Introduction to Suffix Automaton

JOI 夏季セミナー 2020　KoD

はじめに

- 高速文字列解析班の KoD です
- Suffix Automaton について紹介します！

はじめに

- Suffix Automaton って，何ができるの…?

はじめに

- Suffix Automaton って，何ができるの…?
  - 部分文字列の種類数を数える

はじめに

- Suffix Automaton って，何ができるの…?
  - 部分文字列の種類数を数える
  - 部分文字列を辞書順に並べたときの k 番目を求める

はじめに

- Suffix Automaton って，何ができるの…?
  - 部分文字列の種類数を数える
  - 部分文字列を辞書順に並べたときの k 番目を求める
  - 他にもたくさん（後ほど紹介）

はじめに

- Suffix Automaton って，何ができるの…?
  - 部分文字列の種類数を数える
  - 部分文字列を辞書順に並べたときの k 番目を求める
  - 他にもたくさん（後ほど紹介）

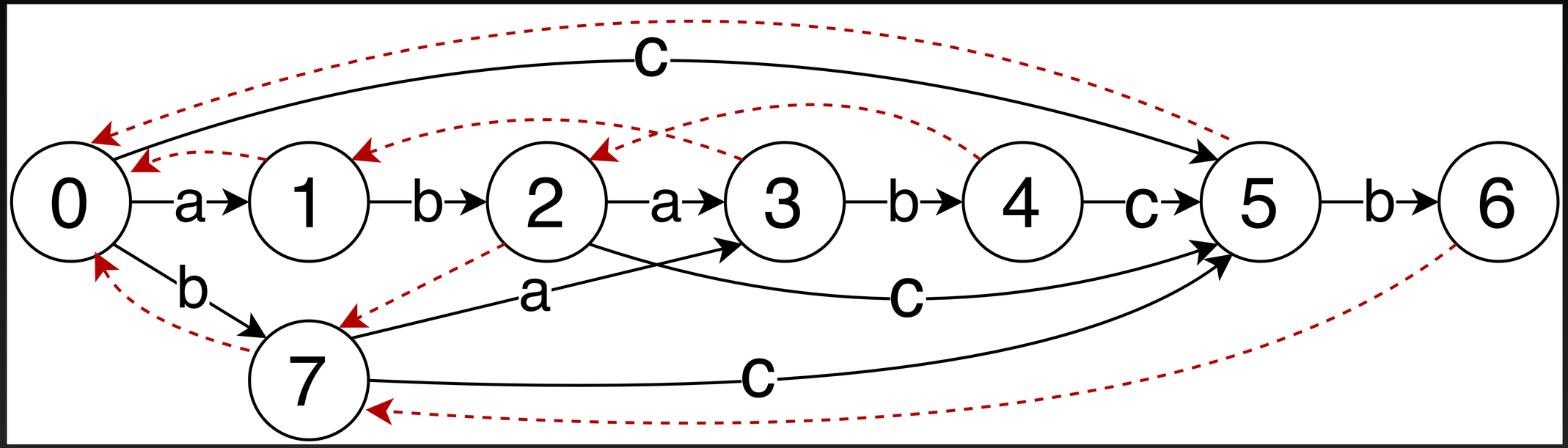
つよそう！（小並感）

目次

1. Suffix Automaton の性質
2. 具体的な構築方法
3. 計算量などの解析
4. 実際に使ってみよう！

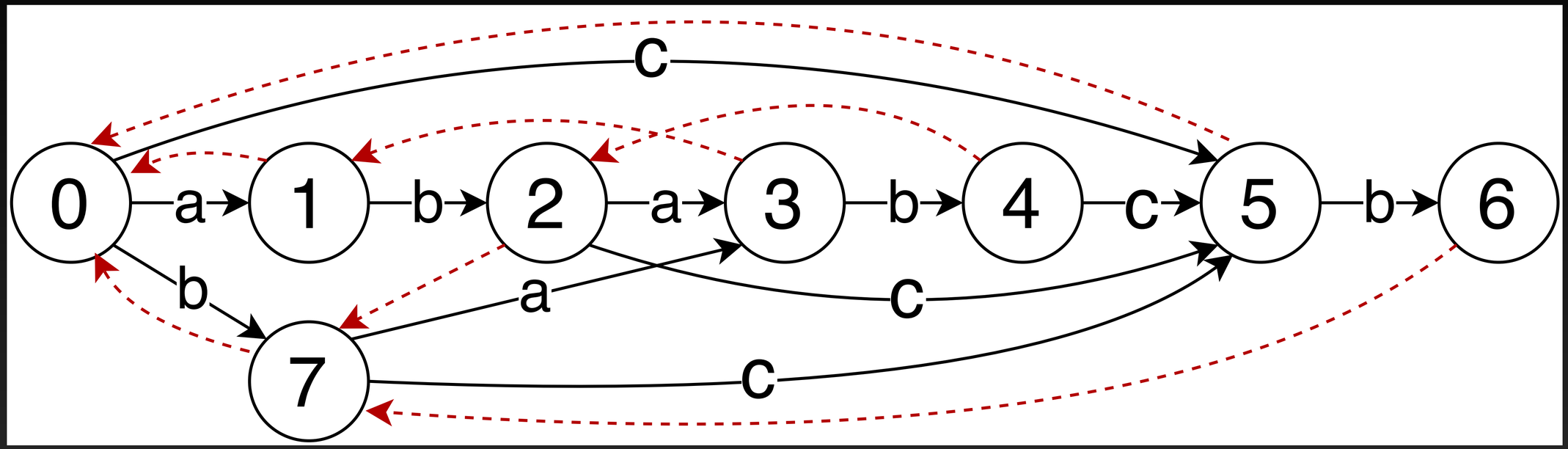
- ▶ 1. Suffix Automaton の性質
- 2. 具体的な構築方法
- 3. 計算量などの解析
- 4. 実際に使ってみよう！

Suffix Automaton の性質



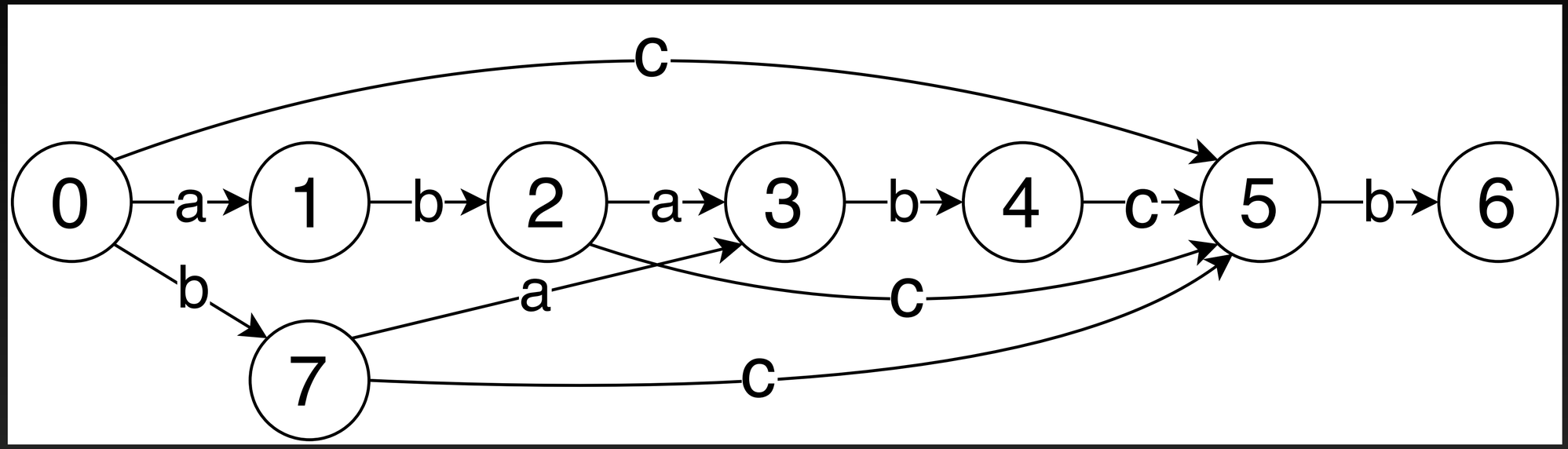
- 文字列 S（上の例では “ababcb”）
  の部分文字列を，漏れなく重複なく管理

Suffix Automaton の性質



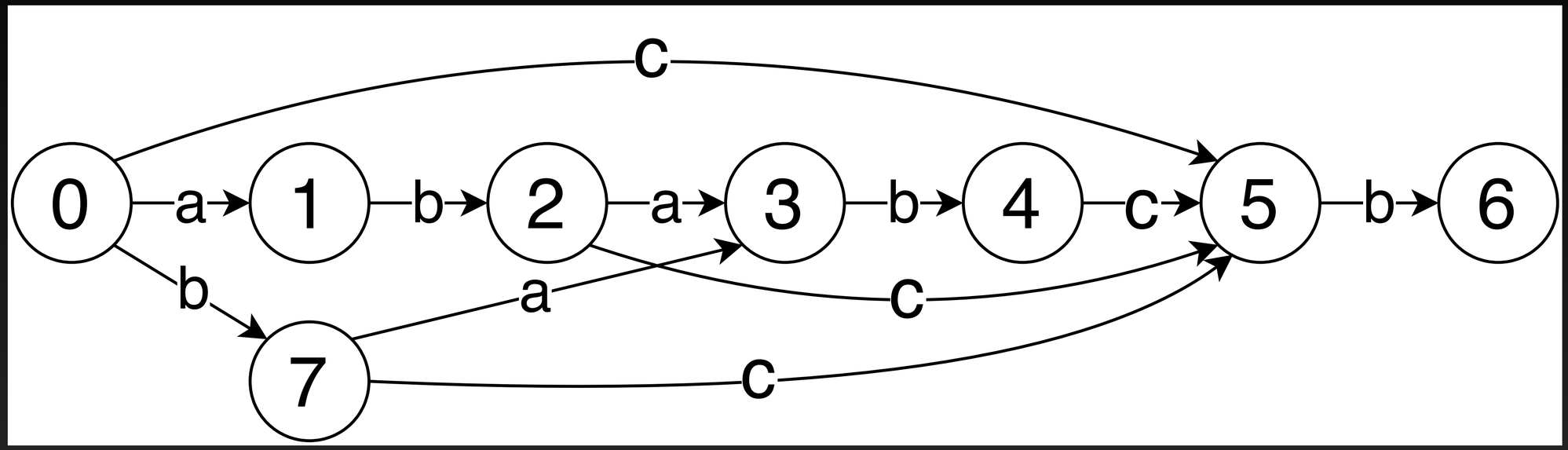
- 図が複雑すぎる
  -> 実線と破線を分離

Suffix Automaton の性質



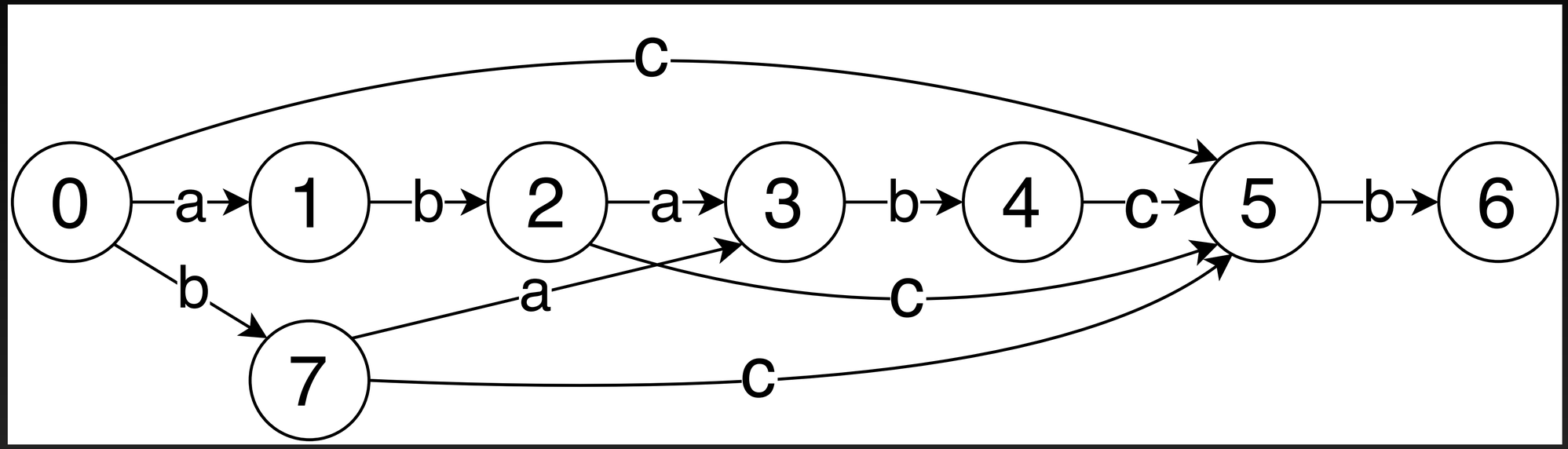
- 実線部分だけ取り出すと，DAG になっている

Suffix Automaton の性質



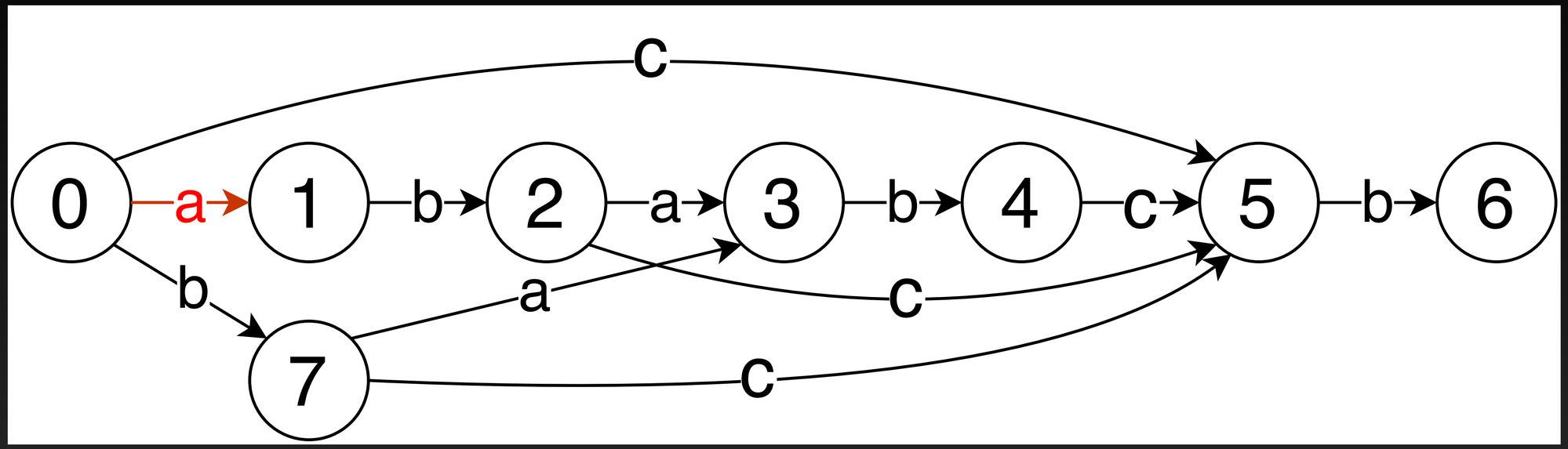
- 各ノードは，始点からのパスで表されるような部分文字列を管理

Suffix Automaton の性質



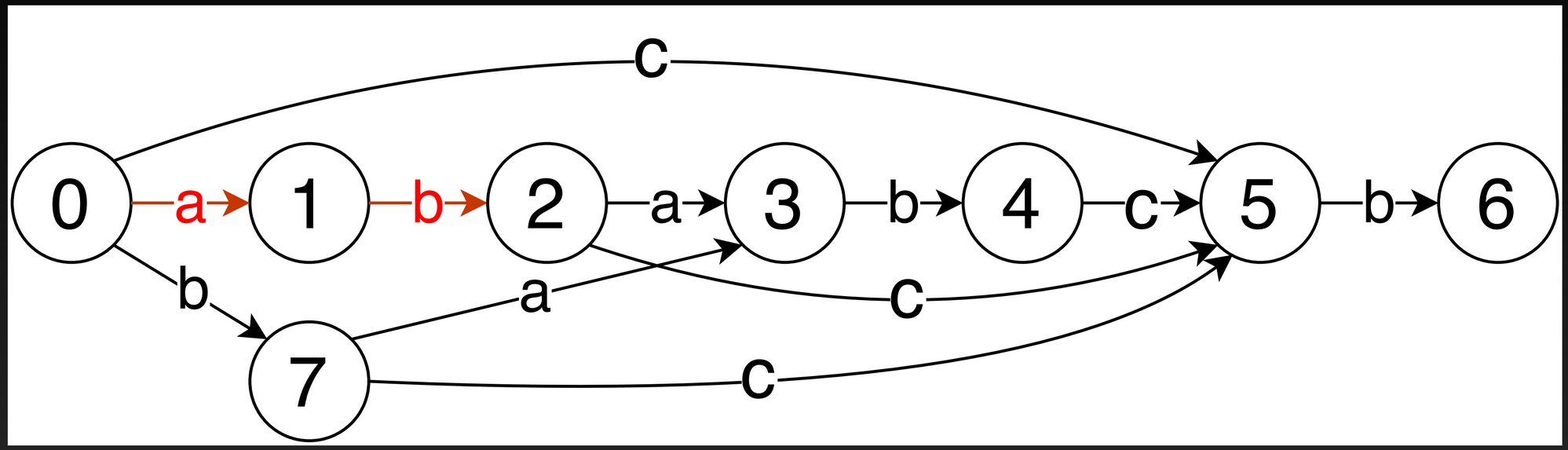
ノード	1	2	3	4	5	6	7
管理する 部分文字列							

Suffix Automaton の性質



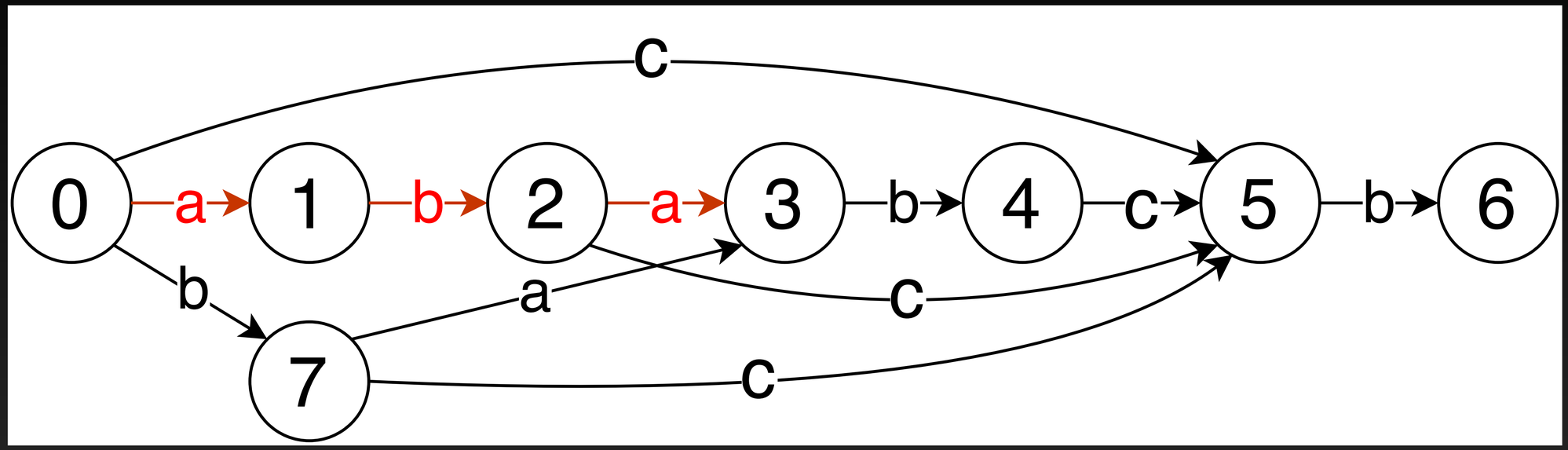
ノード	1	2	3	4	5	6	7
管理する 部分文字列	a						

Suffix Automaton の性質



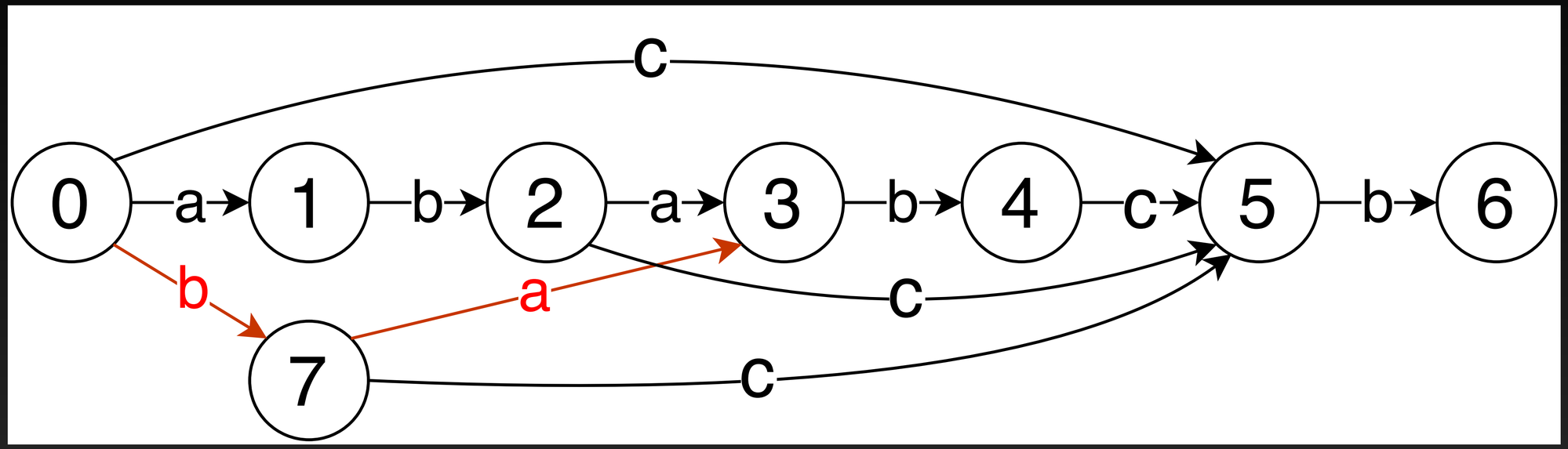
ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab					

Suffix Automaton の性質



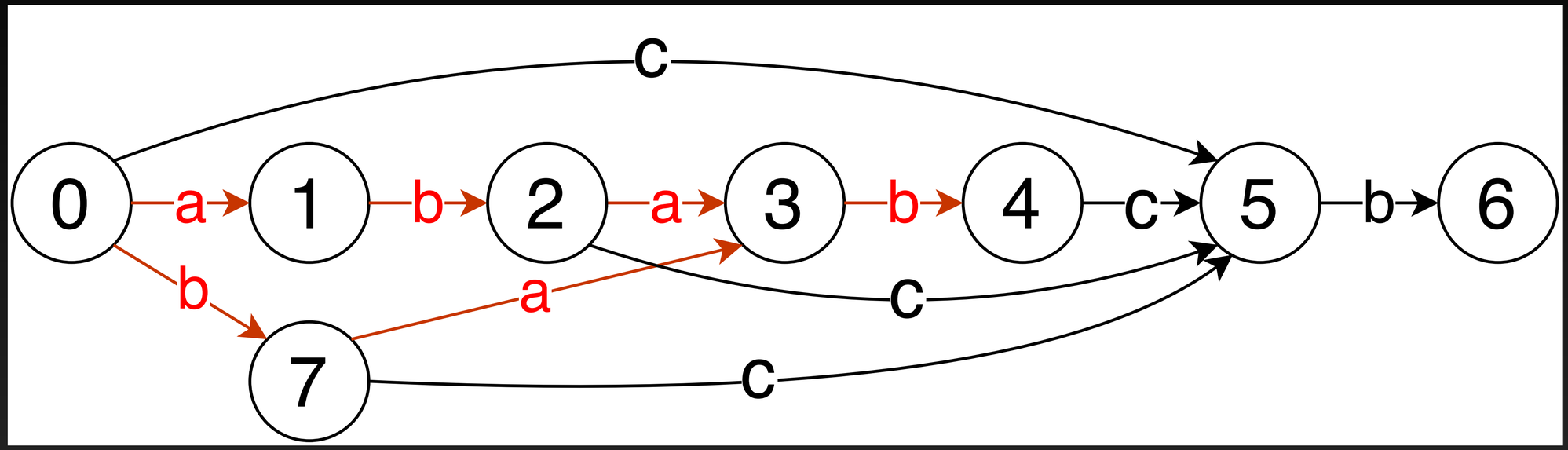
ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba				

Suffix Automaton の性質



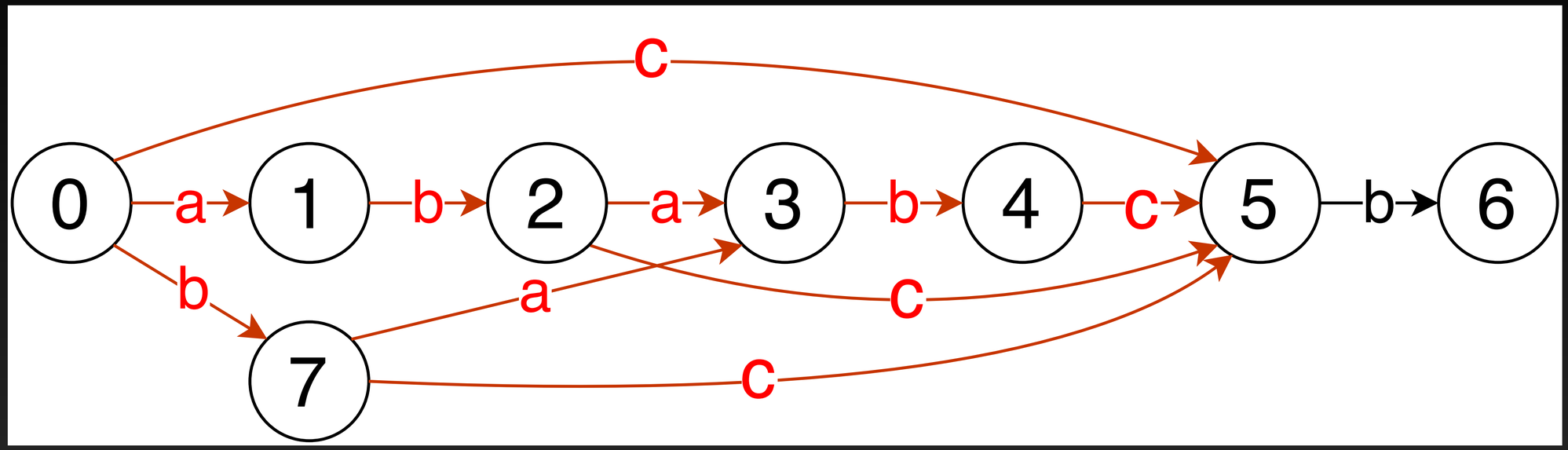
ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba				

Suffix Automaton の性質



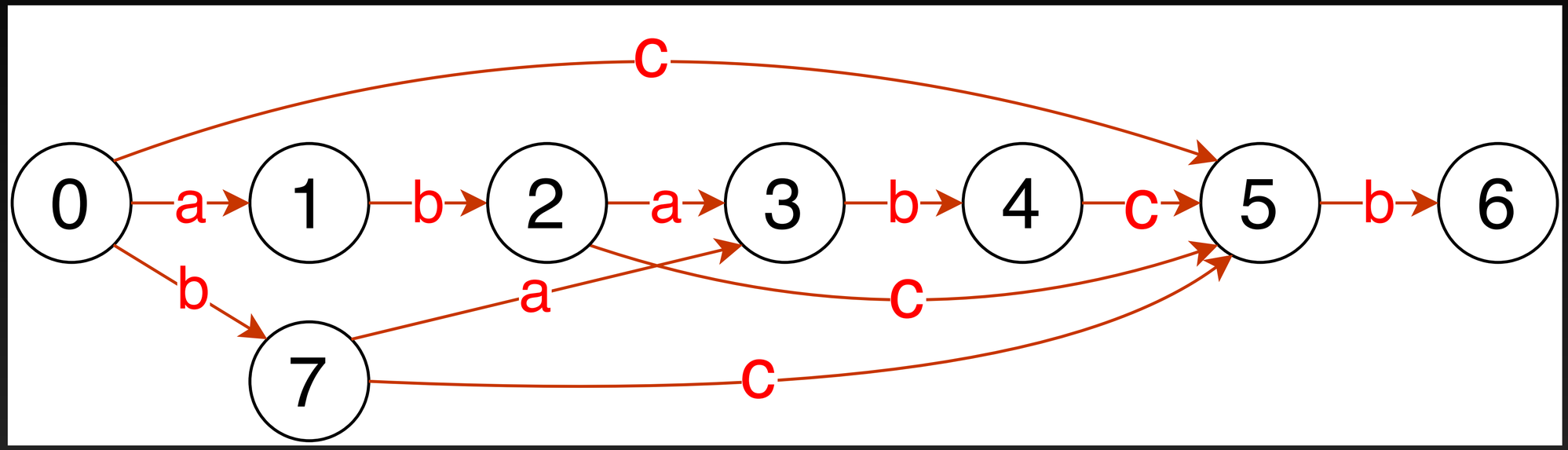
ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab			

Suffix Automaton の性質



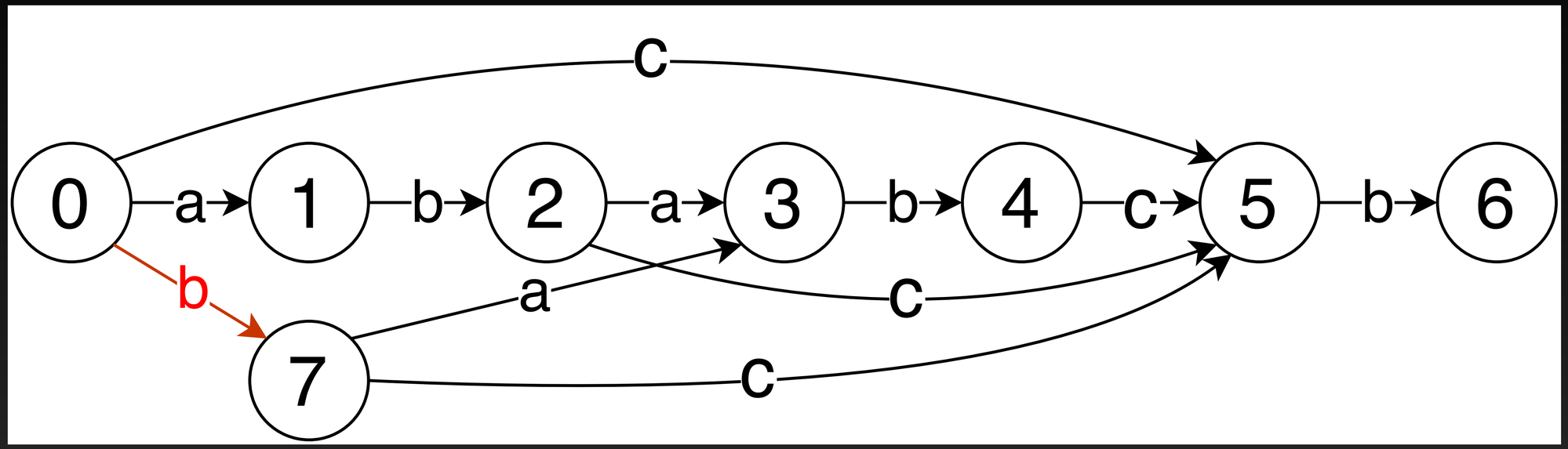
ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c		

Suffix Automaton の性質



ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	

Suffix Automaton の性質



ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- 重要な性質: 各ノードが管理する文字列は，順番に先頭を削っていったものになっている

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- つまり，ノード u が表す最長の文字列を T[u] で表すと，ノード u が管理する文字列は T[u] の接尾辞

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- 前から削っていき，途中で無くなる場合がある（ノード 4 など）

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab ab b	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- 前から削っていき，途中で無くなる場合がある（ノード 4 など）

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab ab b	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- この場合，“ab” はノード 2 が管理する最も長い文字列（すなわち，T[2]）

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab b	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- さらに，“b” はノード 7 が管理する最も長い文字列（すなわち，T[7]）

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- 重要な性質:
  T[u] の接尾辞のうち, u が管理していない
  最長のもの(存在すれば)を X[u] で表す

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- 重要な性質:

  例: X[4] = “ab”, X[6] = “b”

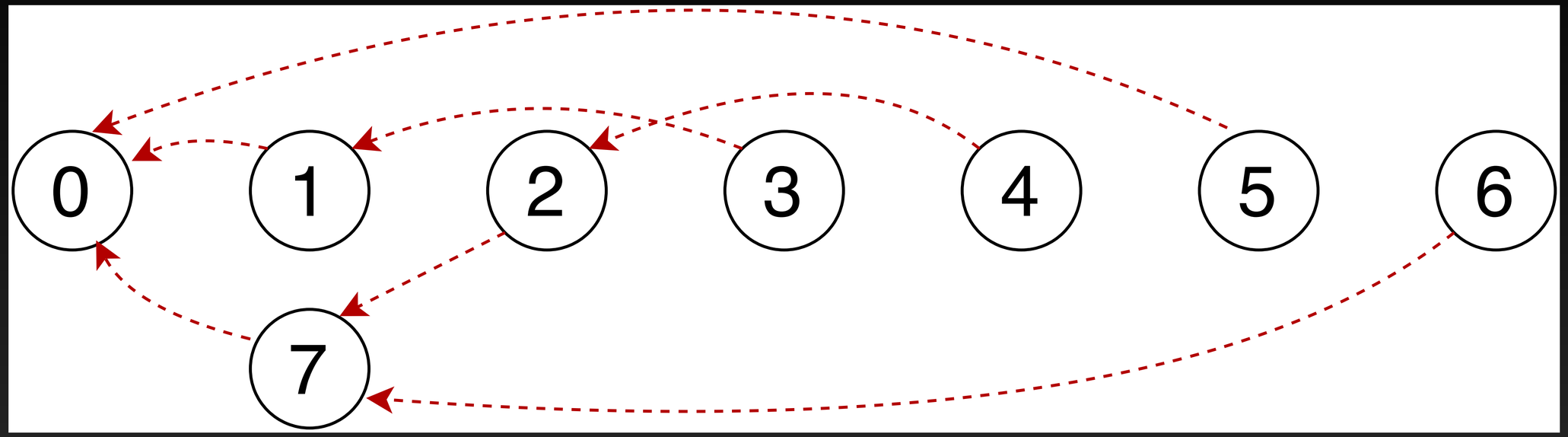
  X[u] = T[v] となる v がただ一つ存在する

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab b	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

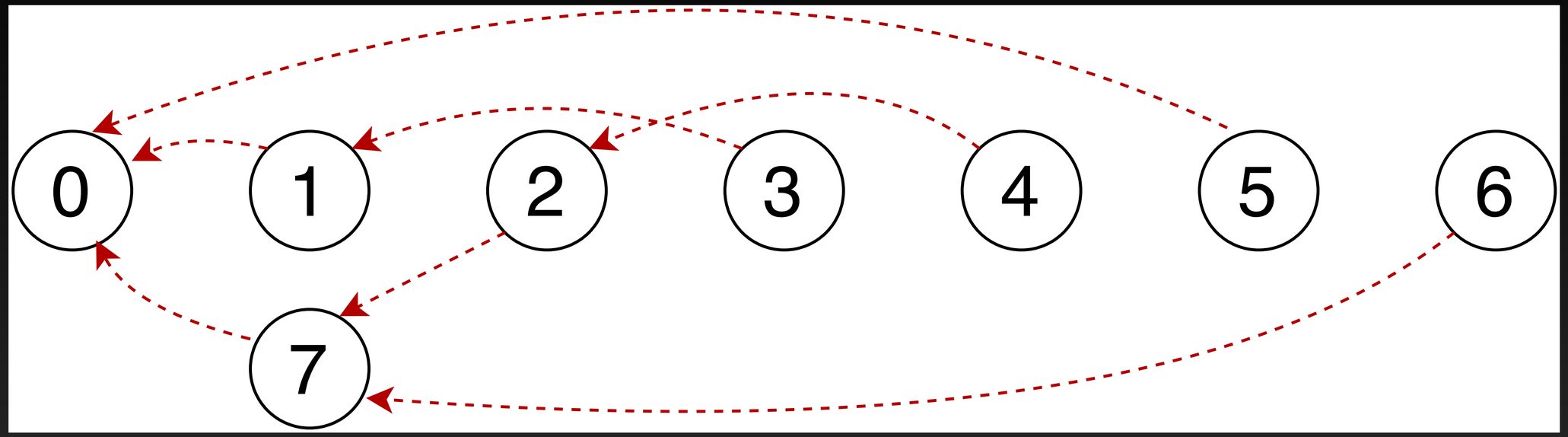
- このような u, v の関係を表現したい
  (例: 4 -> 2, 2 -> 7)

Suffix Automaton の性質



- 先ほど出てきた破線の部分
- Suffix Link という

Suffix Automaton の性質



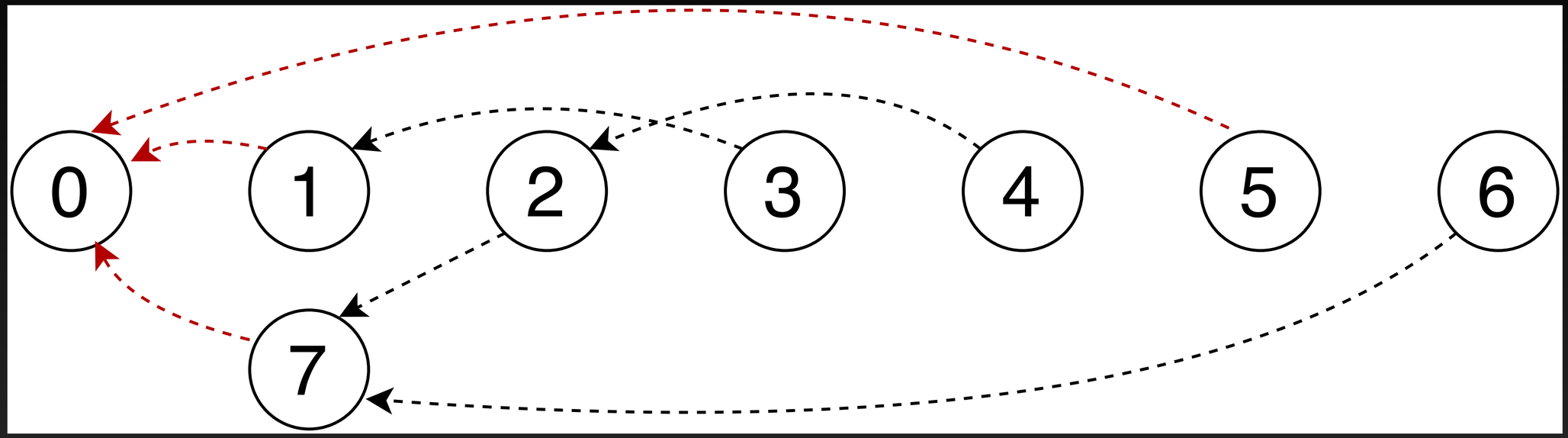
- 破線部分だけ取り出すと，ノード 0 を根とする根付き木になっている

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- u が T[u] の接尾辞全てを管理する場合:
  Suffix Link はノード 0 につながれる

Suffix Automaton の性質



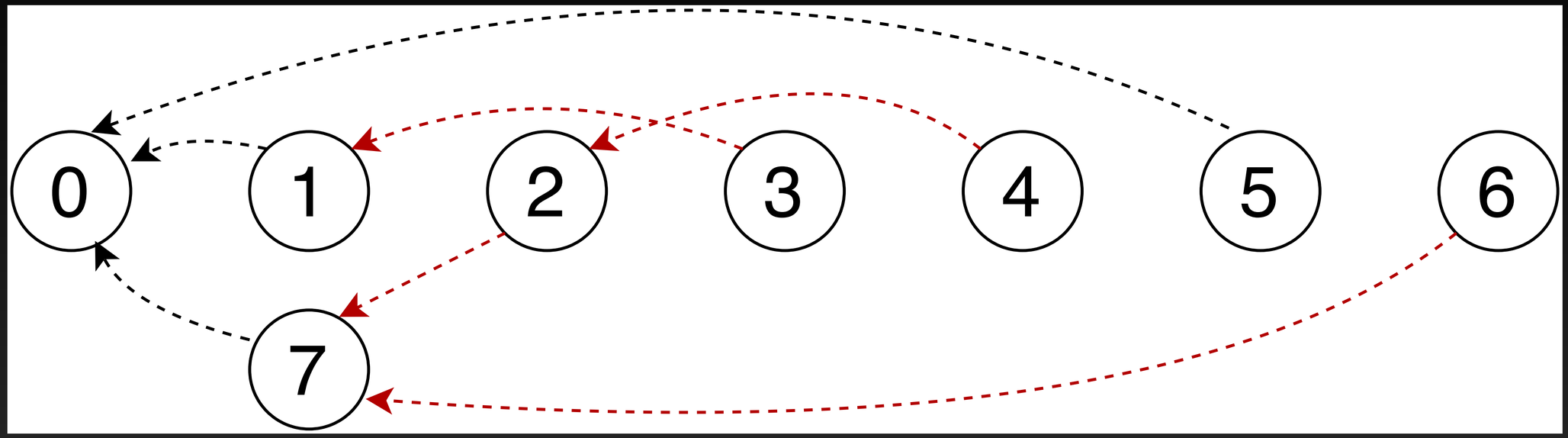
- u が T[u] の接尾辞全てを管理する場合:
  Suffix Link はノード 0 につながれる

Suffix Automaton の性質

ノード	1	2	3	4	5	6	7
管理する 部分文字列	a	ab	aba ba	abab bab	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- それ以外: X[u] = T[v] となる v につながれる

Suffix Automaton の性質



- それ以外: X[u] = T[v] となる v につながれる

Suffix Automaton の性質

- いろいろと複雑

- 構築する方法を通して理解しましょう

- ▶ 1. Suffix Automaton の性質
- 2. 具体的な構築方法
- 3. 計算量などの解析
- 4. 実際に使ってみよう！

1. Suffix Automaton の性質

▶2. 具体的な構築方法

3. 計算量などの解析

4. 実際に使ってみよう！

具体的な構築方法

- 大まかな方針:
  - 新たに文字 c を付け足すことを考える
  - 文字列 S に c を付け足すとき，新たに増える部分文字列は S の接尾辞に c を付け足したもの

具体的な構築方法

- 大まかな方針:
  - 最も深いノードを u とすると T[u] = S
  - 新たにノードを一つ作り, v とする
    v への辺を追加しながら Suffix Link を
    上っていく

具体的な構築方法

- 大まかな方針:
  - すでに c の辺を持つノード（w とおく）に着くことがある
  - つまり，“(S の接尾辞) + c” という形がすでに現れている場合

具体的な構築方法

- 大まかな方針:
  - これ以上辺を追加すると重複してしまう
  - v からの Suffix Link は，w から c の辺を辿った先のノードにつながれるべき

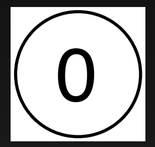
    -> Suffix Link の条件を満たしているのか？

具体的な構築方法

- 大まかな方針:
  - S = “abbb” に対応する Suffix Automaton を構築する様子を見てみる

具体的な構築方法

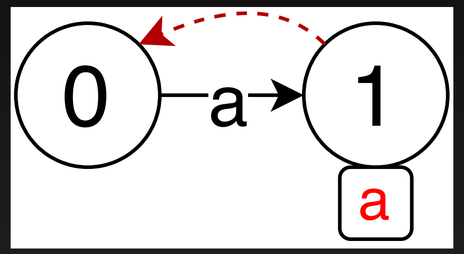
現在の文字列: “”



- はい
- 寂しそう

具体的な構築方法

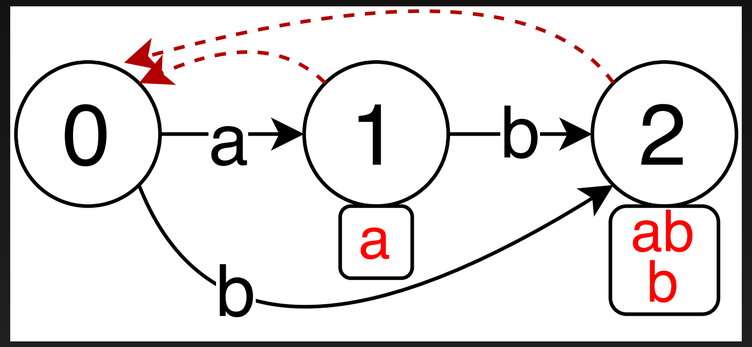
現在の文字列: “a”



- まだ簡単
- ノードのそばに管理される文字列を示す

具体的な構築方法

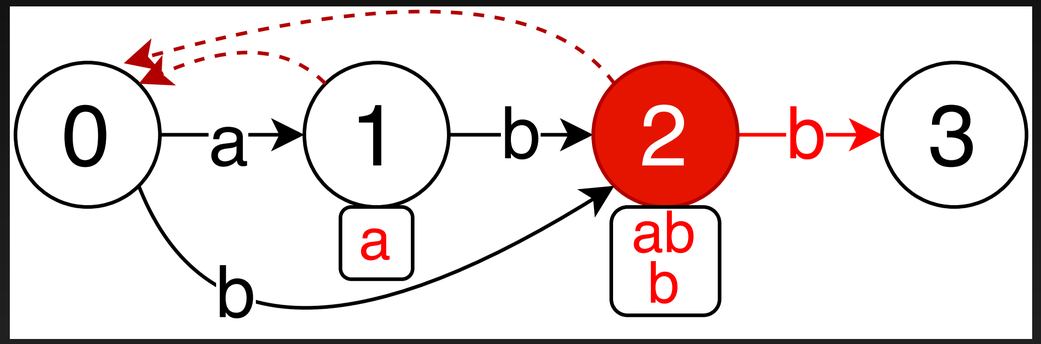
現在の文字列: “ab”



・ノード 1 から ‘b’ の辺を追加し，
Suffix Link を辿り，ノード 0 からも辺を追加

具体的な構築方法

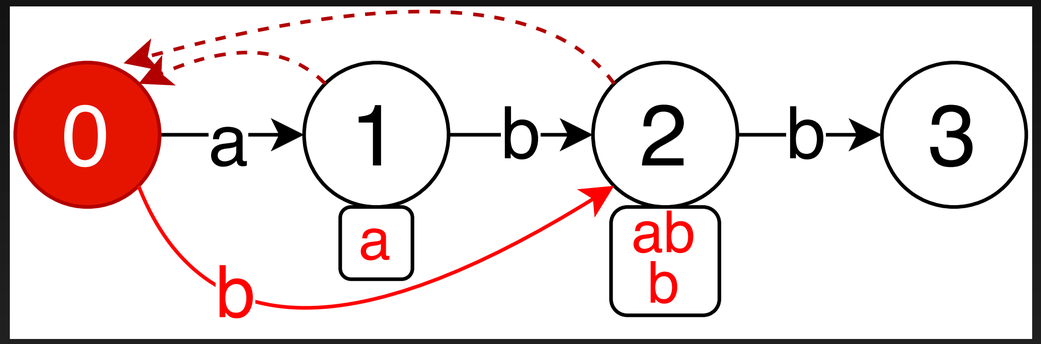
現在の文字列: “abb”



- ノード 2 から ‘b’ の辺を追加
- Suffix Link を辿ってノード 0 へ

具体的な構築方法

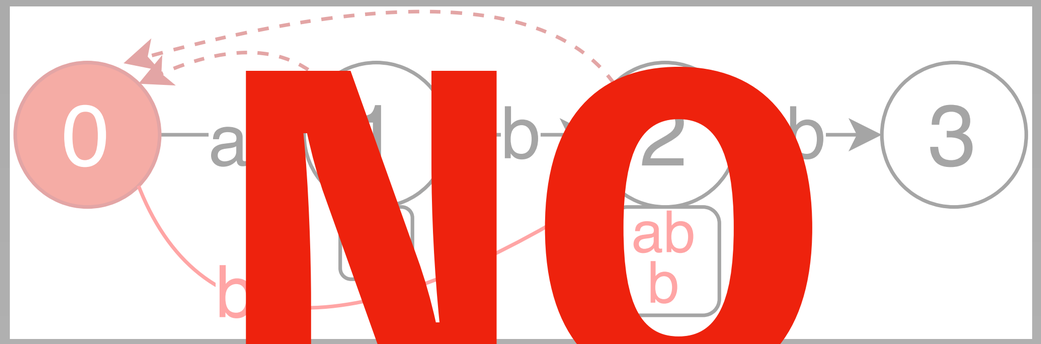
現在の文字列: “abb”



- ノード 0 は ‘b’ の辺をすでに持っている
- これ以上は上らない
  ノード 2 に Suffix Link をつないで終了…?

具体的な構築方法

現在の文字列: “abb”



NO

- ノード 0 は ‘b’ の辺をすでに持っている
- これ以上は上らない
  ノード 2 に Suffix Link をつないで終了…?

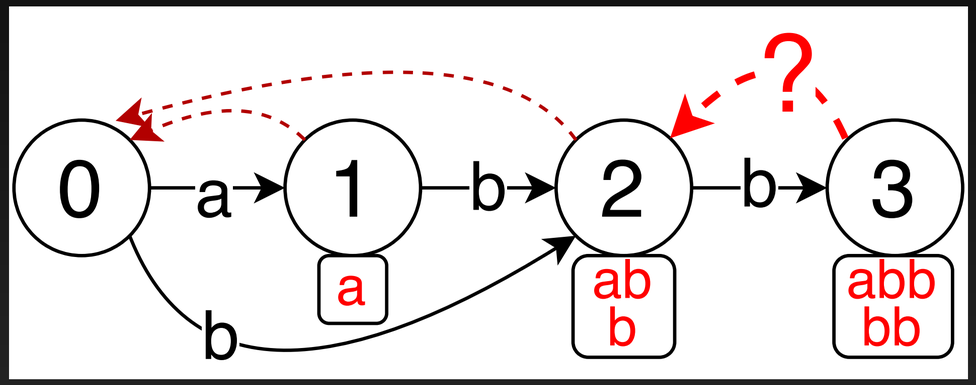
Suffix Automaton の性質（復習）

ノード	1	2	3	4	5	6	7
管理する部分文字列	a	ab	aba ba	abab bab ab b	ababc babc abc bc c	ababcb babcb abcb bob cb	b

- Suffix Link は X[u] = T[v] となる v につながれる

具体的な構築方法

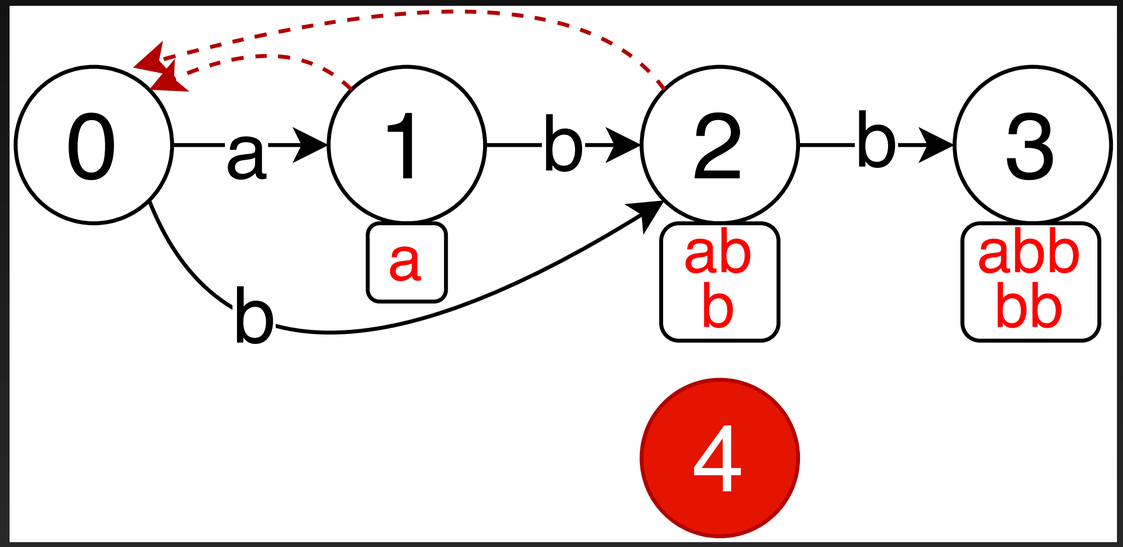
現在の文字列: “abb”



- ノード 3 から Suffix Link をつなぐ先は，
  T[v] = “b” となる必要がある！
- ノード 2 は “ab” も管理している

具体的な構築方法

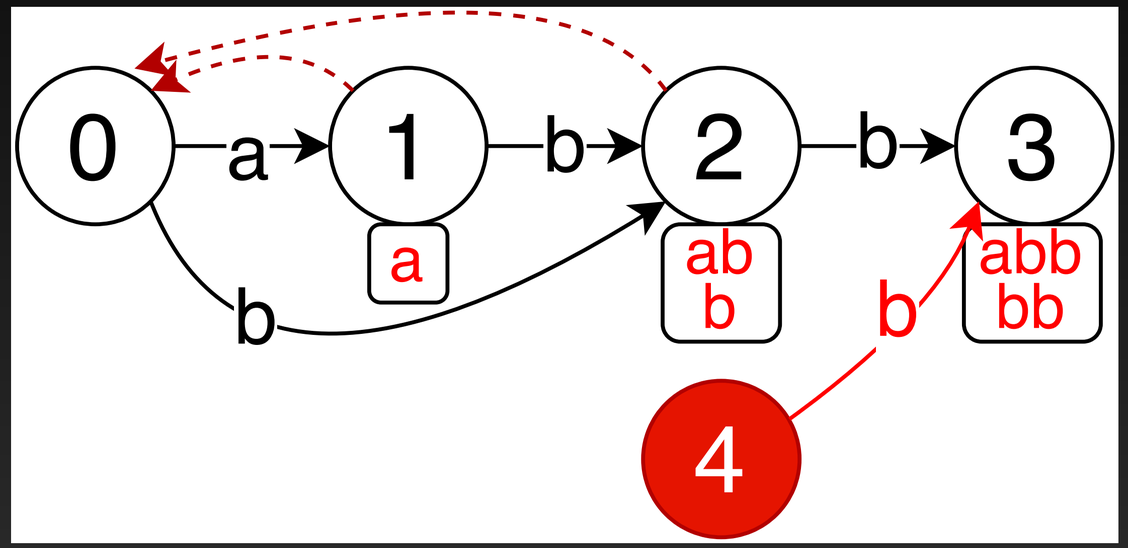
現在の文字列: “abb”



・ノード２を分離し，“b”の管理権限をノード２から新しいノードに移せばよい

具体的な構築方法

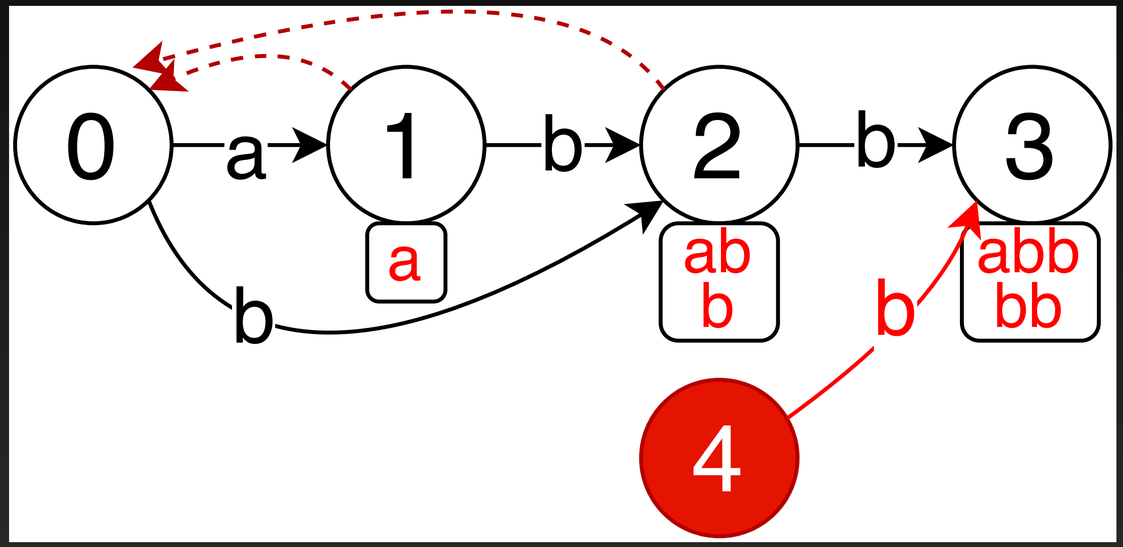
現在の文字列: “abb”



・ノード２から出ている辺をコピー

具体的な構築方法

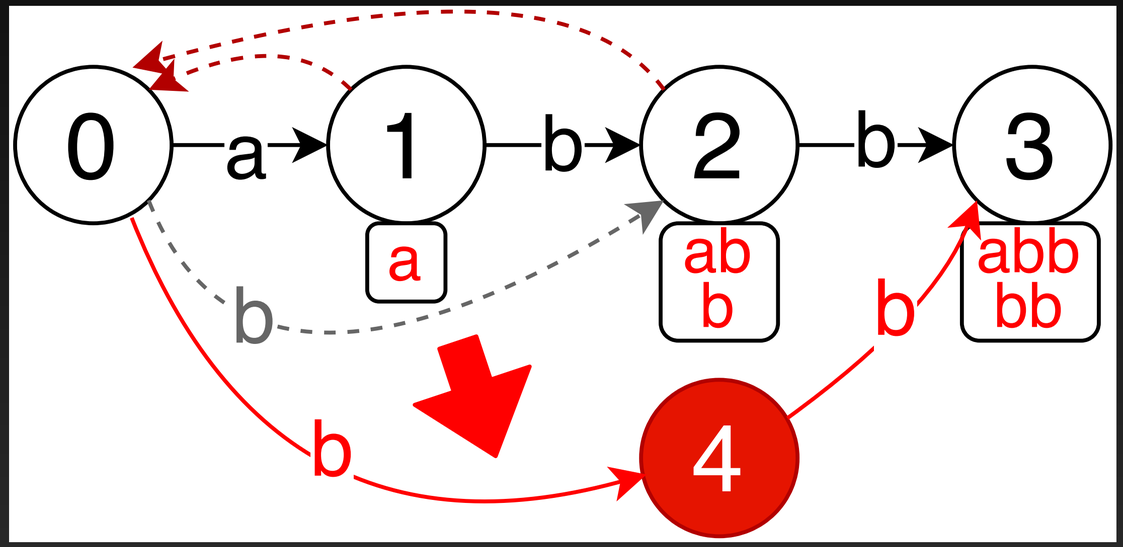
現在の文字列: “abb”



・ノード 0 から辿り，‘b’ の辺がノード 2 に
つながっていたら，ノード 4 につなぎかえる

具体的な構築方法

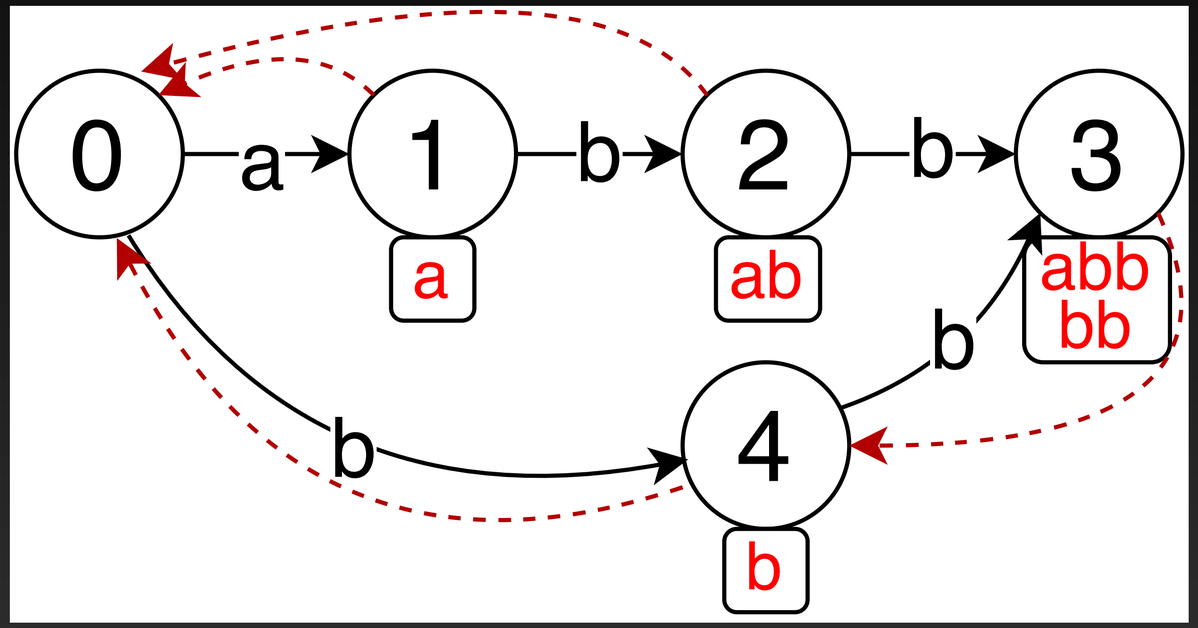
現在の文字列: “abb”



・ノード 0 から辿り，‘b’ の辺がノード 2 に
つながっていたら，ノード 4 につなぎかえる

具体的な構築方法

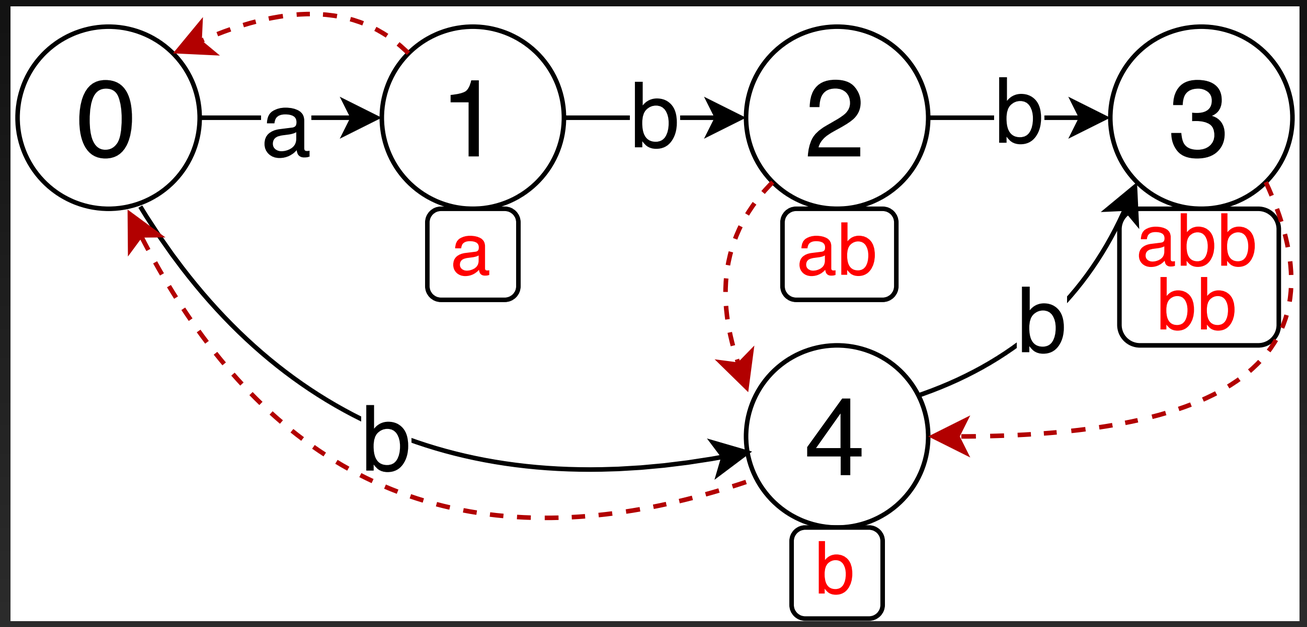
現在の文字列: “abb”



・ノード 3 からノード 4 に Suffix Link をつなぐ

具体的な構築方法

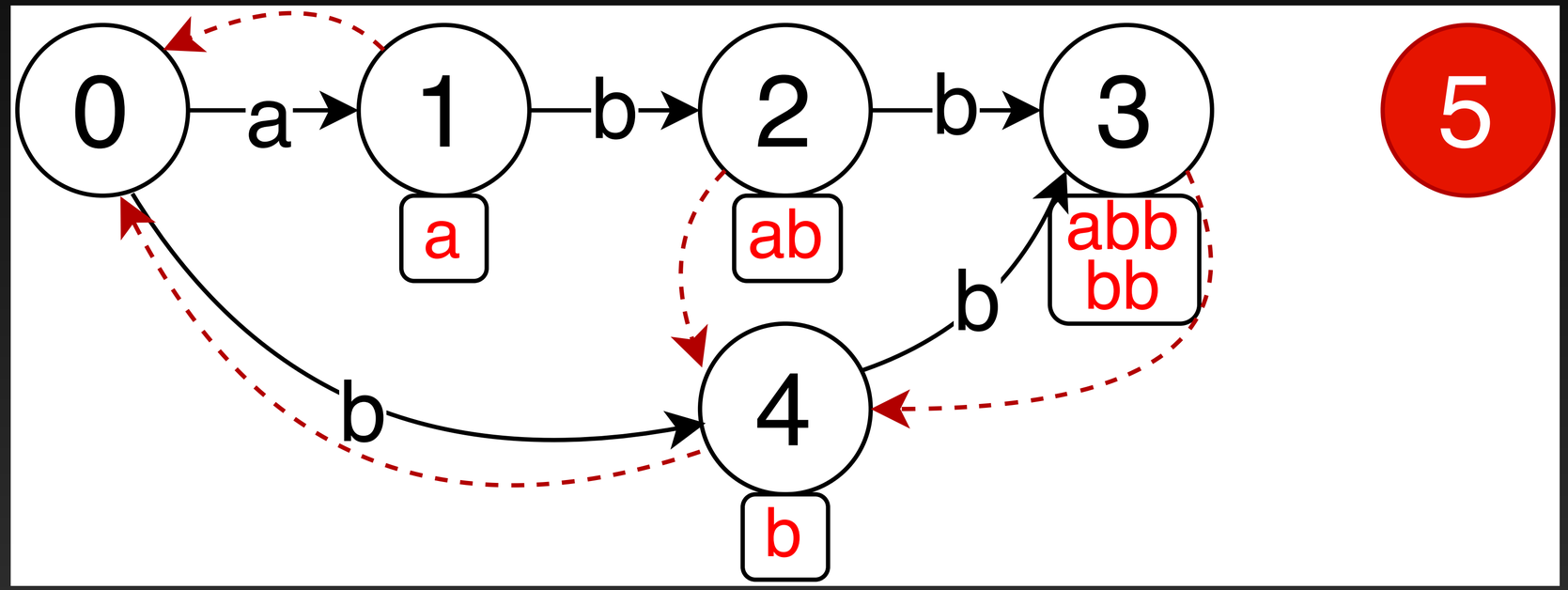
現在の文字列: “abb”



・ノード 2 からノード 4 に Suffix Link をつなぐ

具体的な構築方法

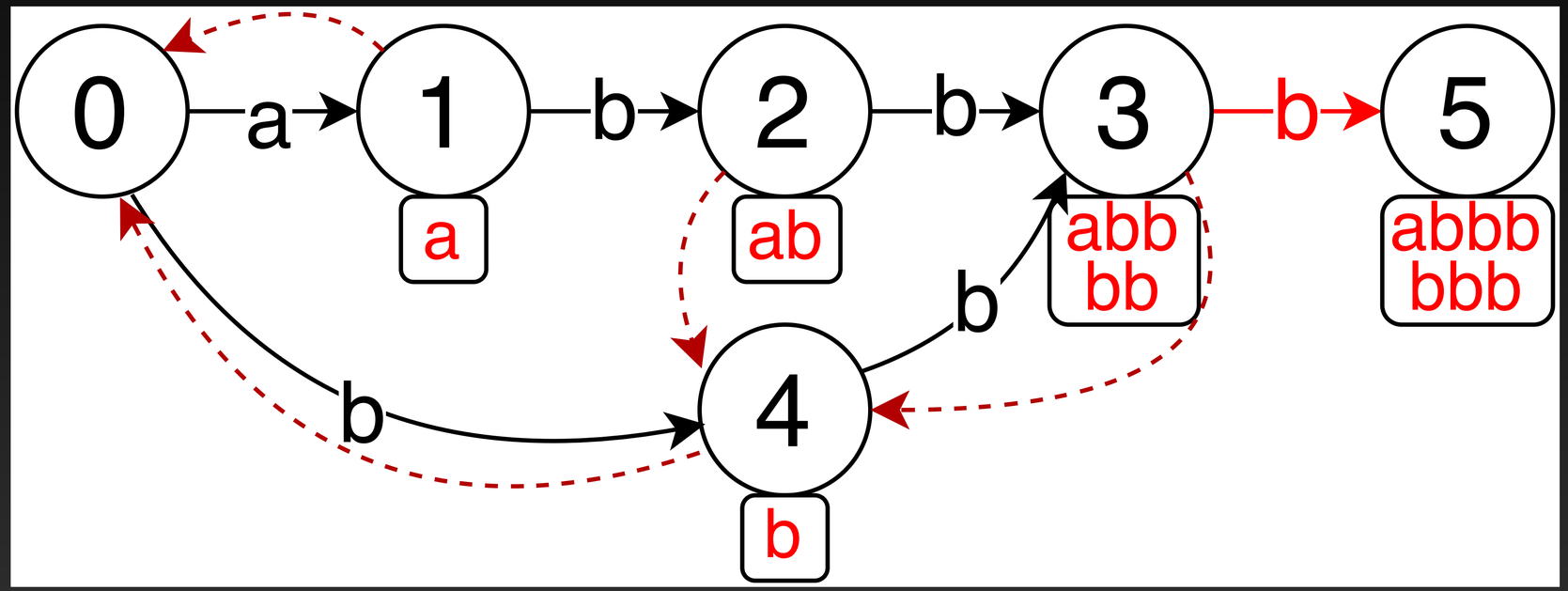
現在の文字列: “abbb”



・先ほどと同様に

具体的な構築方法

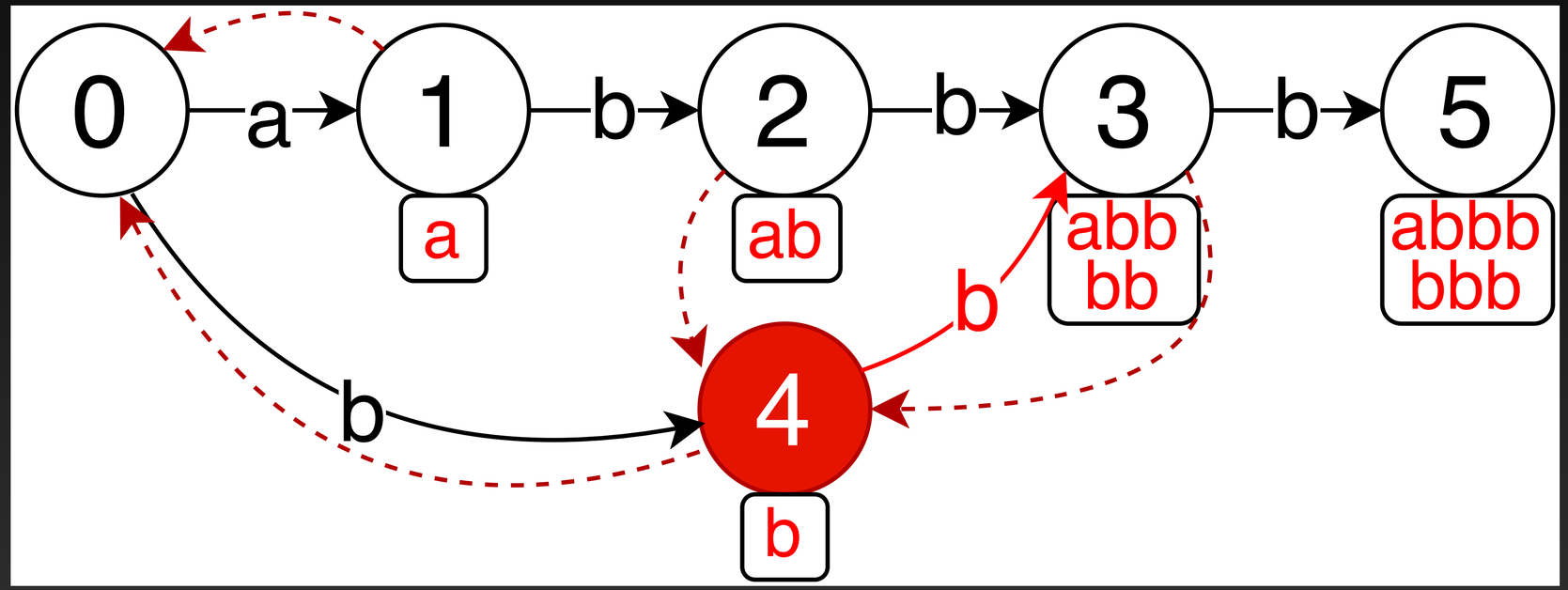
現在の文字列: “abbb”



・先ほどと同様に

具体的な構築方法

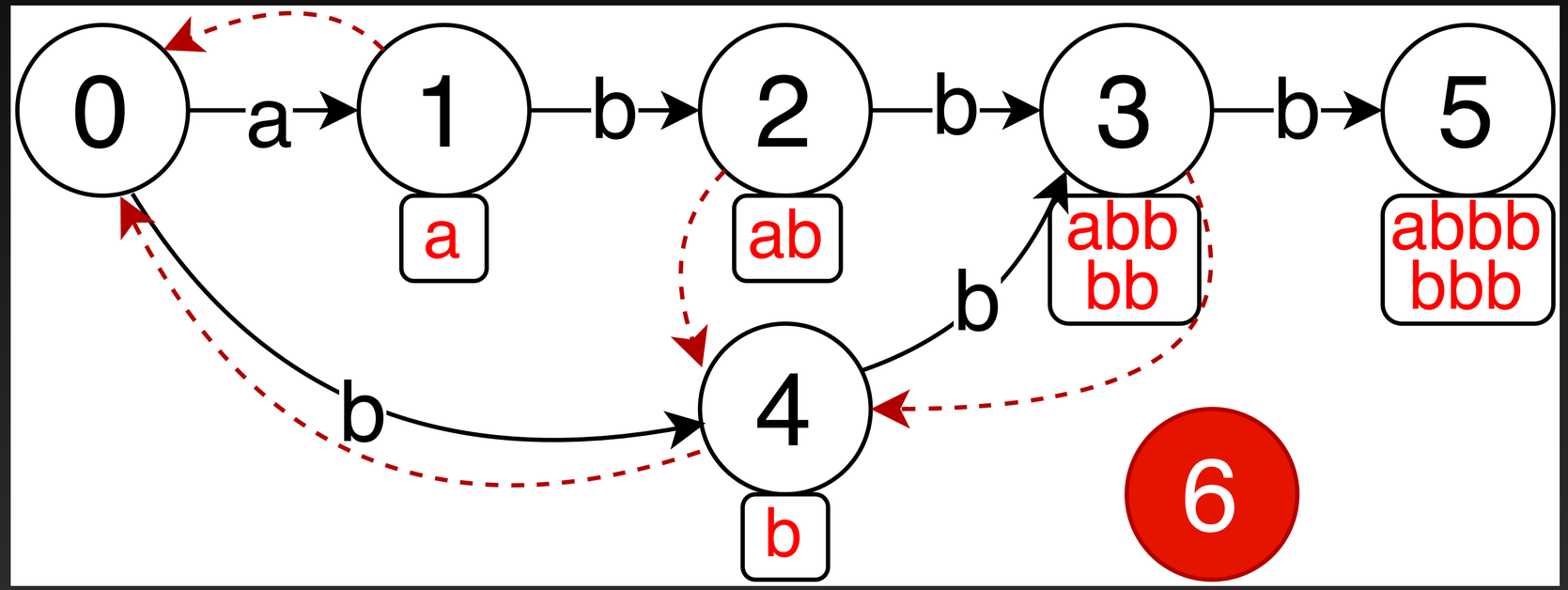
現在の文字列: “abbb”



・先ほどと同様に

具体的な構築方法

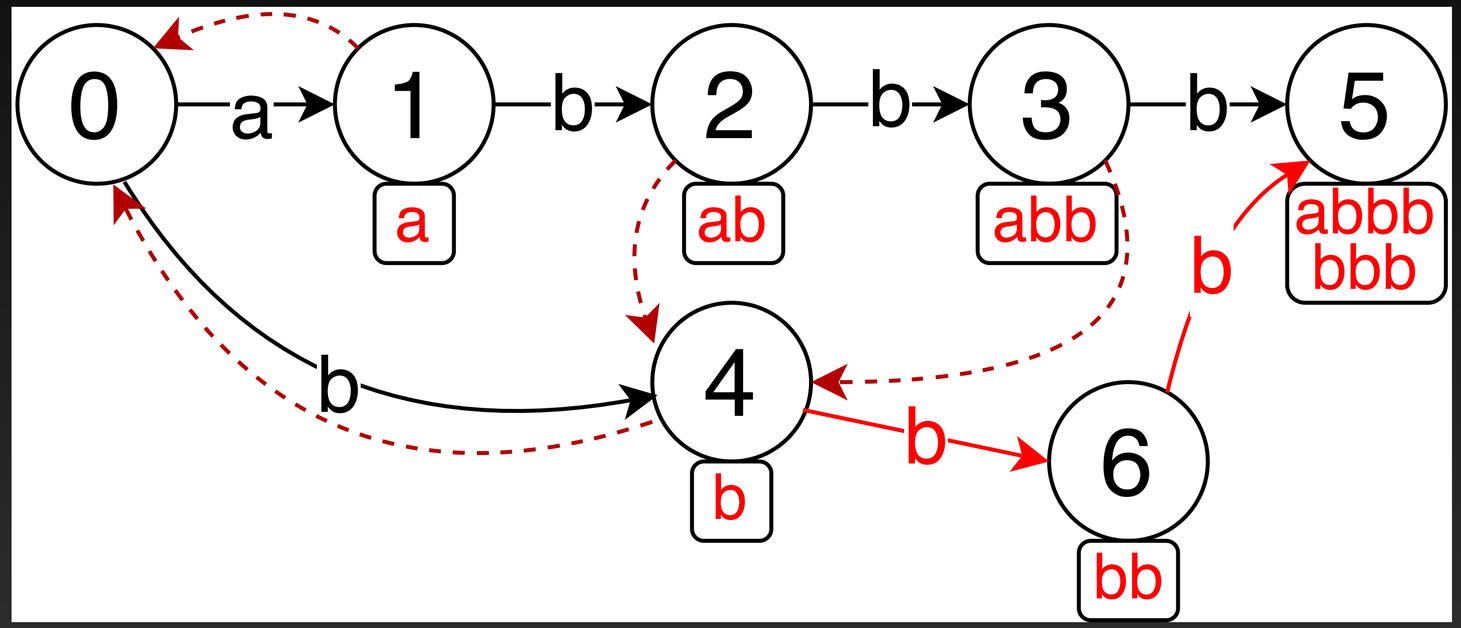
現在の文字列: “abbb”



・先ほどと同様に

具体的な構築方法

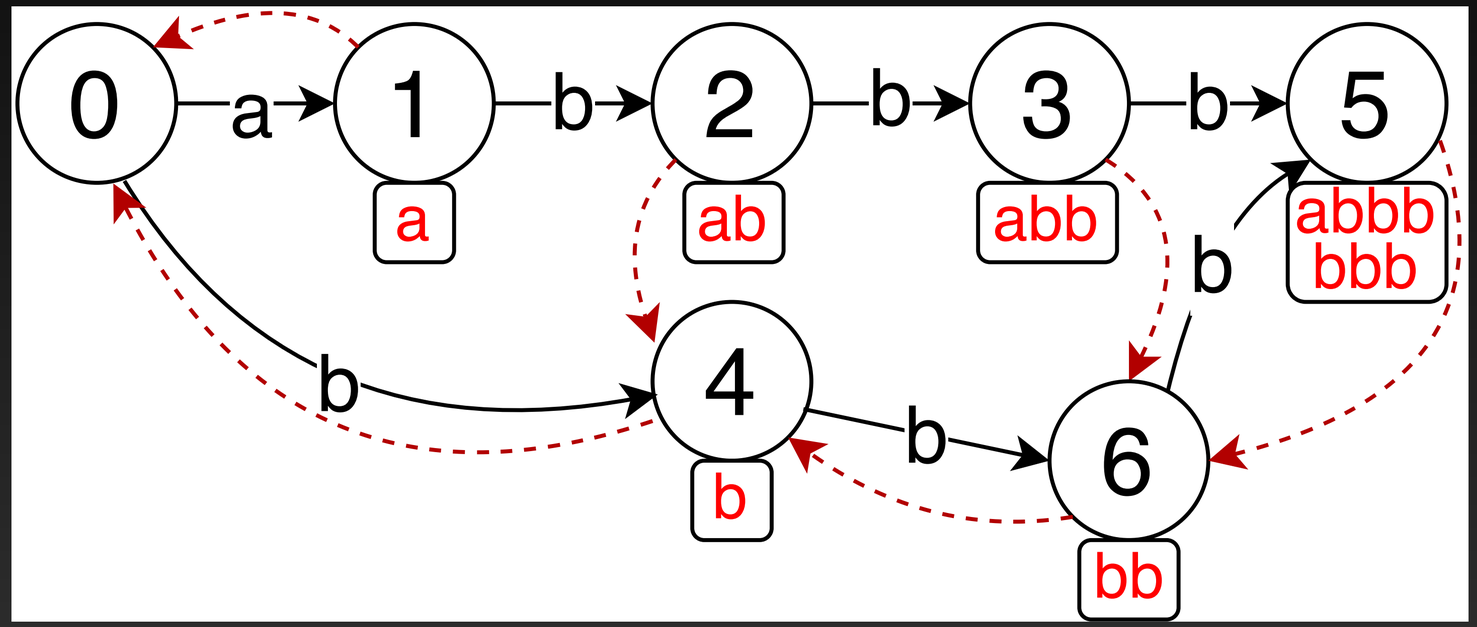
現在の文字列: “abbb”



・先ほどと同様に

具体的な構築方法

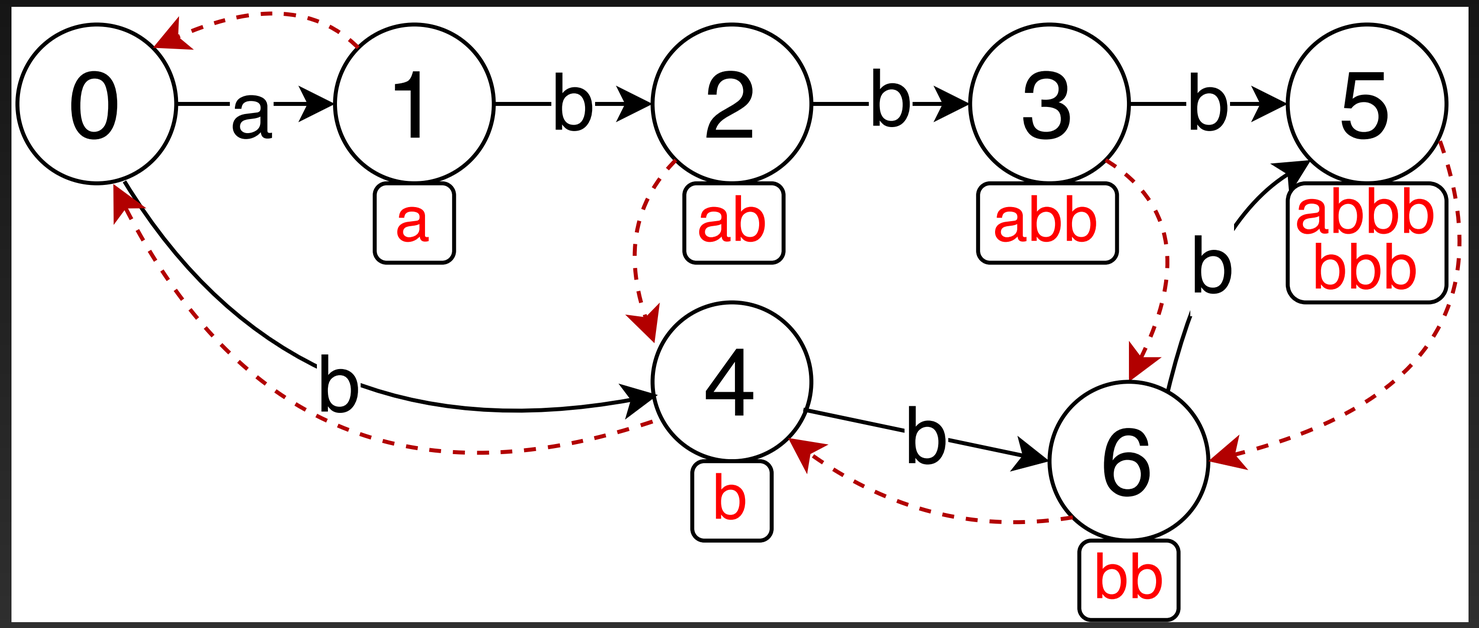
現在の文字列: “abbb”



・先ほどと同様に

具体的な構築方法

完成！



具体的な構築方法

- 実装しました

https://kodamad.github.io/JOISS2020/library/src/suffix_automaton.cpp.html

1. Suffix Automaton の性質

▶2. 具体的な構築方法

3. 計算量などの解析

4. 実際に使ってみよう！

1. Suffix Automaton の性質

2. 具体的な構築方法

▶3. 計算量などの解析

4. 実際に使ってみよう！

計算量などの解析

- 文字列 S の長さを n とおく

- 頂点数:
  一文字追加するごとに高々 2 個しか増えない
  始点も入れて高々 2n+1 個

計算量などの解析

- 辺数:
  各ノードについて始点からの最長パスのみ
  を使ったような全域木を考える
  この全域木の辺数は高々 2n

計算量などの解析

- 辺数:
  全域木に含まれない辺 p -> q について考える
  - p -> q は文字 c による辺だとする
  - 始点 -> p の最長パスが表す文字列… P
  - q -> 最遠点 の最長パスが表す文字列… Q

計算量などの解析

- 辺数:
  - Q は S の接尾辞
  - P+c+Q は S の部分文字列

  よって P+c+Q は S の接尾辞

  さらに, (p, q) が異なるとき, P+c+Q も異なる

  (部分文字列は重複なく管理されている)

計算量などの解析

- 辺数:
  S の相異なる接尾辞は n 個以下
  -> 全域木に含まれない辺は高々 n
  -> Suffix Automaton の辺数は高々 3n

計算量などの解析

- 構築

  文字の種類数を K として

  - 空間 O(N), 時間 O(NlogK)
  - 空間 O(NK), 時間 O(N)

1. Suffix Automaton の性質

2. 具体的な構築方法

▶3. 計算量などの解析

4. 実際に使ってみよう！

1. Suffix Automaton の性質

2. 具体的な構築方法

3. 計算量などの解析

▶4. 実際に使ってみよう！

実際に使ってみよう！

Q. 文字列 S の部分文字列の種類数を求めよ.

<考察>

- Suffix Automaton におけるパスは,
  S の部分文字列を表している
- パスを数え上げればよい

実際に使ってみよう！

Q. 文字列 S の部分文字列の種類数を求めよ.

<解法>

・トポロジカルソートする

・dp[始点] = 1 として配り，総和を求める

実際に使ってみよう！

- 他にも…?
  - 部分文字列のうち辞書順で k 番目
  - Suffix Array を構築
  - パターン T が何回出現するか

  など, いろいろ出来ます

おわりに

・昨年と比べて理解力が上がった(?)

・有意義に過ごせて楽しかった

・オンライン開催，寂しくない？

ありがとうございました

JOI 夏季セミナー 2020　KoD