講義「情報知識ネットワーク特論」
～情報検索とパターン照合

第１回　準備：用語と予備知識

情報理工学専攻 情報知識ネットワーク研究室
喜田拓也

今日の内容

パターン照合問題とは

逐次検索と索引データ構造を用いた検索

テキストアルゴリズムの基本用語

有限オートマトン

パターン照合問題とは？

テキストT 中に含まれるパターン P の出現を求める問題

パターン P: compress

テキスト T:

We introduce a general framework which is suitable to capture an essence of compressed pattern matching according to various dictionary based compressions. The goal is to find all occurrences of a pattern in a text without decompression, which is one of the most active topics in string matching. Our framework includes such compression methods as Lempel-Ziv family, (LZ77, LZSS, LZ78, LZW), byte-pair encoding, and the static dictionary based method. Technically, our pattern matching algorithm extremely extends that for LZW compressed text presented by Amir, Benson and Farach [Amir94]..

有名なアルゴリズム：

- KMP法 (Knuth&Morris&Pratt[1974])
- BM法 (Boyer&Moore[1977])
- Karp-Rabin法 (Karp&Rabin[1987])

パターン照合問題の種類

Existence problem:

テキスト T: プルルンプルンファミファミファ

パタン P: ファミファ

Yes!

All-occurrences problem:

テキスト T: プルルンプルンファミファミファ

パタン P: ファミファ

8　11

文書を単位として検索する場合は，Existence problemで充分

本講義では，基本的にAll-Occurrences problemを取り扱う

本問題に対する2つのアプローチ

文字列照合による検索

長所：
- ✓ 余計なデータ構造が不要
- ✓ データの更新に対して柔軟

短所：
- ✓ 検索がおそい　$O(n)$
- ✓ スケーラビリティが低い

小規模文書群に対する検索向き
(例：UNIX の grep)

本講義では，主にこちらをします

索引データ構造を用いた検索

長所：
- ✓ 検索がはやい　$O(m \log n)$
- ✓ スケーラビリティが高い

短所：
- ✓ 索引構造を構築する手間がかかる
- ✓ データの更新に対して柔軟性に欠ける
- ✓ 索引構造のためのスペースが必要

中・大規模DBに対する検索向き
(例：namazu, sufary, Googleほか)

索引技術（転置索引，接尾辞木，接尾辞配列，etc）

基本用語(計算量の記法)

アルゴリズムの良し悪しを明確にするためには，計算の複雑さを明確に判断する必要がある
計算にかかる時間や必要となる主記憶容量(メモリ量)が入力(データ)長 n に対してどのくらいかかるのか？

漸近的計算量の上界を示す記法
(下界を示す Ω 記法もある)

big-O 記法：

fとgを整数から整数への関数とする．このとき，ある定数CおよびNについて，任意の$n > N$に対し$f(n) < C \cdot g(n)$ならば，$f(n) = O(g(n))$と書く．このとき，fはオーダーgという．

fとgが同じオーダーのとき，すなわち$f(n) = O(g(n))$かつ$g(n) = O(f(n))$が成り立つとき，$f = \Theta(g)$と書く．

計算時間を入力長nの関数$T(n)$で表したとき，$T(n) = O(g(n))$であるとする．これはすなわち，「漸近的には高々$g(n)$に比例した時間しかかからない」ということを意味している．

例：$O(n)$，$O(n \cdot \log n)$

テキストアルゴリズムの基本用語

Σ : 文字（letters, characters, またはsymbols）の空でない集合.
これをアルファベットと呼ぶ.
例）$\Sigma = \{a, b, c, \ldots, z\}$, $\Sigma = \{0,1\}$, $\Sigma = \{0x00, 0x01, \ldots, 0xFF\}$

$x \in \Sigma^*$: アルファベット中の文字を0個以上並べたもので，文字列（string）と呼ぶ．語（word）とかテキスト（text）と呼ばれることもある.

$|x|$: 文字列 x の長さ．例）$|aba| = 3$.

ε : 長さが 0 の文字列．空語（empty word）と呼ぶ．$|\varepsilon| = 0$.

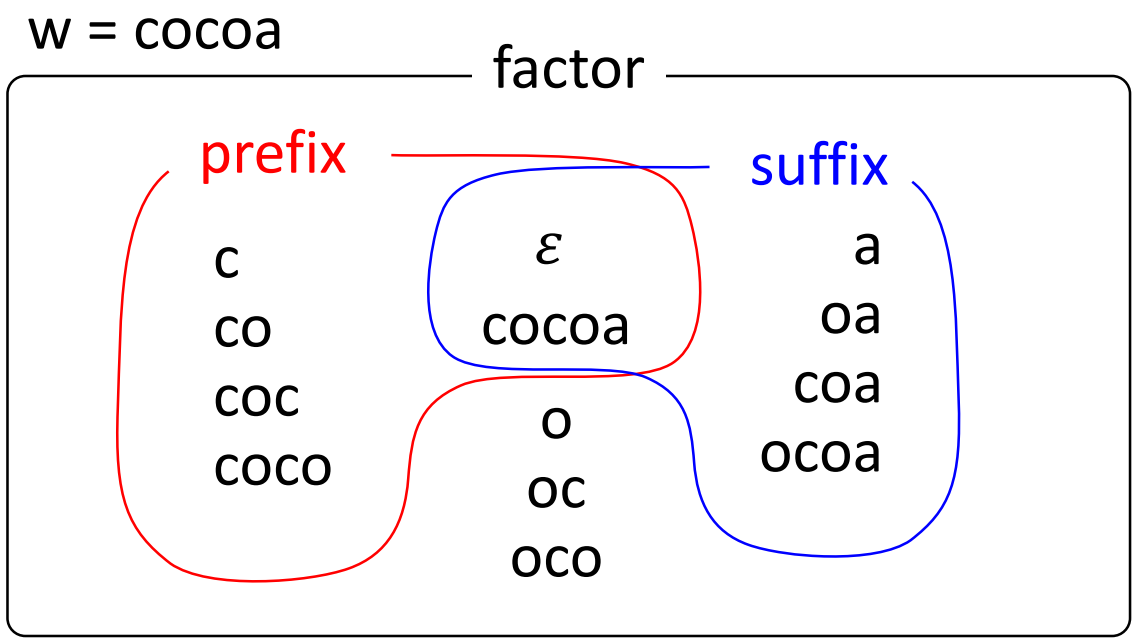
$x[i]$: 文字列 x の i 番目の文字.

$x[i:j]$: 文字列 x の i 番目から j 番目までの連続した文字のならび.
ただし，$i > j$ の場合は，便宜上 $x[i:j] = \varepsilon$ とする．$x[i:j]$ を，文字列 x の部分文字列（substring, subword）またはファクター（factor）と呼ぶ．$x[i..j]$と書くこともある.

x^R : 文字列 $x = a_1 a_2 \ldots a_k \in \Sigma^*$ に対して，これを反転させた文字列.
すなわち，$x^R = a_k a_{k-1} \ldots a_1$.

接頭辞(Prefix)と接尾辞(Suffix)

部分文字列(factor)のうち，$x[1:i]$ を特に x の接頭辞(prefix)という．
また，$x[i:|x|]$ を特に x の接尾辞(suffix)という．



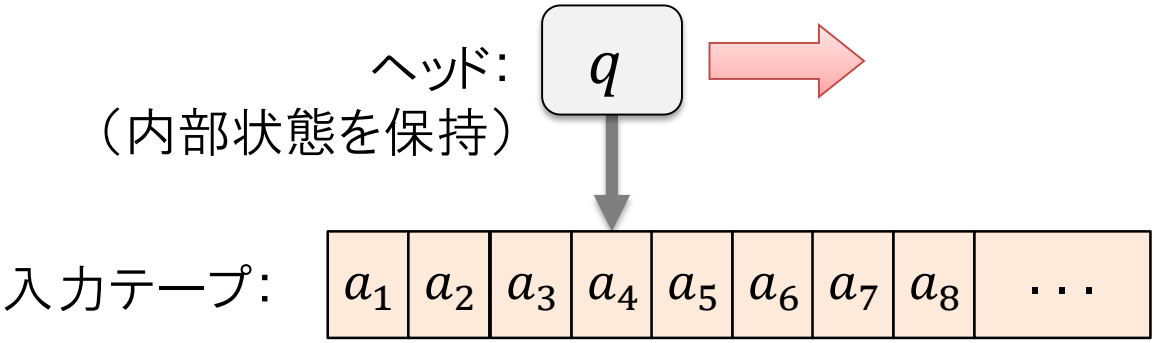
文字列 x と y について，y から 0 文字以上の文字を取り除くと，
x に等しくなるとき，x を y の部分列(subsequence)という
例：$x = \mathrm{abba}$ は，$y = \mathrm{aaababaab}$ の部分列

部分文字列との違いに注意

有限オートマトン(Finite Automaton)

計算機科学分野の基礎的概念！

文字列の入力を読み取りながら内部状態を変化させ，最終的にその文字列を受理するか否かを決定する自動機械．状態遷移の定義の仕方や補助記憶領域の有無によって，さまざまな種類（非決定性オートマトン，プッシュダウンオートマトンなど）がある



（機械的な）言語を定義する能力があり，文字列の字句解析に用いられることが多い

パターン照合問題にも深く関係がある！

参考文献：
「オートマトンと計算可能性」 有川節夫・宮野悟著，培風館
「形式言語とオートマトン」 守屋悦朗著，サイエンス社

決定性有限オートマトン

決定性有限オートマトン M は5つ組 $(K, \Sigma, q_0, \delta, F)$ で定義される.

K : Mの内部状態の集合 $\{q_0, q_1, \ldots, q_n\}$
Σ : 入力データのアルファベット $\{a_0, a_1, \ldots, a_k\}$
q_0 : Mの最初の内部状態. 初期状態と呼ばれる
δ : 入力データから1文字を読んだとき, 次の状態を返す関数.
遷移関数と呼ばれる. $K \times \Sigma \to K$
F : 受理状態(終状態)の集合. $F \subseteq K$

遷移関数の定義域を次のように $K \times \Sigma$ から $K \times \Sigma^*$ へ拡張する.

$$\delta(q, \varepsilon) = q \quad (q \in K)$$
$$\delta(q, ax) = \delta(\delta(q, a), x) \quad (q \in K, a \in \Sigma, x \in \Sigma^*)$$

すると, $\delta(q_0, w)$ は, 文字列 w を入力したときの状態を表す.

$\delta(q_0, w) = p \in F$ であるとき, M は w を受理するという.

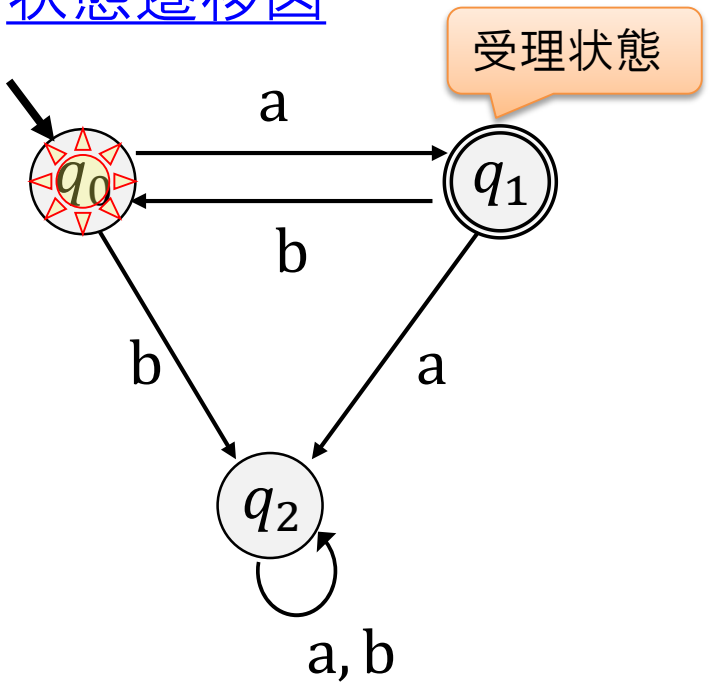
M が受理する文字列全体 $L(M) = \{w \mid \delta(q_0, w) \in F\}$ を
有限オートマトン M によって受理される言語という.

決定性有限オートマトンの例

状態遷移

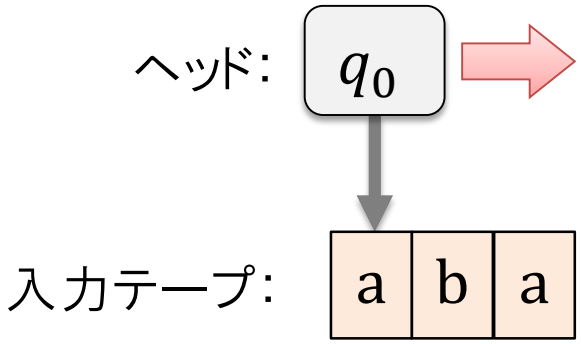
$$\begin{aligned}\delta(q_0, \mathrm{aba}) &= \delta(\delta(q_0, \mathrm{a}), \mathrm{ba}) \\ &= \delta(q_1, \mathrm{ba}) \\ &= \delta(\delta(q_1, \mathrm{b}), \mathrm{a}) \\ &= \delta(q_0, \mathrm{a}) \\ &= q_1 \in F\end{aligned}$$

状態遷移図



Mが受理する言語

$$L(M) = \{\mathrm{a}(\mathrm{ba})^n \mid n \geq 0\}$$



非決定性有限オートマトン

非決定性有限オートマトン M は次の5つ組 $(K, \Sigma, Q_0, \delta, F)$ で定義される

K ：Mの内部状態の集合 $\{q_0, q_1, \ldots, q_n\}$

Σ ：入力データのアルファベット $\{a_0, a_1, \ldots, a_k\}$

Q_0：Mの初期状態の集合．$Q_0 \subset K$

δ ：Mの遷移関数．$K \times \Sigma \to 2^K$

F ：受理状態（終状態）の集合．$F \subseteq K$

つまり，遷移する先が一意には決まらない！

現在の状態が複数存在しうる

遷移関数の定義域を次のように $K \times \Sigma$ から $K \times \Sigma^*$ へ拡張する．

$$\delta(q, \varepsilon) = \{q\}$$

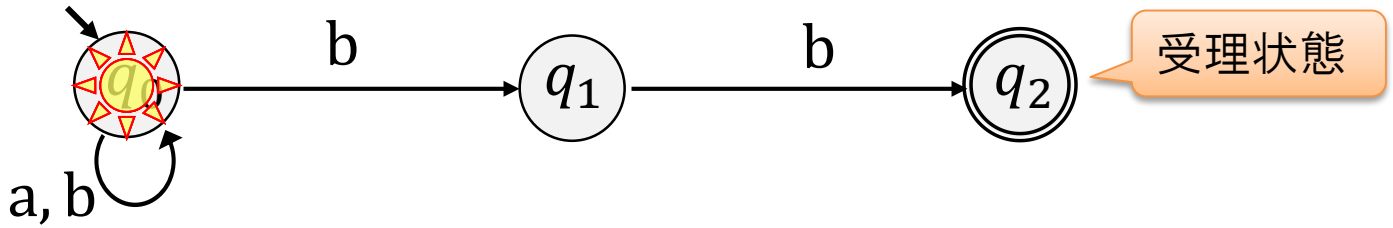
$$\delta(q, ax) = \cup_{p \in \delta(q,a)} \delta(p, x) \qquad (q \in K, a \in \Sigma, x \in \Sigma^*)$$

さらに，$2^K \times \Sigma^*$ に拡張する．

$$\delta(S, x) = \cup_{q \in S} \delta(q, x)$$

$x \in \Sigma^*$ について $\delta(Q_0, x) \cap F \neq \phi$ であるとき，M は x を受理するという

非決定性オートマトンの例



非決定性有限オートマトンの状態図

例えば abb に対しては　$\delta(\{q_0\}, \mathrm{abb}) = \delta(\{q_0\}, \mathrm{bb})$

$= \delta(\{q_0\}, \mathrm{b}) \cup \delta(\{q_1\}, \mathrm{b})$

$= \{q_0\} \cup \{q_1\} \cup \{q_2\}$

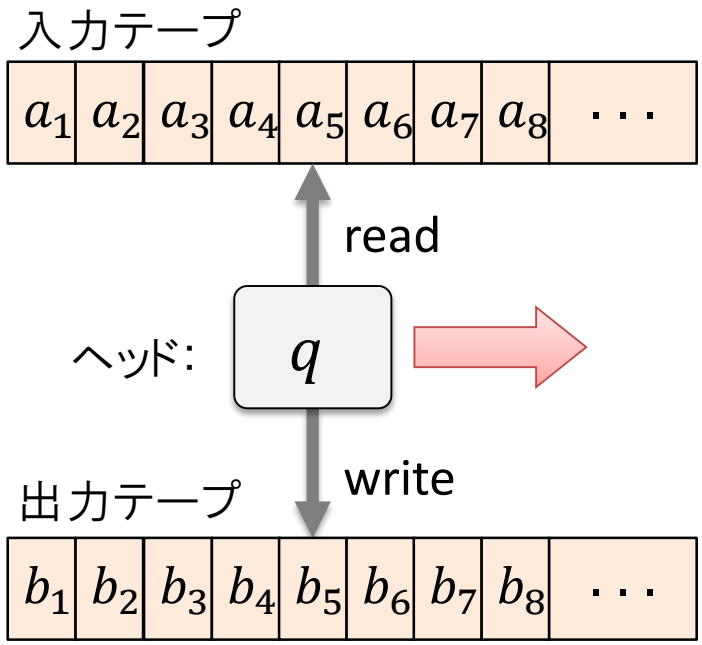
$= \{q_0, q_1, q_2\}$

となり $q_2 \in F$ であるから $\mathrm{abb} \in L(M)$ である.

順序機械

順序機械とは，入力に対して逐次に出力がある有限オートマトンで，
次の6つ組 $(K, \Sigma, \Delta, q_0, \delta, \lambda)$ で定義される．

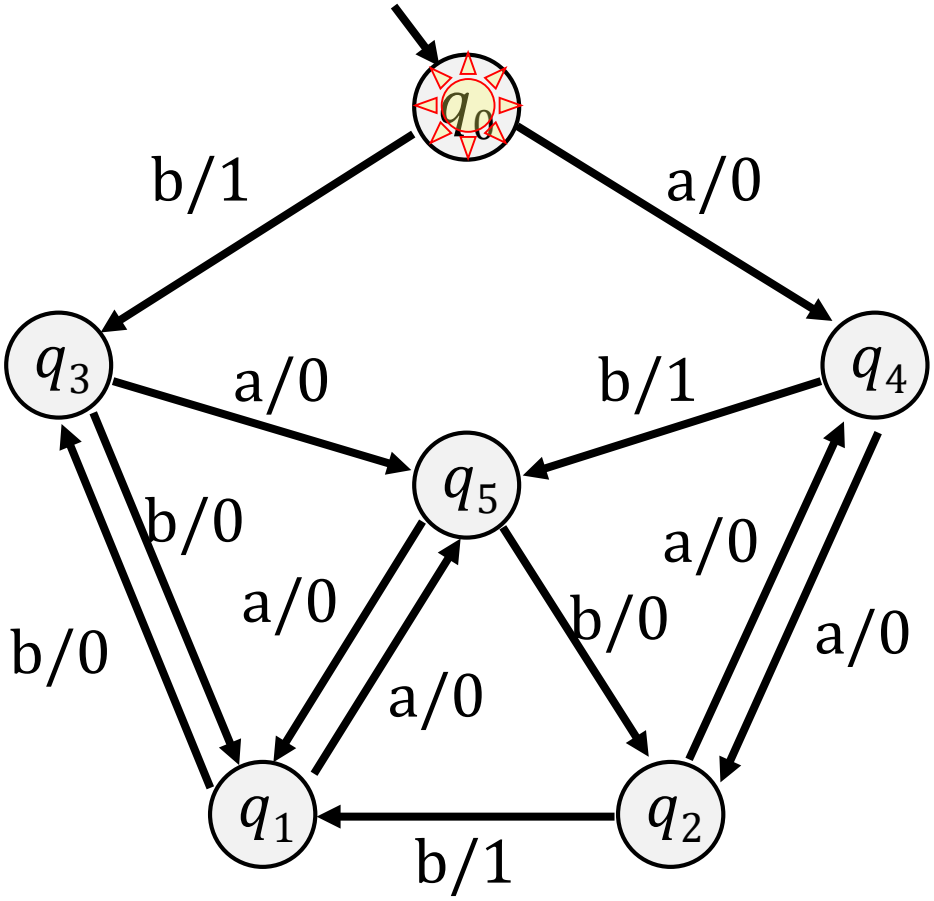
K ：内部状態の集合 $\{q_0, q_1, \ldots, q_n\}$
Σ ：入力データのアルファベット
$\{a_0, a_1, \ldots, a_k\}$
Δ ：出力データのアルファベット
$\{b_0, b_1, \ldots, b_\ell\}$
q_0 ：初期状態．
δ ：遷移関数．$K \times \Sigma \to K$
λ ：出力関数．$K \times \Sigma \to \Delta$



遷移関数の定義域を有限オートマトンと同様に $K \times \Sigma^*$ 上の関数へと拡張し，さらに出力関数 λ を次のようにして $K \times \Sigma^*$ から Δ^* への関数に拡張する．

$$\lambda(q, \varepsilon) = q \quad (q \in K)$$
$$\lambda(q, ax) = \lambda(q, a)\lambda(\delta(q, a), x) \quad (q \in K, a \in \Sigma, x \in \Sigma^*)$$

順序機械の例



順序機械の状態図

出力関数の様子

$\lambda(q_0, \mathrm{abb})$

$= \lambda(q_0, \mathrm{a})\lambda(\delta(q_0, \mathrm{a}), \mathrm{bb})$

$= 0\lambda(q_4, \mathrm{bb})$

$= 0\lambda(q_4, \mathrm{b})\lambda(\delta(q_4, \mathrm{b}), \mathrm{b})$

$= 01\lambda(q_5, \mathrm{b})$

$= 010$

一種の翻訳機！

第1回 まとめ

パターン照合問題とは，
　テキストT中に含まれるパターンPの出現を求める問題
　Existence problemとAll-occurrence problemがある

索引データ構造を用いた検索との違い
　文字列照合による検索は，索引データ構造を用いた検索に
　比べて遅いというのが一般的な見方だが，優れた利点もある.

テキストアルゴリズムの基本用語
　計算量の記法：big-O記法
　アルファベット，文字列，Prefix，Factor，Suffix

有限オートマトン
　決定性有限オートマトン：　言語を定義できる
　非決定性有限オートマトン：　現在の状態と遷移先が複数ある
　順序機械：　入力に対して，逐次に出力がある