



情報知識ネットワーク特論 「情報検索とパターン照合」

情報科学研究科 コンピュータサイエンス専攻
情報知識ネットワーク研究室

喜田拓也

第1回

準備：用語と予備知識

パターン照合問題とは
テキストアルゴリズムの基本用語
有限オートマトンについて
索引データ構造を用いた検索との違い



パターン照合問題 (Pattern Matching Problem) 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
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 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].



Existence problem と All-occurrences problem

Existence problem

テキスト T: テクマクマヤコンテクマクマヤコン
パターン P: クマクマ

Yes!

All-occurrences problem

テキスト T: テクマクマヤコンテクマクマヤコン
パターン P: クマクマ

2

10

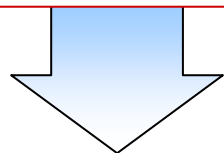
全文検索で文書を単位として検索する場合は、Existence problemで充分
しかし、一般的にはAll-Occurrences problemを指すことが多い



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

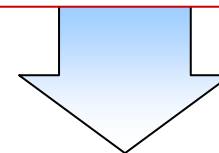
- 長所
 - ✓ 余計なデータ構造が不要
 - ✓ データの更新に対して柔軟
- 短所
 - ✓ 検索がおそい
 - ✓ スケーラビリティが低い



小規模文書群に対する検索向き
(例: UNIX の grep)

索引データ構造を用いた検索

- 長所
 - ✓ 検索がはやい
 - ✓ 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, mg, Googleほか)



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？

- big-O記法

- 定義:

- 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。
(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、いま $T(n) = O(g(n))$ 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。

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)

例: $O(n)$ 、 $O(n \cdot \log n)$



基本用語 (アルファベット、文字列の定義ほか)

■ 用語の定義

- Σ : **文字**の空でない有限集合 (**アルファベット**)
 - 例: $\Sigma = \{a, b, c, \dots\}$, $\Sigma = \{0, 1\}$, $\Sigma = \{0x00, 0x01, 0x02, \dots, 0xFF\}$
- $x \in \Sigma^*$: **文字列** (string)、**語** (word)、**テキスト** (text)
 - $|x|$: 文字列 x の長さ 例: $|aba| = 3$
 - ε : 長さが0の文字列 = **空語** (empty word) という。すなわち $|\varepsilon| = 0$
- $x[i]$: 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 **連続した文字の並び**
 - $i > j$ の場合は、便宜上 $x[i..j] = \varepsilon$ とする
 - $x[1..i]$ を特に x の **接頭辞** (prefix) という
 - $x[i..|x|]$ 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: $x = abba$ は、 $y = aaababaab$ の部分列
- $x = a_1 a_2 \dots a_k \in \Sigma^*$ に対して、これを反転させた文字列 $a_k a_{k-1} \dots a_1$ を x^R と表記する

文字 $a \in \Sigma$ は、
letters, characters, symbols
とも呼ばれる

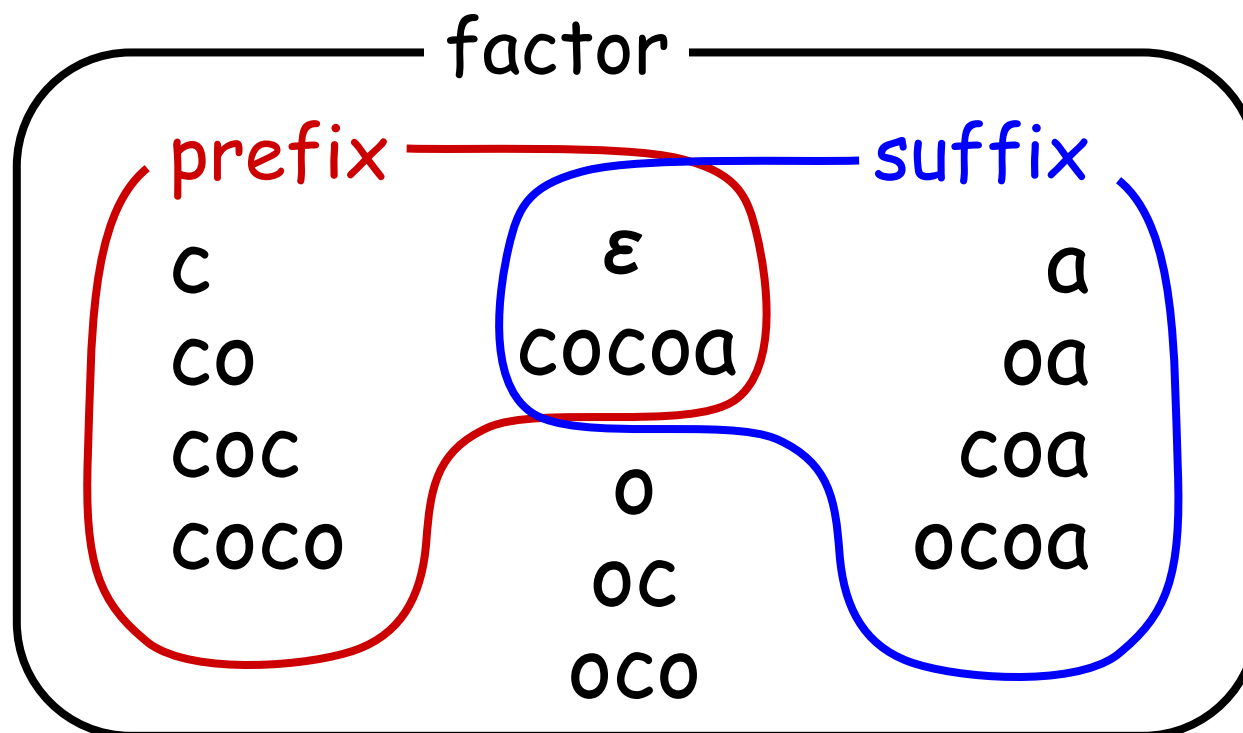
部分文字列 (substring, subword)
あるいは **ファクター** (factor) と呼ばれる

部分文字列との違いに注意!



Prefix, Factor, Suffixの例

$W = \text{COCO}a$



演習1: 語 $w = \text{ROYCE}$ のprefixを列挙せよ。

演習2: 語 $w = \text{ろくろくび}$ のfactorを列挙せよ。



有限オートマトン

- 有限オートマトン(Finite Automaton)
 - 出力が入力と内部状態に応じて決定される自動機械 (Automata)
 - 状態数は有限個である
 - 状態遷移や出力の定義や補助記憶領域の有無によって、さまざまな種類がある
 - 非決定性オートマトン、順序機械、プッシュダウンオートマトンなど
 - (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
 - 計算機科学分野ではあらゆる方面ででてくる基礎的概念！
 - ソフトウェア設計 (UMLの状態遷移図) などにも用いられる！
 - もちろん、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



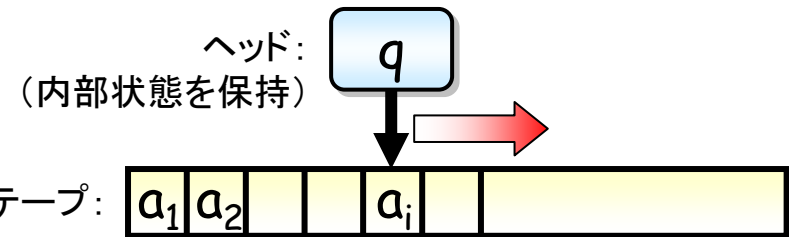
(決定性)有限オートマトンの定義

■ (決定性)有限オートマトン $M = (K, \Sigma, \delta, q_0, F)$

- $K = \{q_0, q_1, \dots, q_n\}$: 状態の集合
- $\Sigma = \{a, b, \dots, c\}$: 文字の集合(アルファベット)
- q_0 : 初期状態
- δ : 遷移関数 $K \times \Sigma \rightarrow K$
- F : 受理状態の集合(K の部分集合)

■ 遷移関数について

- 次のようにして定義域を拡張する
 - $\delta(q, e) = q$ ($q \in K$)
 - $\delta(q, ax) = \delta(\delta(q, a), x)$ ($q \in K, a \in \Sigma, x \in \Sigma^*$)
- 文字列 w に対して $\delta(q_0, w)$ は、語 w を入力したときの状態をあらわす。
- $\delta(q_0, w) = p \in F$ 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。





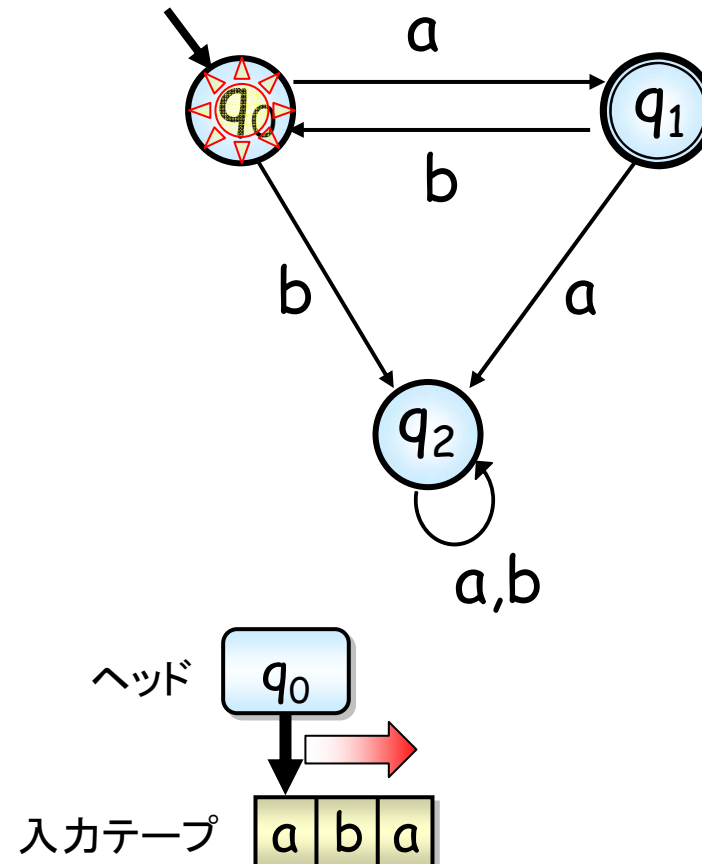
(決定性)有限オートマトンの例

状態遷移

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

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

状態遷移図(図2)





非決定性の有限オートマトンの定義

■ 非決定性有限オートマトン $M=(K, \Sigma, \delta, Q_0, F)$

- $K = \{q_0, q_1, \dots, q_n\}$: 状態の集合
- $\Sigma = \{a, b, \dots, c\}$: 文字の集合(アルファベット)
- $Q_0 \subset K$: 初期状態の集合
- δ : 遷移関数 $K \times \Sigma \rightarrow 2^K$
- F : 受理状態の集合 (K の部分集合)

つまり、遷移する先が一意には決まらない!

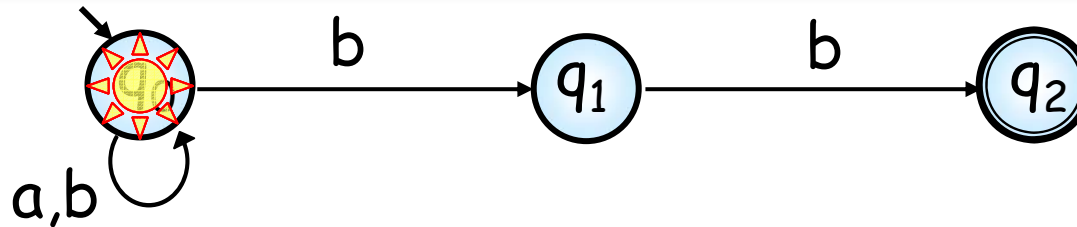
現在の状態が複数存在

■ 遷移関数について:

- 遷移関数の定義域を次のようにして $K \times \Sigma$ から $K \times \Sigma^*$ へ拡張する
 - $\delta(q, e) = \{q\}$
 - $\delta(q, ax) = \bigcup_{p \in \delta(q, a)} \delta(p, x)$ ($q \in K, a \in \Sigma, x \in \Sigma^*$)
- さらに $2^K \times \Sigma^*$ に拡張
 - $\delta(S, x) = \bigcup_{q \in S} \delta(q, x)$
- $x \in \Sigma^*$ について $\delta(Q_0, x) \cap F \neq \emptyset$ 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

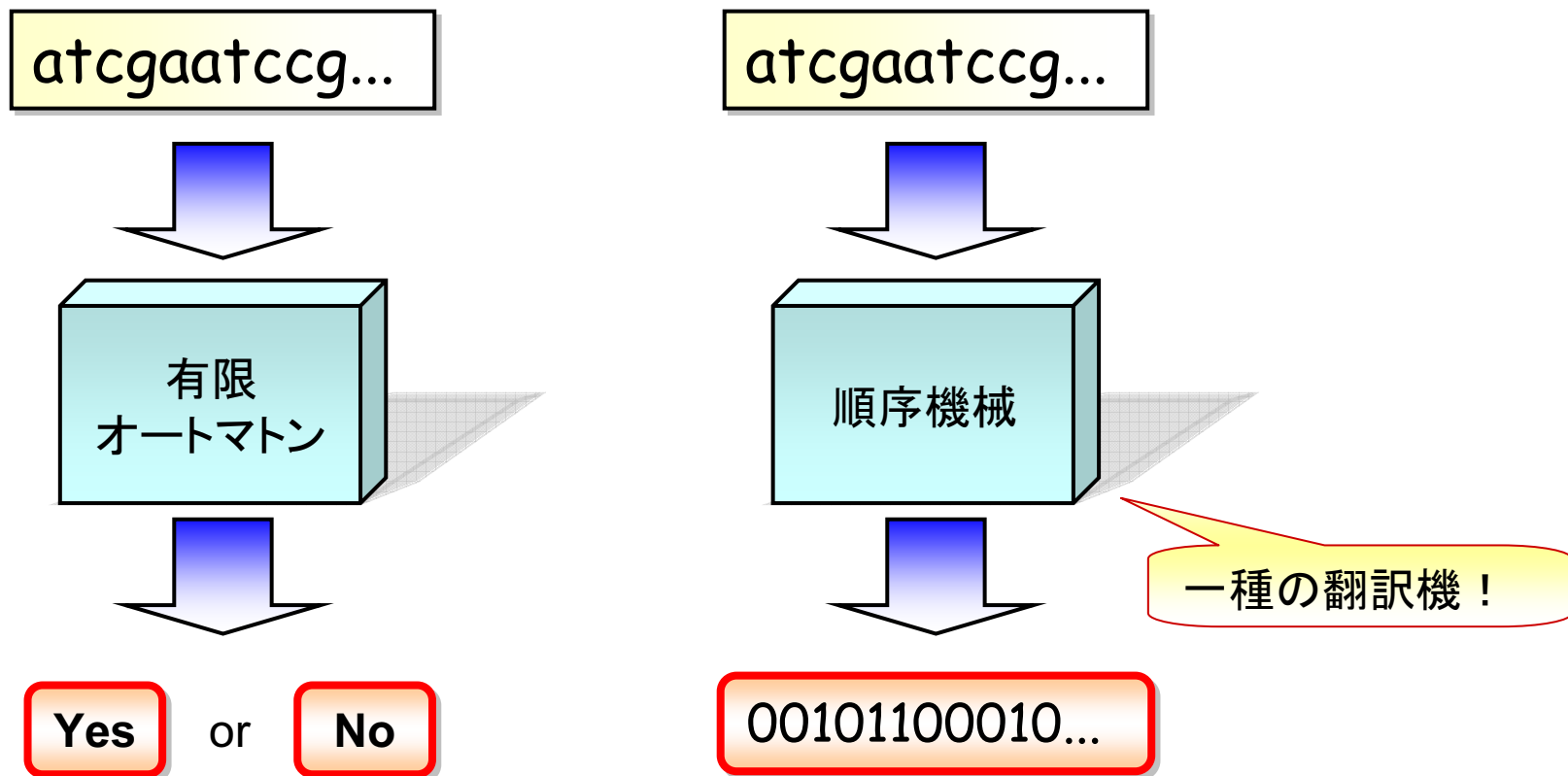
$$\begin{aligned} \text{例えば } abb \text{ に対しては } \delta(q_0, abb) &= \delta(q_0, bb) \\ &= \delta(q_0, b) \cup \delta(q_1, b) \\ &= \{q_0\} \cup \{q_1\} \cup \{q_2\} \\ &= \{q_0, q_1, q_2\} \end{aligned}$$

となり $q_2 \in F$ であるから $abb \in L(M)$ である。



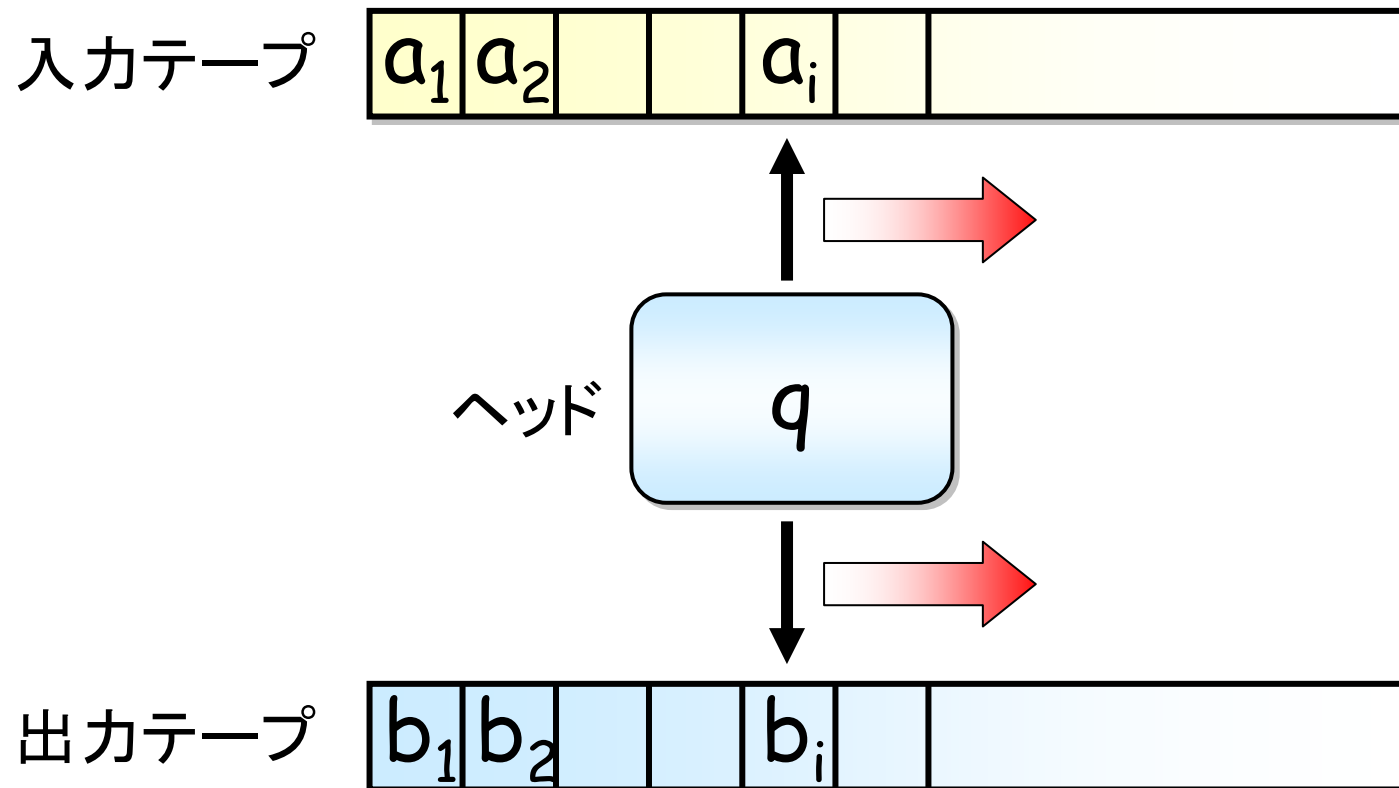
順序機械

順序機械とは...



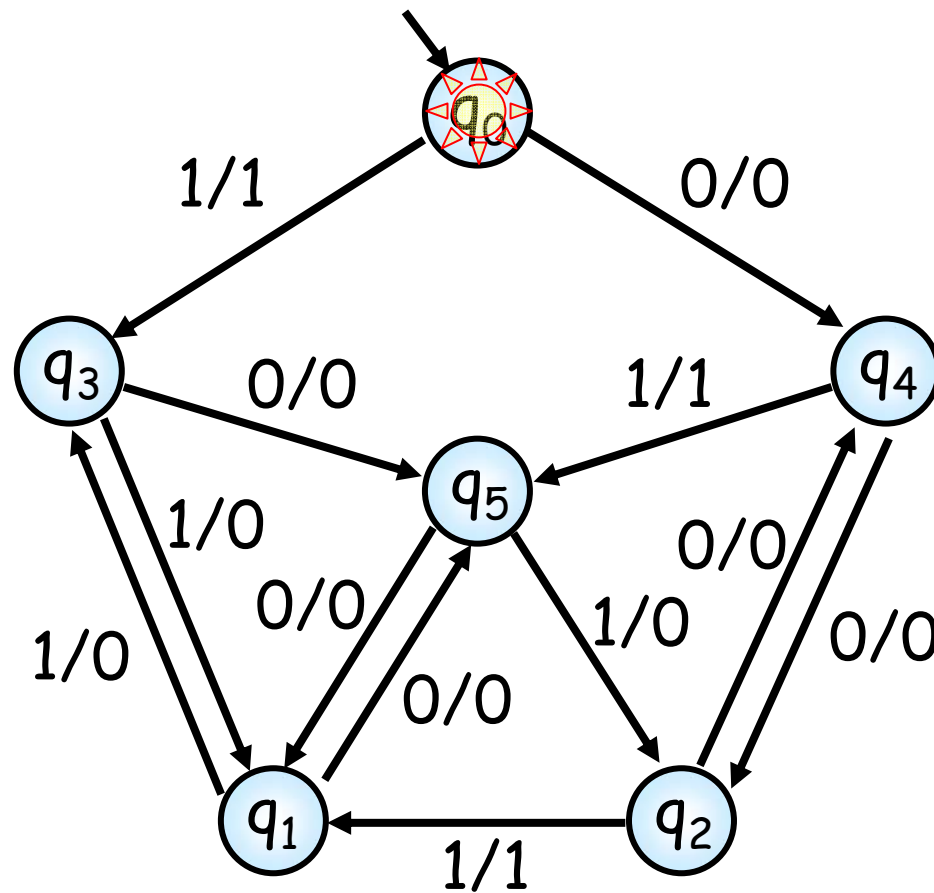


順序機械の概念図





順序機械の例



$$\lambda(q_0, 011)$$

$$= \lambda(q_0, 0) \lambda(\delta(q_0, 0), 11)$$

$$= 0 \lambda(q_4, 11)$$

$$= 0 \lambda(q_4, 1) \lambda(\delta(q_4, 1), 1)$$

$$= 01 \lambda(q_5, 1)$$

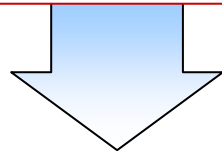
$$= 010$$



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

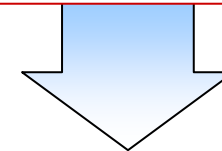
- 長所
 - ✓ 余計なデータ構造が不要
 - ✓ データの更新に対して柔軟
- 短所
 - ✓ 検索がおそい $O(n)$
 - ✓ スケーラビリティが低い



小規模文書群に対する検索向き
(例: UNIX の grep)

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, mg, Googleほか)

文字列照合ベースの
大規模全文検索DBシステムがある！



富士通 Interstage Shunsaku Data Manager

■ 特徴:

- XML形式のテキストをDBにみたてて、逐次検索(パターン照合)による高速なデータアクセスを実現する
 - インデックス不要
 - データ様式の構成を柔軟に変更可能。
- コアの部分は、九州大学の有川節夫教授(現、同大学副学長)と研究グループが開発した検索システム「SIGMA」をベースにしている

■ 導入事例:

- 国立遺伝学研究所 生命情報・DDBJセンター
 - 三大国際DNAデータバンクの一つ、DDBJ(日本DNAデータバンク)のARSA (All-round Retrieval of Sequence and Annotation) システム
- 富士通社内
 - 生産管理システム、新電子電話帳システム

理論的な中核は
Aho-Corasick アルゴリズム



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
- 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
- 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。