key[t] then
      keyのj+1文字目のnext関数値を決定
      next(j+1):=t;
    else
      next(j+1):=t;
    end
  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 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照合
- m: テキスト文字列数
n: キーワード文字列数

Boyer-Moore法

- キーワードの末尾から照合を行う。
- キーワードの末尾と照合したテキストストリングの文字を覚えておく
- その文字とキーワードの文字が一致するまでキーワードをずらす

Boyer-Moore法

Key: a b c a b a

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 c a b x

key a b c a b a

3文字右へ

2文字右へ

3文字右へ

2文字右へ

6文字右へ

| 文字 | skip関数値 |
|---------|---------|
| a | 2 |
| b | 1 |
| c | 3 |
| 上記以外の文字 | 6 |

skip関数

- テキスト文字列中の照合文字cが、キーワードの末尾から何文字目にあるか

?????a
abcaba
6543210

?????b
abcaba
6543210

?????c
abcaba
6543210

?????x
abcaba
6543210

キーワード"abcaba"に対するskip関数

| 文字 | skip関数値 |
|---------|---------|
| a | 2 |
| b | 1 |
| c | 3 |
| 上記以外の文字 | 6 |

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
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

```

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

25

BM法の評価

- 最良の場合 m/n回の文字照合
textの文字 ∩ keyの文字 = ϕ
- 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 → 遅くなる

26

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

27

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
- テキストストリング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

28

goto関数, failure関数, output関数

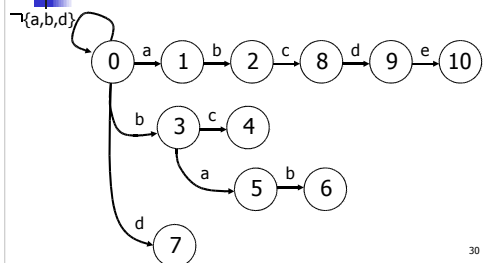
- goto関数
 - ある状態で文字xが入力されたときに遷移する状態
- failure関数
 - goto関数からfailが返された際の照合ポインタの移動先
- output関数
 - ある状態に遷移したときに検出できるキーワード

29

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



30

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

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

31

マシンAC output関数

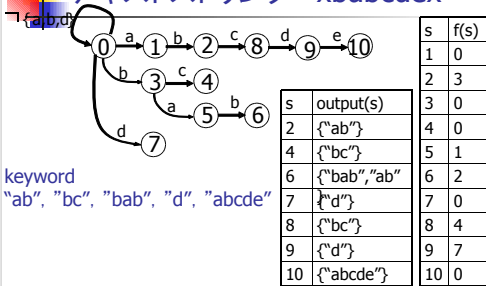
ある状態に遷移したときに検出できるキーワード

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

32

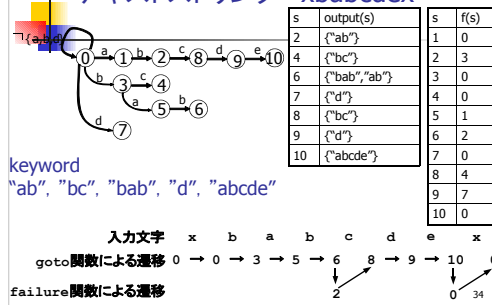
照合ポインタの遷移

テキストストリング "xbabcde"



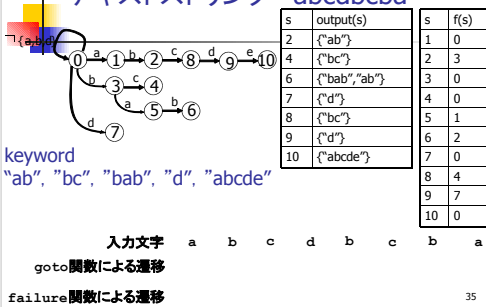
照合ポインタの遷移

テキストストリング "xbabcde"



練習問題 照合ポインタの遷移

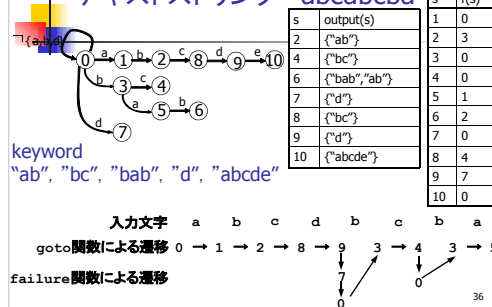
テキストストリング "abcdcbca"



35

練習問題 照合ポインタの遷移

テキストストリング "abcdcbca"



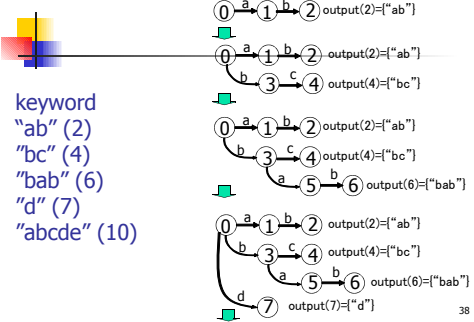
36

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

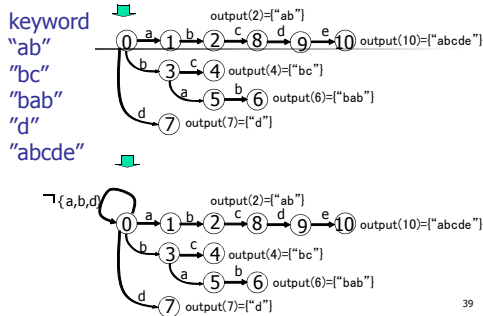
37

goto関数とoutput関数の構成方法 1/2



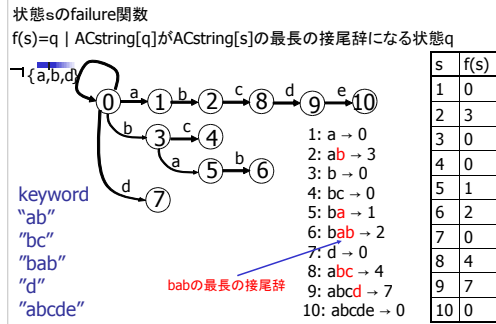
38

goto関数とoutput関数の構成方法 2/2



39

failure関数の構成方法



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 (ロッシー圧縮)

41

モールス信号の符号

- ・ (短点) と - (長点) を用いてアルファベットを表現する
- 情報を早く送るための工夫
 - よく使われる文字 (例えば e, t) は短い
 - e: ・ (短点1文字)
 - t: - (長点1文字)
 - あまり使われない文字 (例えば q は4文字) は長い
 - q: - - - -

42

モールス信号の符号



- (短点)と-(長点:短点3つ分の長さ)を用いてアルファベットを表現する
- 区切り記号
 - 文字の切れ目:短点3つ分の間隔
 - 単語の切れ目:短点7つ分の間隔
- L: ·-·· (LifeカードのCMIに使われていた)
- SOS:··· ---- ···