Extended string matching

光吉　健汰

June 11, 2021

北海道大学　情報知識ネットワーク研究室　M2

目次

出典

タイトル

Flexible Pattern Matching in Strings

著者

Gonzalo Navarro, Mathieu Raffinot

今回説明する章

4 章 Extended string matching

準備

定義

文字列に関する定義

文字列 文字集合 Σ の要素からなる列。

部分文字列 文字列 $T = T_1 T_2 \ldots T_n$ に対して、
$T_i \ldots T_j (1 \leq i \leq j \leq n)$ であるような文字列。

空文字列 0 個の文字からなる文字列。

部分文字列 文字列 $T = T_1 T_2 \ldots T_n$ に対して、
$T_i \ldots T_m (1 \leq i \leq n)$ であるような文字列。

文字列照合問題

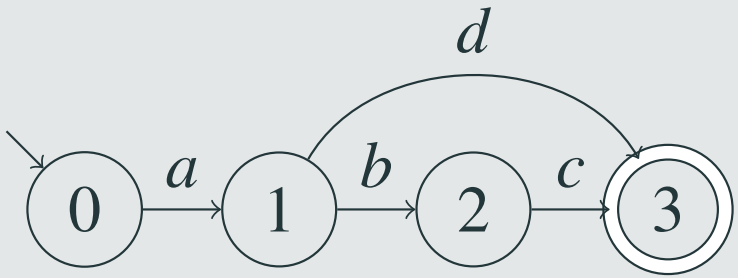
文字列 T、P が与えられたとき、P が T の部分文字列であるか？

有限オートマトン

非決定性有限オートマトン（Nondeterministic Finite Automaton; NFA）

以下の要素からなる有限オートマトンの一種

- 状態集合 Q
- 入力文字集合 Σ
- 遷移関数 $\delta : (\Sigma \cup \epsilon) \to 2^Q$
- 開始状態 $S \subset Q$
- 受理状態 $T \subset Q$



- 文字列を受け取ることで、 受理できるかを判定することができる。

拡張文字列

拡張文字列

文字集合 Σ と、以下の記号からなる文字列。
文字列集合の部分集合を表す。

- クラス文字（[a, b]）：文字の部分集合。
- 有界な文字の繰り返し（x(a, b)）：ある文字を a 回以上、b 回以下繰り返して得られる文字列。
- オプション文字（$x?$）：x を 0 個、または 1 個並べた文字列。
- 文字の繰り返し（$x*, x+$）：x を 0 個以上、または 1 個以上並べた文字列。

例 T = ab?c*de+f には、adef、abcdef、abccdeeeef
などが含まれる。

拡張文字列照合問題

拡張文字列

文字列 T と拡張文字列 P が与えられたとき、P の要素となるような T の部分文字列が存在するか？

例

T = acccdfabdeeef

P = ab?c*de+f

拡張文字列照合問題

拡張文字列

文字列 T と拡張文字列 P が与えられたとき、P の要素となるような T の部分文字列が存在するか？

例

T = acccdfabdeeef

P = ab?c*de+f

文字列照合アルゴリズム

文字列照合アルゴリズム

ビット並列テクニックを用いた、文字列照合アルゴリズムを紹介する。

- **Shift-And**
- **BNDM**

ビット並列テクニック

ビット並列テクニック

ワードマシンモデルにおいて、ビット演算の並列処理を利用して、処理の高速化を計るテクニック。

ワードマシンモデル

W ビットのビット演算について、定数時間で計算可能であるような計算機モデル。

例 $A = (6, 1, 5, 2), B = (2, 4, 5, 0)$ のとき、 $A + B$ の計算

A	0	1	1	0	0	0	0	1	0	1	0	1	0	0	1	0
$+ \quad B$	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0
$A + B$	1	0	0	0	0	1	0	1	1	0	1	0	0	0	1	0

Shift-And：アルゴリズム

アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa

$B[a]$	1	1	0
$B[b]$	0	0	1
$B[c]$	0	0	0

Shift-And：アルゴリズム

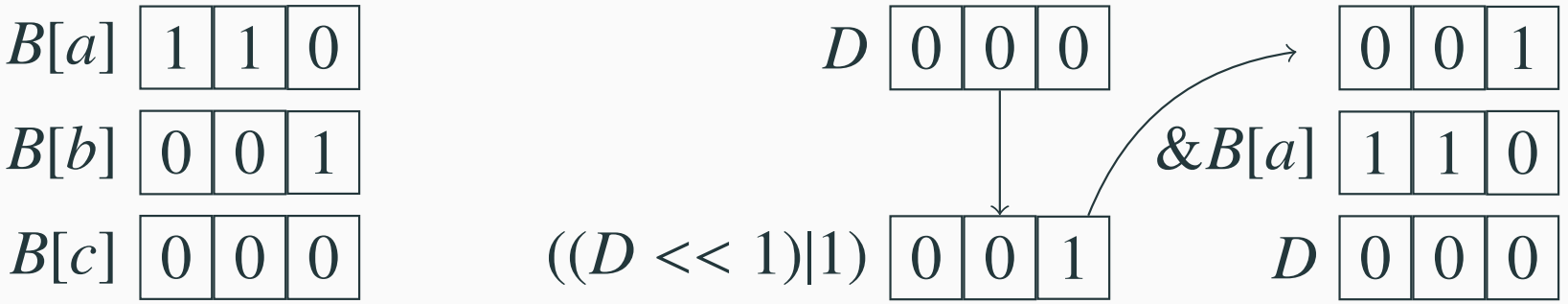
アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa



Shift-And：アルゴリズム

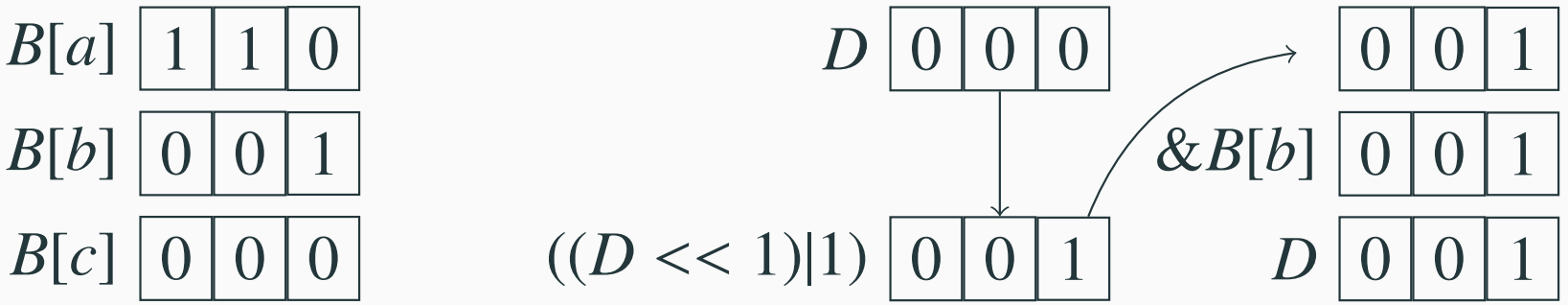
アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa



Shift-And：アルゴリズム

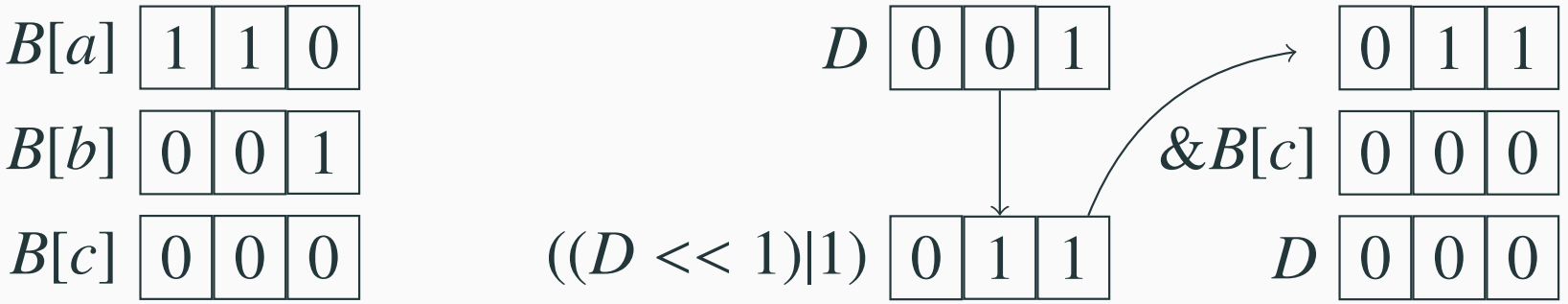
アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa



Shift-And：アルゴリズム

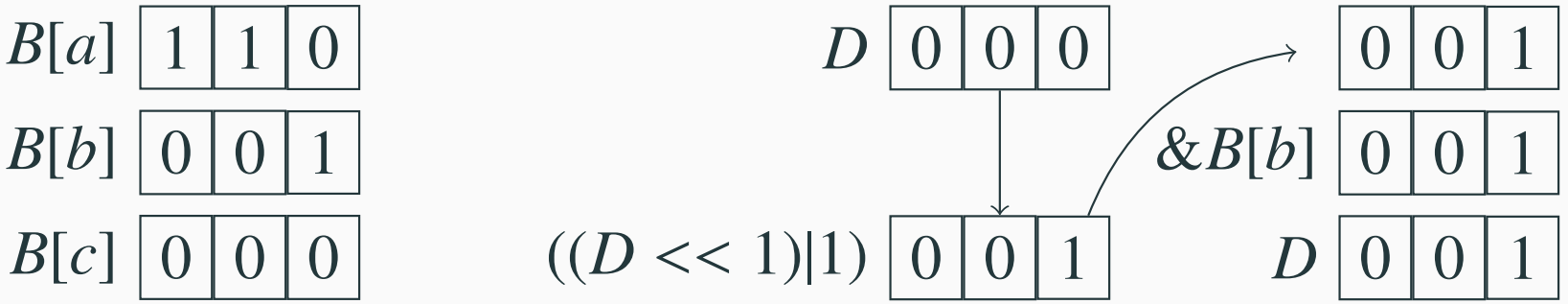
アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa



Shift-And：アルゴリズム

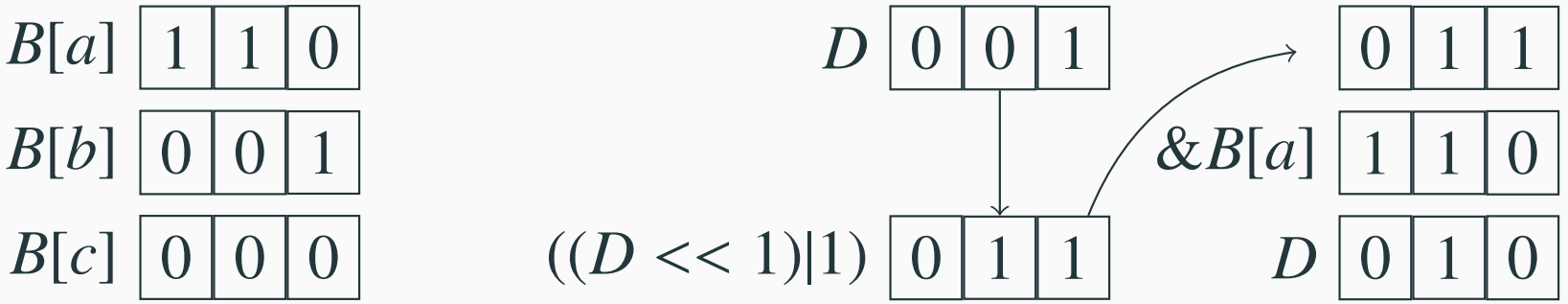
アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa



Shift-And：アルゴリズム

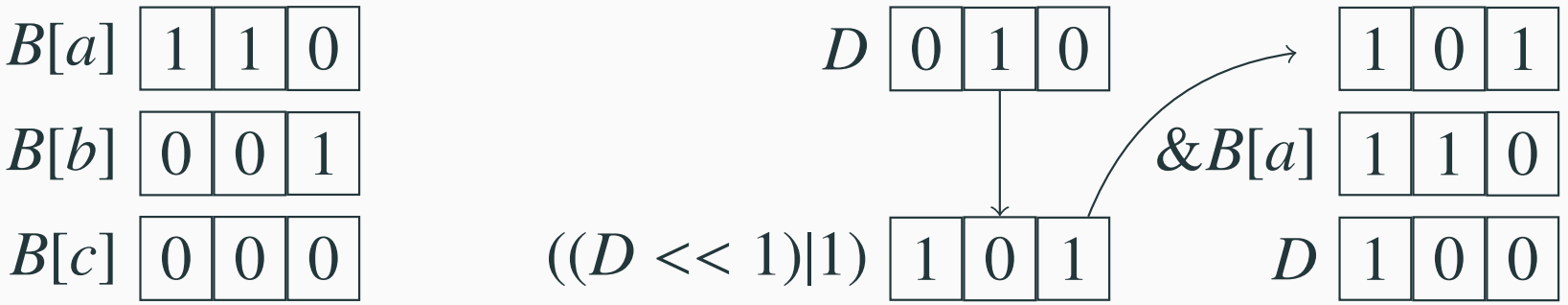
アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa



Shift-And：アルゴリズム

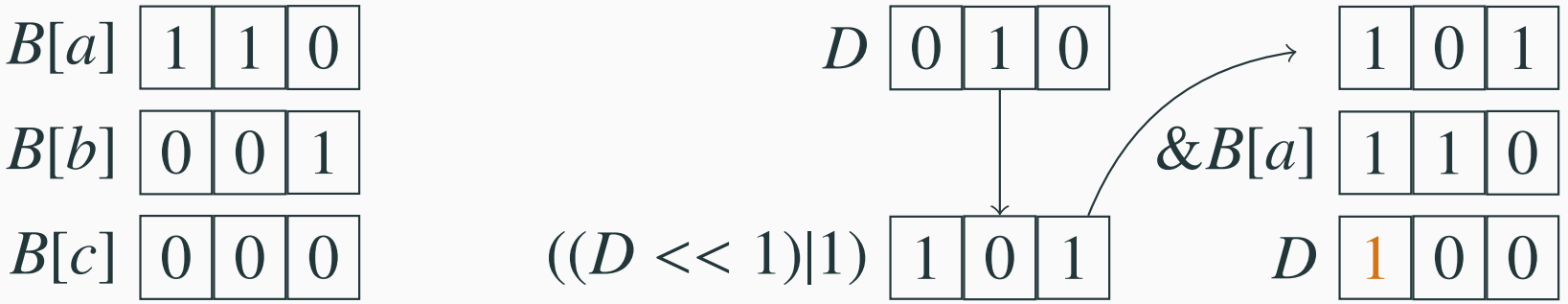
アルゴリズム

- パターンに対応した NFA の遷移の模倣を、
  ビット並列によって行う。
- パターン P から $|P-1|$ ビット長のビットマスク B を $|\Sigma|$ 個構築
- テキスト T から 1 文字ずつ受け取り、
  ビットマスク D を以下の式で更新

$$D \leftarrow ((D << 1)|1)\&B[T_i]$$

- D の最上位ビットが 1 であれば、テキストの位置を報告

例 T = abcbaa、 P = baa

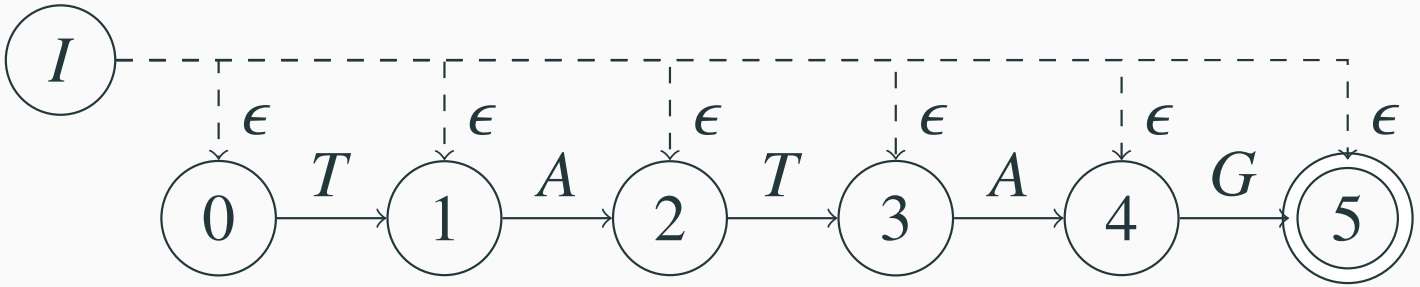


BNDM：基本アイデア

基本アイデア

- テキスト T 上でサイズ $|P|$ の窓をずらして照合していく。
- 窓内部では逆順に文字を読み込み照合する。
- 1 文字毎の照合は **Shift-And** と同様。
- 窓をずらすときに、一致しない箇所を飛ばす。
  - パターンを逆順にした場合の接尾辞を持つことで、接尾辞と照合した箇所まで飛ばすことができる。
  - 接尾辞集合を NFA で持つ。

例 P = GATAT

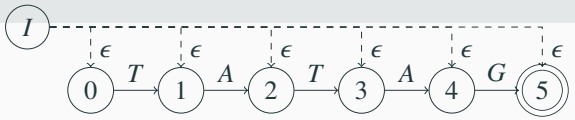


BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = ACGATATATAC、P = GATAT



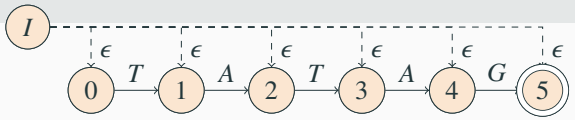
$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 $T =$ AGATA CGATATATAC、$P =$ GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

D

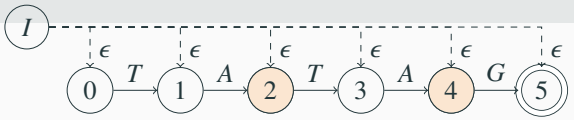
1	1	1	1	1

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 $T =$ AGATACGATATATAC、$P =$ GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

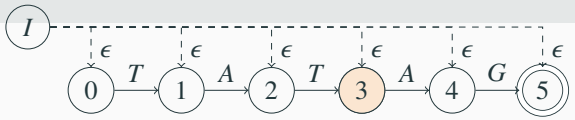
D	1	1	1	1	1
$\&B[A]$	0	1	0	1	0
D	0	1	0	1	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 $T =$ AGATA CGATATATAC、$P =$ GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

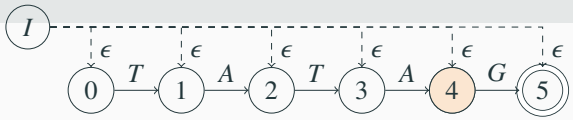
D	1	0	1	0	0
$\&B[T]$	0	0	1	0	1
D	0	0	1	0	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 $T =$ AGATA CGATATATAC、$P =$ GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

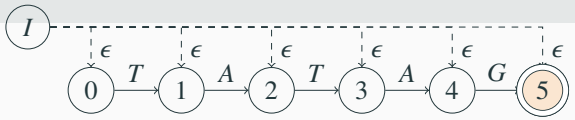
D	0	1	0	0	0
$\&B[A]$	0	1	0	1	0
D	0	1	0	0	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 $T =$ AGATA CGATATATAC、$P =$ GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

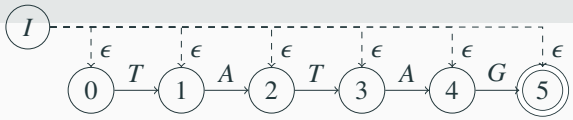
D	1	0	0	0	0
$\&B[G]$	1	0	0	0	0
D	1	0	0	0	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 $T =$ AGATA CGATATATAC、$P =$ GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

D

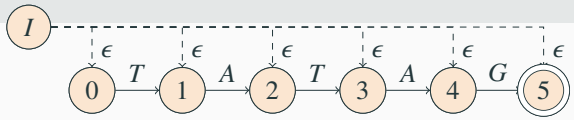
0	0	0	0	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = A GATAC GATATATAC、P = GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

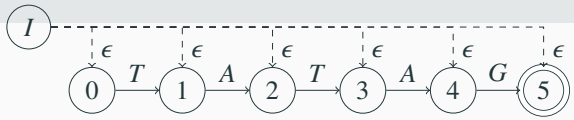
D

1	1	1	1	1

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。



例 T = A GATAC GATATATAC、P = GATAT

$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

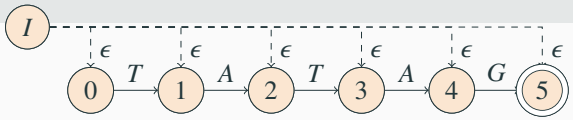
D	1	1	1	1	1
$\&B[C]$	0	0	0	0	0
D	0	0	0	0	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = AGATAC GATAT ATAC、P = GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

D

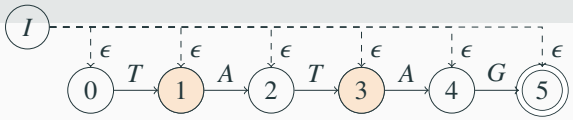
1	1	1	1	1

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = AGATAC GATAT ATAC、P = GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

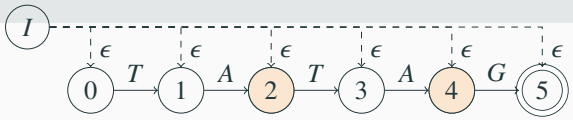
D	1	1	1	1	1
$\&B[T]$	0	0	1	0	1
D	0	0	1	0	1

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = AGATAC GATAT ATAC、P = GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

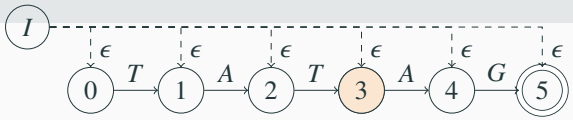
D	0	1	0	1	0
$\&B[A]$	0	1	0	1	0
D	0	1	0	1	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = AGATAC GATAT ATAC、P = GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

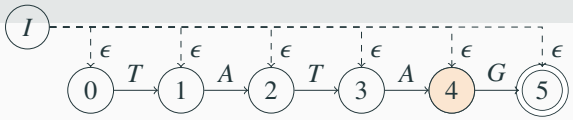
D	1	0	1	0	0
$\&B[T]$	0	0	1	0	1
D	0	0	1	0	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = AGATAC GATAT ATAC、P = GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

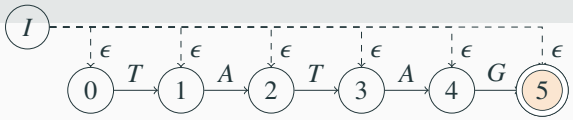
D	0	1	0	0	0
$\&B[A]$	0	1	0	1	0
D	0	1	0	0	0

BNDM：アルゴリズム

アルゴリズム

- NFA の状態集合をビット列で持つ。
- 初めにビット列 D をすべて 1 で初期化する。
- 窓の逆順で 1 文字ずつ受け取り、$D \leftarrow D\&B[T_i]$ で更新。
- 終了状態が 1 のとき、
  - 窓を走査し終えていなければ、窓の移動する箇所として保存。
  - 走査し終えていれば、パターンの出現位置として報告。

例 T = AGATAC GATAT ATAC、P = GATAT



$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

D	1	0	0	0	0
$\&B[G]$	1	0	0	0	0
D	1	0	0	0	0

拡張文字列照合への適用

拡張文字列照合への適用

前述のアルゴリズムを拡張文字列照合に適用した、以下のアルゴリズムを紹介する。

- **Extended-Shift-And**
- **Extended-BNDM**

方針

拡張文字列の各記号について、問題を分けて適用する。

- 文字クラス
- 有界な文字の繰り返し
- オプション文字
- 文字の繰り返し

文字クラス

文字クラス

文字集合の部分集合を表す記号

例 [a-d] = {a, b, c, d}

文字列照合問題への適用

- パターンに文字クラスが含まれる
- テキストに文字クラスが含まれる

パターンに文字クラスが含まれる場合

基本アイデア

文字クラス中の文字に対応するビットマスクのビットを立てる

例 $P = [\mathrm{C}, \mathrm{G}]\mathrm{ATAT}$

- **Shift-And** アルゴリズム

$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	1
$B[G]$	0	0	0	0	1
$B[T]$	1	0	1	0	0

- **BNDM** アルゴリズム

$B[A]$	0	1	0	1	0
$B[C]$	1	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1

テキストに文字クラスが含まれる場合

基本アイデア

クラス中の文字のビットマスク全てのビットオアをとったビットマスクを用いる。

- **Shift-And** アルゴリズム

$B[A]$	0	1	0	1	0
$B[C]$	0	0	0	0	1
$B[G]$	0	0	0	0	1
$B[T]$	1	0	1	0	0
$B[\{A,T\}]$	1	1	1	1	0

- **BNDM** アルゴリズム

$B[A]$	0	1	0	1	0
$B[C]$	1	0	0	0	0
$B[G]$	1	0	0	0	0
$B[T]$	0	0	1	0	1
$B[\{A,T\}]$	0	1	1	1	1

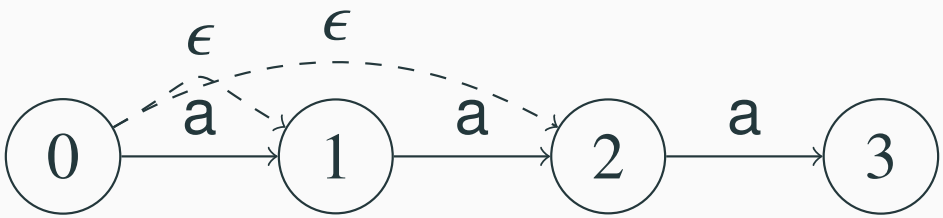
有界な文字の繰り返し：基本アイデア

- NFA を考える。
- b 個の状態を用意し、x で遷移する。
- これらのうち始端の状態は、$b - a$ 個の状態へ ϵ 遷移を持つ。
- ϵ 遷移を持つ状態は、アクティブかどうかを
  伝搬させなければならない。

ϵ 遷移

NFA 中で空文字を受け取る遷移

例 $a(1, 3)$



有界な文字の繰り返し：Shift-And への適用

有界な文字の繰り返しの場合、**Shift-And** と **BNDM** で行う改良は同じなので、**Shift-And** について考える。

状態を管理するビットマスク D について、$epsilon$ 遷移を考える。

この遷移は、$b - a$ ビットの区間に 1 を立てるような処理となる。

→ 引き算の繰り下がりによって区間の更新を行う。

例 35 – 8 の繰り下がりの様子

35	1	0	0	0	1	1
−8	0	0	1	0	0	0
27	0	1	1	0	1	1

有界な文字の繰り返し：Shift-And アルゴリズム

改良した Shift-And アルゴリズム

以下の更新式を加える

$$D \leftarrow D|((F - (D\&I))\& \sim F)$$

- I：繰り返しに対応する状態の始端を 1 としたビットマスク
- F：始端から $b - a + 1$ 個目の状態を 1 としたビットマスク

例 $P = AT(1,3)G(2,3)C$

I	0	0	0	0	1	0	0	1
F	0	0	1	0	1	0	0	0

D	0	0	0	0	0	0	0	1
$\&I$	0	0	0	0	0	0	0	1
$D\&I$	0	0	0	0	0	0	0	1

有界な文字の繰り返し：Shift-And アルゴリズム

改良した Shift-And アルゴリズム

以下の更新式を加える

$$D \leftarrow D|((F - (D\&I))\& \sim F)$$

- I：繰り返しに対応する状態の始端を 1 としたビットマスク
- F：始端から $b - a + 1$ 個目の状態を 1 としたビットマスク

例 $P = AT(1,3)G(2,3)C$

I	0	0	0	0	1	0	0	1
F	0	0	1	0	1	0	0	0

F	0	0	1	0	1	0	0	0
$-(D\&I)$	0	0	0	0	0	0	0	1
$F - (D\&I)$	0	0	1	0	0	1	1	1

有界な文字の繰り返し：Shift-And アルゴリズム

改良した Shift-And アルゴリズム

以下の更新式を加える

$$D \leftarrow D|((F - (D\&I))\& \sim F)$$

- I：繰り返しに対応する状態の始端を 1 としたビットマスク
- F：始端から $b - a + 1$ 個目の状態を 1 としたビットマスク

例 $P = AT(1,3)G(2,3)C$

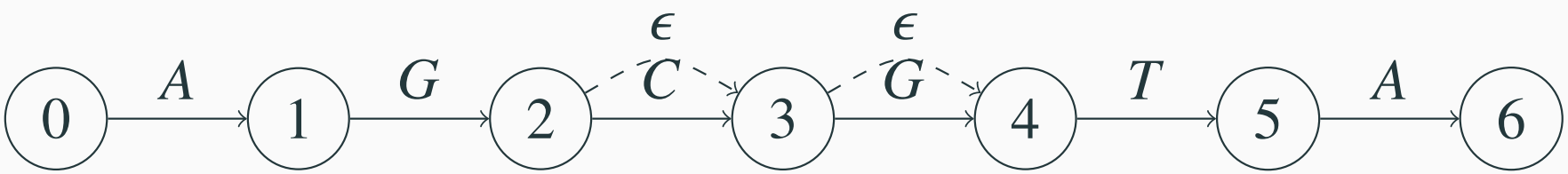
I	0	0	0	0	1	0	0	1
F	0	0	1	0	1	0	0	0

$F - (D\&I)$	0	0	1	0	0	1	1	1
$\& \sim F$	1	1	0	1	0	1	1	1
$(F - (D\&I))\& \sim F$	0	0	0	0	0	1	1	1

オプション文字

- ϵ 遷移を用いて、文字が存在しない場合でも
  遷移するように構築する。
- オプション文字が連続したとき、
  ϵ 遷移によって複数の状態に伝搬する。
- 有界な文字の繰り返しと同様に、
  引き算の繰り下がりによって伝搬する。

例 $P = AGC?G?TA$



オプション文字：Shift-And への適用

改良した Shift-And アルゴリズム

以下の更新式を加える

$$Df \leftarrow D|F \qquad D \leftarrow D|(A\&((\sim(Df - I)) \wedge Df))$$

- A：オプション文字がある状態を 1 としたビットマスク
- I：直後がオプション文字で、
  オプション文字でない状態を 1 としたビットマスク
- F：直後がオプション文字でなく、
  オプション文字である状態を 1 としたビットマスク

例 $P = AC?G?AT?CG$

A	0	0	0	1	0	1	1	0
I	0	0	0	0	1	0	0	1
F	0	0	0	1	0	1	0	0

D	0	0	0	0	0	0	0	1
$\mid F$	0	0	0	1	0	1	0	0
Df	0	0	0	1	0	1	0	1

オプション文字：Shift-And への適用

改良した Shift-And アルゴリズム

以下の更新式を加える

$$Df \leftarrow D|F \qquad D \leftarrow D|(A\&((\sim(Df - I)) \wedge Df))$$

- A：オプション文字がある状態を 1 としたビットマスク
- I：直後がオプション文字で、
  オプション文字でない状態を 1 としたビットマスク
- F：直後がオプション文字でなく、
  オプション文字である状態を 1 としたビットマスク

例 $P = AC?G?AT?CG$

A	0	0	0	1	0	1	1	0
I	0	0	0	0	1	0	0	1
F	0	0	0	1	0	1	0	0

Df	0	0	0	1	0	1	0	1
$-I$	0	0	0	0	1	0	0	1
$Df - I$	0	0	0	0	1	1	0	0

オプション文字：Shift-And への適用

改良した Shift-And アルゴリズム

以下の更新式を加える

$$Df \leftarrow D|F \qquad D \leftarrow D|(A\&((\sim(Df - I)) \wedge Df))$$

- A：オプション文字がある状態を 1 としたビットマスク
- I：直後がオプション文字で、
  オプション文字でない状態を 1 としたビットマスク
- F：直後がオプション文字でなく、
  オプション文字である状態を 1 としたビットマスク

例 $P = AC?G?AT?CG$

A	0	0	0	1	0	1	1	0
I	0	0	0	0	1	0	0	1
F	0	0	0	1	0	1	0	0

$\sim(Df - I)$	1	1	1	1	0	0	1	1
$\wedge Df$	0	0	0	1	0	1	0	1
$(\sim(Df - I)) \wedge Df$	1	1	1	0	0	1	1	0

オプション文字：Shift-And への適用

改良した Shift-And アルゴリズム

以下の更新式を加える

$$Df \leftarrow D|F \qquad D \leftarrow D|(A\&((\sim(Df - I)) \wedge Df))$$

- A：オプション文字がある状態を 1 としたビットマスク
- I：直後がオプション文字で、
  オプション文字でない状態を 1 としたビットマスク
- F：直後がオプション文字でなく、
  オプション文字である状態を 1 としたビットマスク

例 $P = AC?G?AT?CG$

A	0	0	0	1	0	1	1	0
I	0	0	0	0	1	0	0	1
F	0	0	0	1	0	1	0	0

A	0	0	0	1	0	1	1	0
$\&(\sim(Df - I)) \wedge Df$	0	1	1	0	0	1	1	0
$A\&((\sim(Df - I)) \wedge Df)$	0	0	0	0	0	1	1	0

文字の繰り返し

基本アイデア

繰り返しに対応するビットマスクを利用する

$D \leftarrow ((D << 1)|1)\&B[T_i])|(D\&S[T_i])$

例 $P =$ AT*AT

- **Shift-And** アルゴリズム

$S[A]$	0	0	0	0
$S[C]$	0	0	0	0
$S[G]$	0	0	0	0
$S[T]$	1	0	1	0

- **BNDM** アルゴリズム

$S[A]$	0	0	0	0
$S[C]$	0	0	0	0
$S[G]$	0	0	0	0
$S[T]$	0	1	0	0

Extended-Shift-And：最終的なアルゴリズム

アルゴリズム

一文字ずつ読み込む毎に以下の処理を行う。

$$D \leftarrow ((D << 1)|1)\&B[T_i]|D\&S[T_i])$$
$$Df \leftarrow D|F$$
$$D \leftarrow D|(A\&((\sim (Df - I)) \wedge Df))$$

有界な文字の繰り返しは、オプション文字によって表現できる。

→ 前処理でパターンを変換し、有界な文字の繰り返しの処理を省く。

時間計算量は、$O(|T|\lceil \frac{|P|}{W} \rceil)$ である。

Extended-BNDM：最終的なアルゴリズム

アルゴリズム

一文字ずつ読み込む毎に以下の処理を行う。

$$
\begin{aligned}
Df &\leftarrow D|F \\
D &\leftarrow D|(A\&((\sim (Df - I)) \wedge Df)) \\
D &\leftarrow ((D << 1)|1)\&B[T_i]|D\&S[T_i])
\end{aligned}
$$

有界な文字の繰り返しについては、**Extended-Shift-And** と同様。

ただし、テキスト中のパターンが窓の区間を超える場合があるので、窓を走査し終えて受理状態が 1 の場合は、その位置にパターンがあるかを確認する必要がある。
このため、最悪時間計算量は $O(|T|^2)$ となる。

まとめ

まとめ

- 以下のビット並列テクニックを用いた文字列照合アルゴリズムを紹介した。
  - **Shift-And**
  - **BNDM**
- これらを拡張した、以下のアルゴリズムを紹介した。
  - **Extended-Shift-And**
  - **Extended-BNDM**