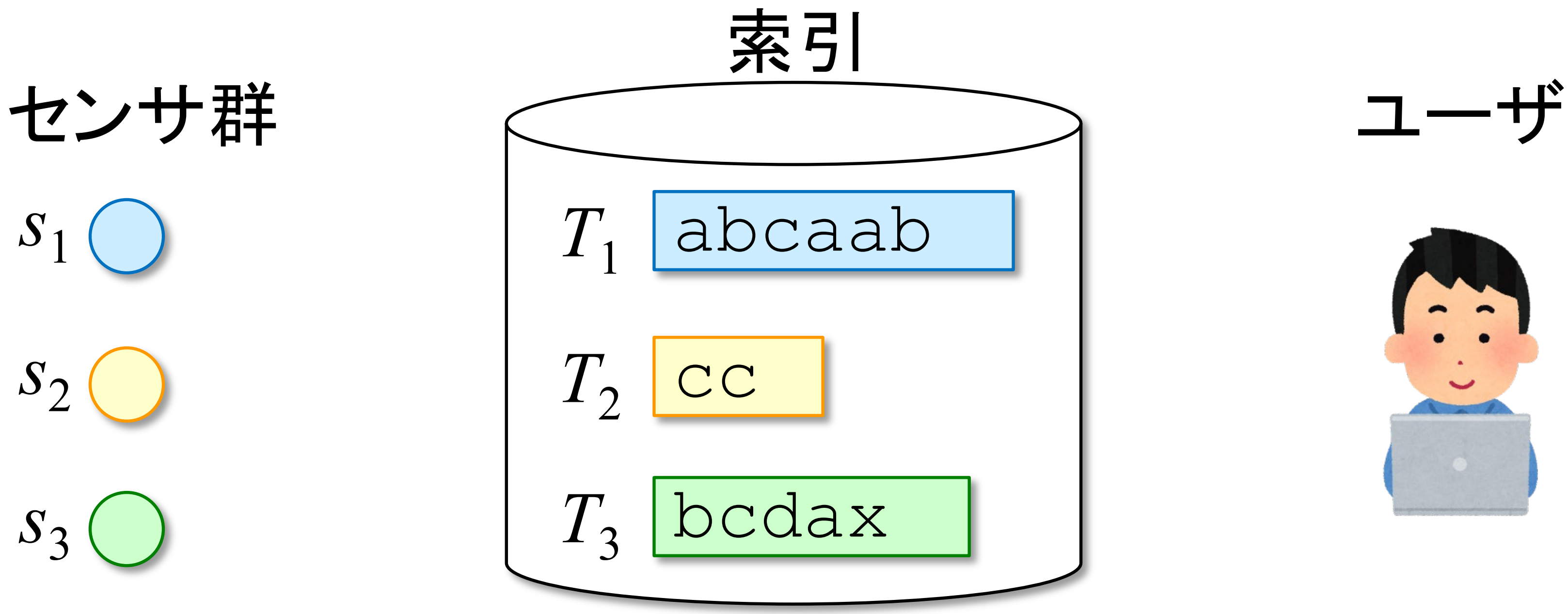
複数テキスト索引構造のオンライン構築

高木 拓也（富士通研究所）
○ 稲永 俊介（九州大学）
有村 博紀（北海道大学）
Dany Breslauer（無所属）
Diptarama Hendrian（東北大学）

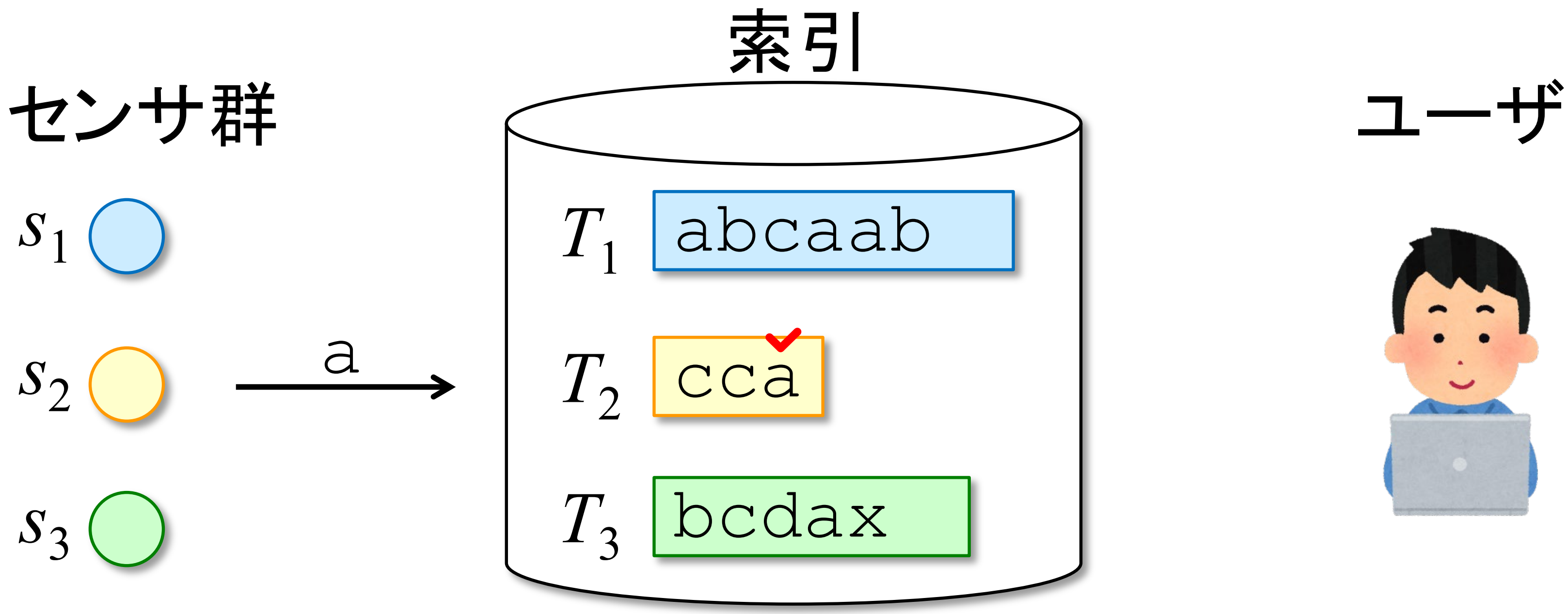
オンライン複数文字列の索引

- 問題: 末尾に文字の追加を許す複数の文字列に対する索引を高速に構築せよ.
- 動機: マルチストリームデータ処理
  - 各種センサ, トラジェクトリ, Twitter, などなど



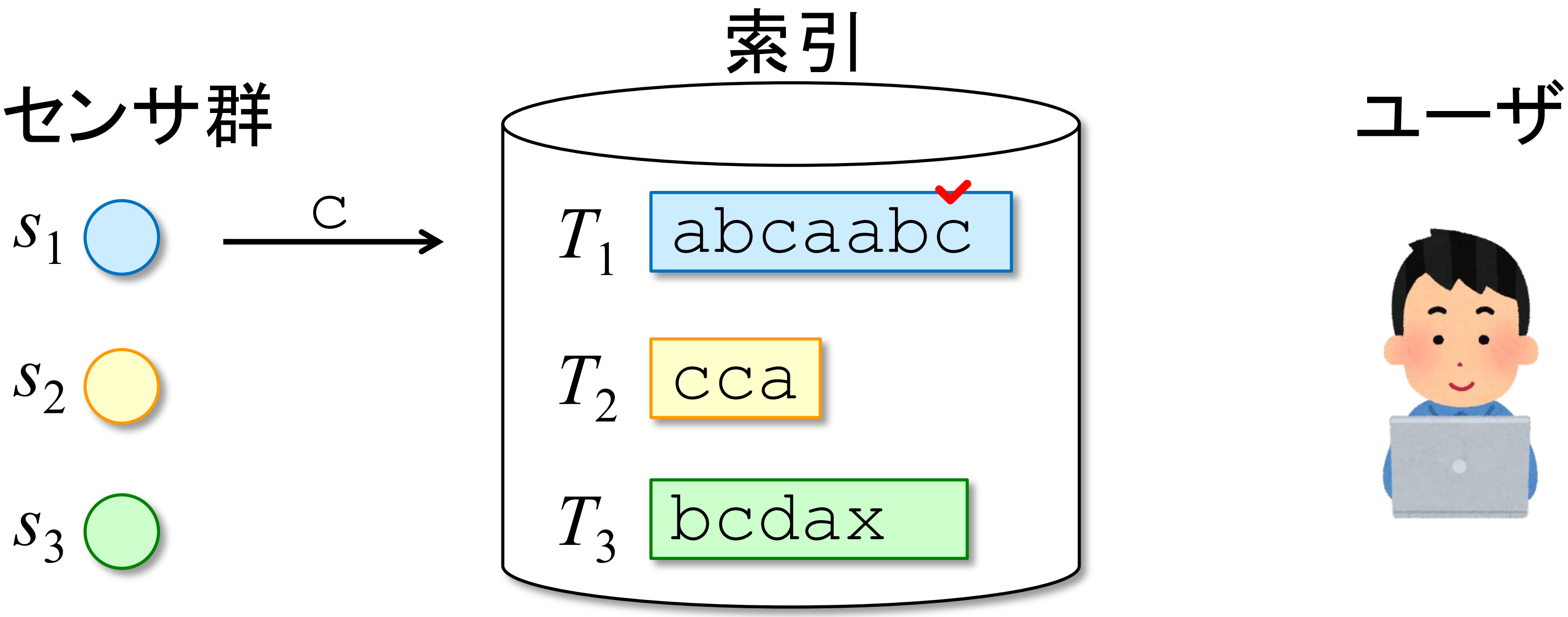
オンライン複数文字列の索引

- 問題: 末尾に文字の追加を許す複数の文字列に対する索引を高速に構築せよ.
- 動機: マルチストリームデータ処理
  - 各種センサ, トラジェクトリ, Twitter, などなど



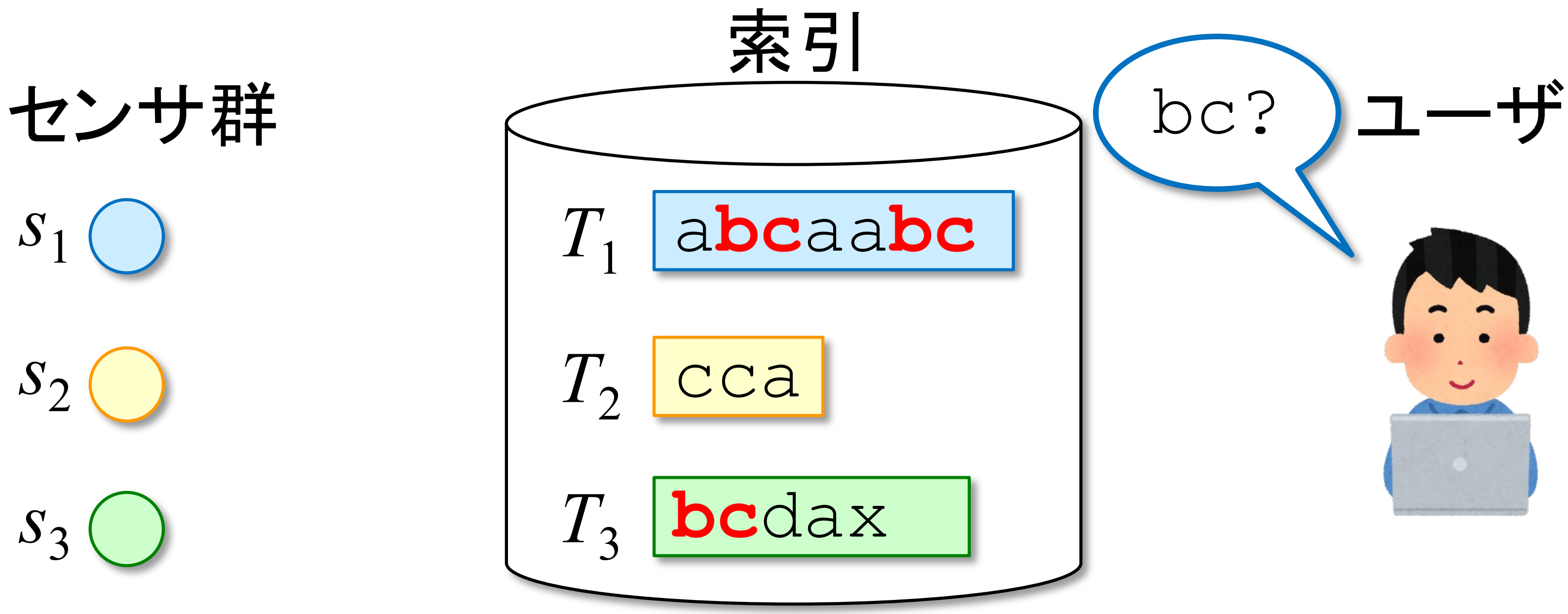
オンライン複数文字列の索引

- 問題: 末尾に文字の追加を許す複数の文字列に対する索引を高速に構築せよ.
- 動機: マルチストリームデータ処理
  - 各種センサ, トラジェクトリ, Twitter, などなど



オンライン複数文字列の索引

- 問題: 末尾に文字の追加を許す複数の文字列に対する索引を高速に構築せよ.
- 動機: マルチストリームデータ処理
  - 各種センサ, トラジェクトリ, Twitter, などなど

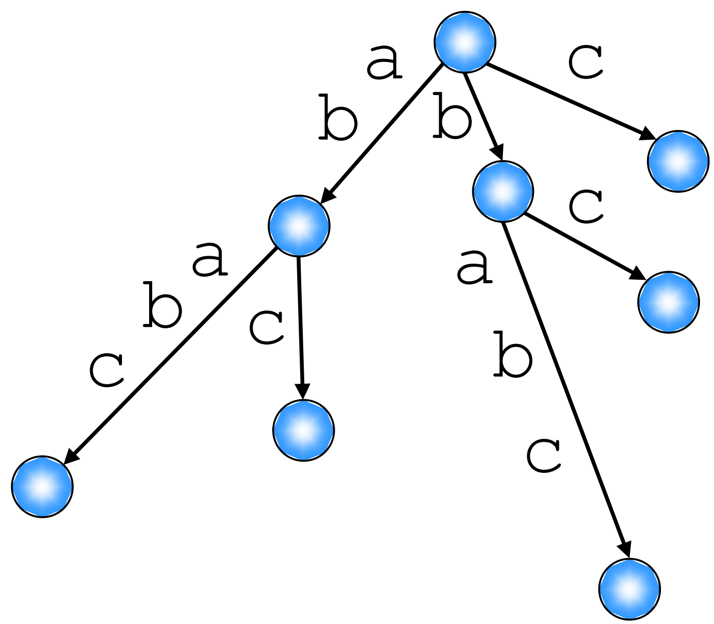


接尾辞木 (Suffix Tree)

- 入力文字列 T のすべての接尾辞を表現する根つき木 (オートマトンの亜種)

T = ababc

T の接尾辞
ababc
babc
abc
bc
c



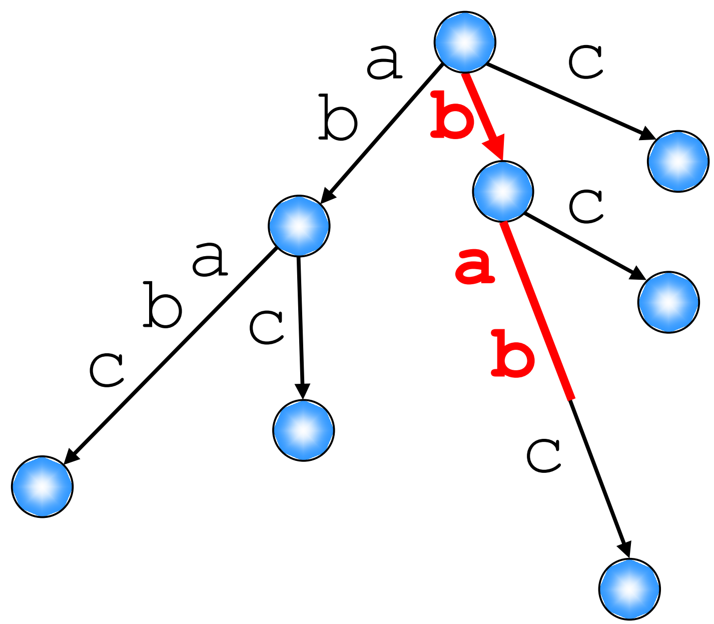
接尾辞木 (Suffix Tree)

- 入力文字列 T のすべての接尾辞を表現する根つき木 (オートマトンの亜種)

T = a**bab**c

T の接尾辞

ababc
babc
abc
bc
c



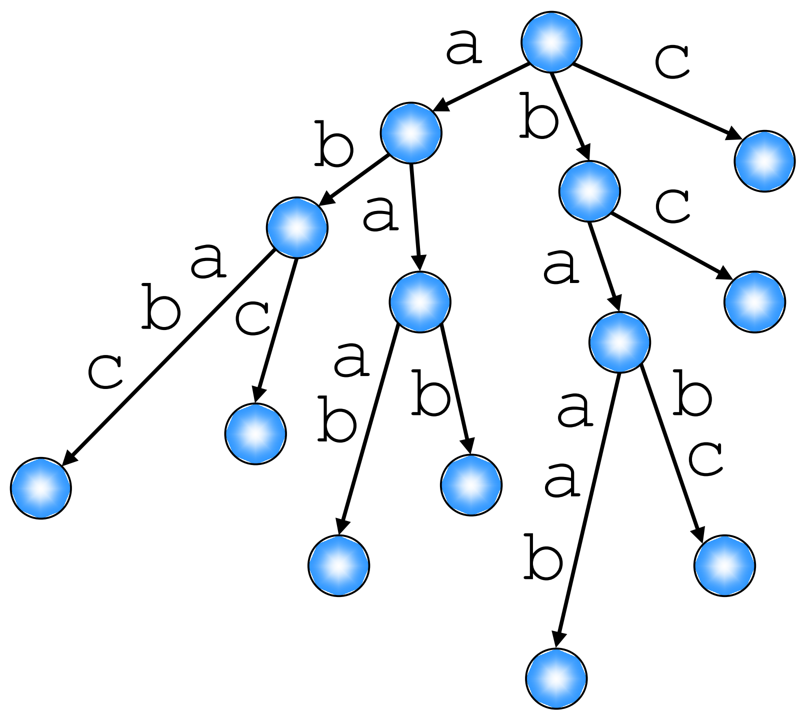
複数文字列に対する接尾辞木

- 複数の入力文字列 $T_1, \ldots, T_K$ の
  すべての接尾辞を表現する根つき木

$T_1 = \mathrm{a\,b\,a\,b\,c}$

$T_2 = \mathrm{b\,a\,a\,a\,b}$

$K = 2$ の場合



複数文字列に対する接尾辞木

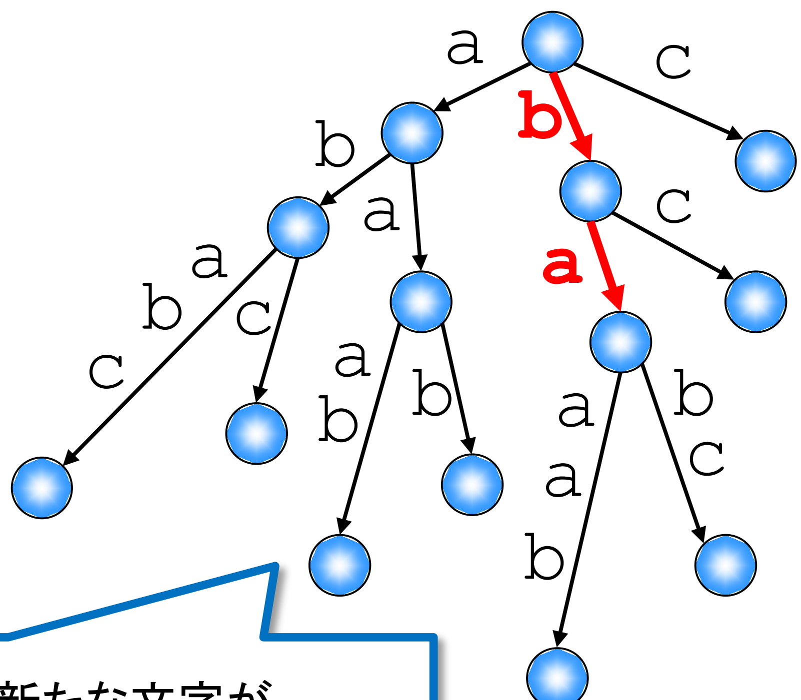
- 複数の入力文字列 $T_1, \ldots, T_K$ の
  すべての接尾辞を表現する根つき木

複数文字列の共通パターン発見

$T_1 =$ a **ba** b c

$T_2 =$ **ba** a a b

$K = 2$ の場合



b a ?

各文字列 T_i の末尾/先頭に新たな文字が
追加される毎に，接尾辞木を高速に更新したい

単一文字列の場合

【本研究の出発点】
複数文字列の場合は？

右左オンライン構築 [Weiner 1973]

文字を**先頭に**追加していく単一文字列の接尾辞木を
$O(n \log \sigma)$時間・$O(n)$ 領域で構築できる.

左右オンライン構築 [Ukkonen 1995]

文字を**末尾に**追加していく単一文字列の接尾辞木を
$O(n \log \sigma)$時間・$O(n)$ 領域で構築できる.

n: 文字列の長さ, σ: 文字種類数

※ 比較モデルでは $O(n \log \sigma)$ 時間が最適
← ソートの下界 $\Omega(n \log \sigma)$ が適用される

研究の初期段階　2015年5月～

<u>**右左**</u>オンライン（<u>**先頭**</u>に文字追加）については，
Weiner のアルゴリズムをそのまま
複数文字列に拡張できそうですね

高木君

有村先生

稲永

研究の初期段階

イケそうですね

イケてますすね

高木君

有村先生

稲永

研究の初期段階

左右オンライン（**末尾**に文字追加）については，
Ukkonen のアルゴリズムをそのまま
複数文字列に拡張するのは難しそうですね？

高木君

有村先生

稲永

研究の初期段階

そこをなんとか

高木君

有村先生

稲永

研究の初期段階

右左オンラインの接尾辞木を並行して構築することで，
左右オンラインの接尾辞木を構築できそうです

高木君

有村先生

稲永

研究の初期段階

イケそうですね

イケてますすね

高木君

有村先生

稲永

本研究にまつわるヒストリー

2015/5	研究開始（高木，稲永，有村）
2016/7	文字列処理に関する国際会議 CPM で発表

CPM 2016 の論文の主張

主張1（<u>右左</u>オンライン構築）

Weiner のアルゴリズムをそのまま適用することで，
<u>先頭に</u>文字が追加されていく**複数文字列**の
接尾辞木を $O(N \log \sigma)$時間・$O(N)$ 領域で構築できる.

主張2（<u>左右</u>オンライン構築）

主張1の結果と，Ukkonen のアルゴリズムを用いて，
<u>末尾に</u>文字が追加されていく**複数文字列**の
接尾辞木を $O(N \log \sigma)$時間・$O(N)$ 領域で構築できる.

N: 複数文字列の長さの総和，σ: 文字種類数

Dany からメールが届く 2017年1月

Dany Breslauer

あなた方の CPM の論文を読みました.
残念ながら**間違ってる**と思います.

Mjsk!?

高木君

有村先生

稲永

Dany からメールが届く 2017年1月

Dany Breslauer

マジです.
複数文字列のオンライン処理特有の問題を見逃していると思います.

やってもた...

高木君

有村先生

稲永

我々の CPM 2016 論文は

完全に間違い!!

主張1（右左オンライン構築）

Weiner のアルゴリズムをそのまま適用することで，**先頭に**文字が追加されていく**複数文字列**の接尾辞木を $O(N \log \sigma)$時間 $O(N)$ 領域で構築できる.

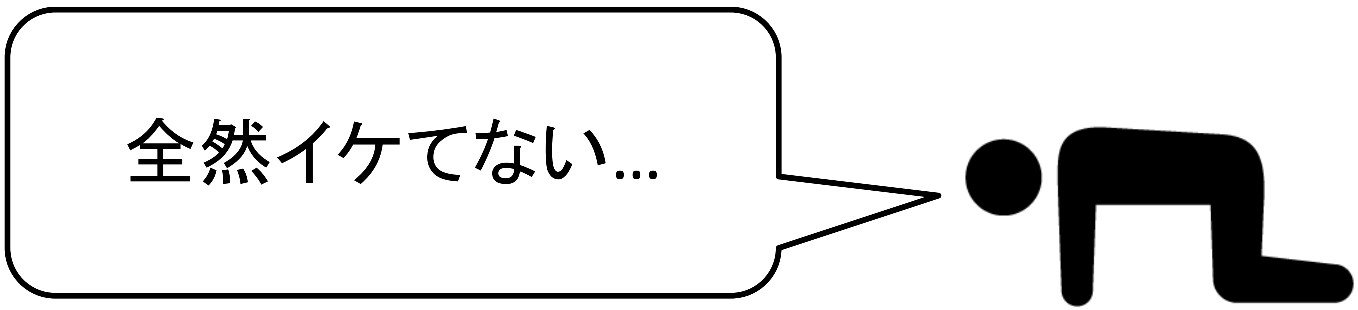
主張2（左右オンライン構築）

主張1の結果と，Ukkonen のアルゴリズムを用いて，**末尾に**文字が追加されていく**複数文字列**の接尾辞木を $O(n \log \sigma)$時間 $O(n)$ 領域で構築できる.

N: 複数文字列の長さの総和，σ: 文字種類数

本研究にまつわるヒストリー

2015/5	研究開始（高木，稲永，有村）
2016/7	文字列処理に関する国際会議 CPM で発表
2017/1	Dany から３人にメールが届き， CPM の論文に致命的なエラーが見つかる



チーム結成!!　2017年2～3月

高木君

有村先生

稲永

一緒に問題を
解決しましょう！

Dany

Diptarama君

Weiner アルゴリズムの下界

定理1

Weiner のアルゴリズムをそのまま
複数文字列に適用すると <u>$\Omega(N^{1.5})$ 時間かかる</u>.

ウソの主張1(再掲)

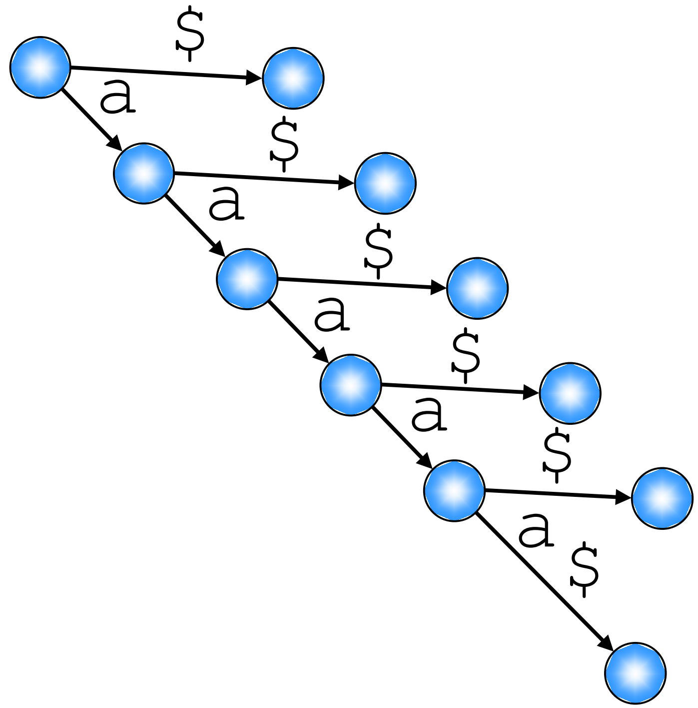
Weiner のアルゴリズムをそのまま適用することで,
<u>**先頭に**</u>文字が追加されていく**複数文字列**の接尾辞木を
<u>$O(N \log \sigma)$時間</u> $O(N)$領域で構築できる.

N: 複数文字列の長さの総和, σ: 文字種類数
➔ $\sigma \leq N$ が常に成り立つ

$\Omega(N^{1.5})$ 時間の証明

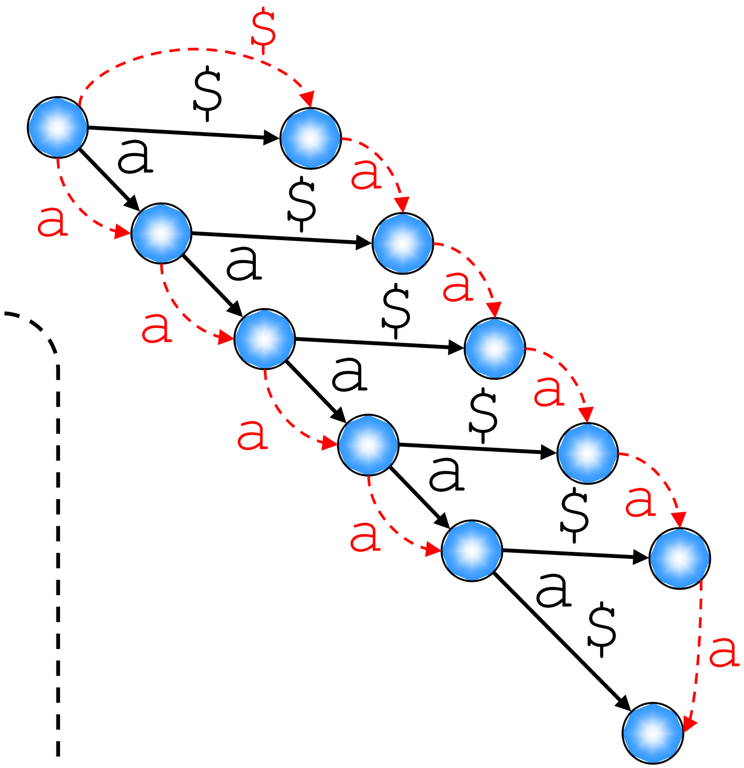
`aaaaa$`	T_1
`aaaa$`	T_2
`aaa$`	T_3
`aa$`	T_4
`a$`	T_5

$\Omega(N^{1.5})$ 時間の証明

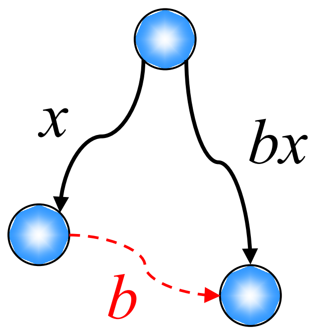


aaaaa\$ T_1
aaaa\$ T_2
aaa\$ T_3
aa\$ T_4
a\$ T_5

$\Omega(N^{1.5})$ 時間の証明



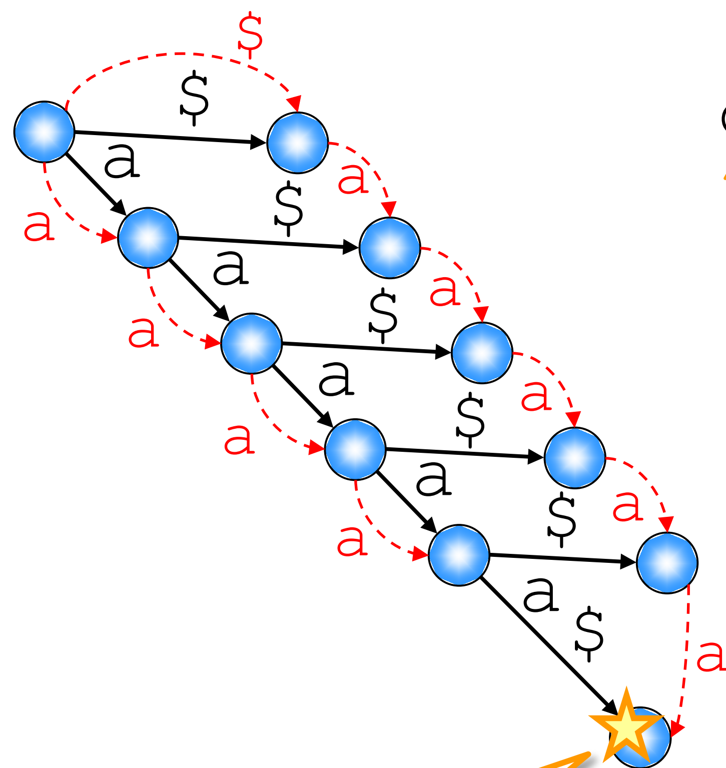
aaaaa\$ T_1
aaaa\$ T_2
aaa\$ T_3
aa\$ T_4
a\$ T_5



x は文字列, b は文字とする.
x と bx を表す頂点があるとき,
b でラベル付けされたリンクを
x から bx に向かって張る.

←文字列を左に伸ばすときに便利

$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1

aaaa$ T_2

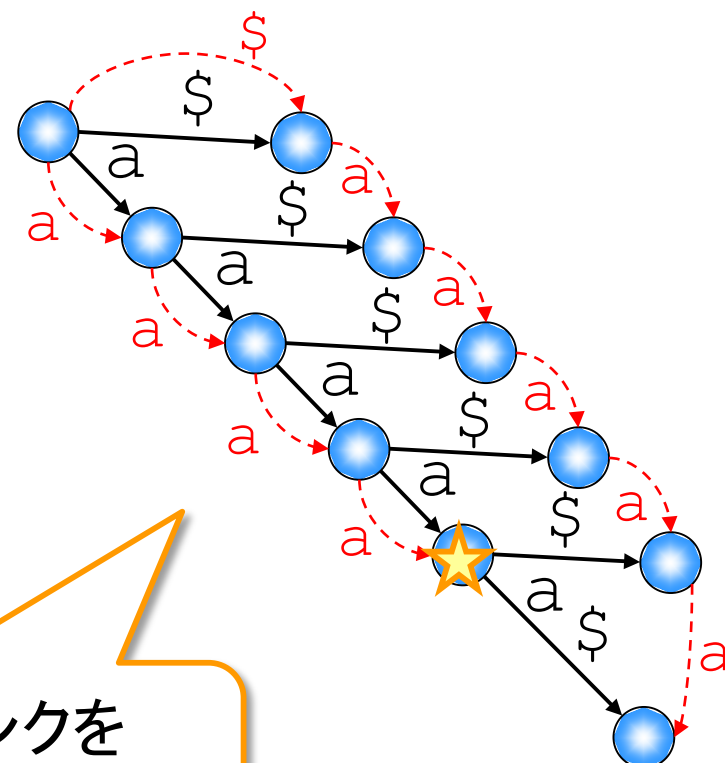
aaa$ T_3

aa$ T_4

a$ T_5

文字追加前の T_1 を
表す葉からスタート

$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1

aaaa\$ T_2

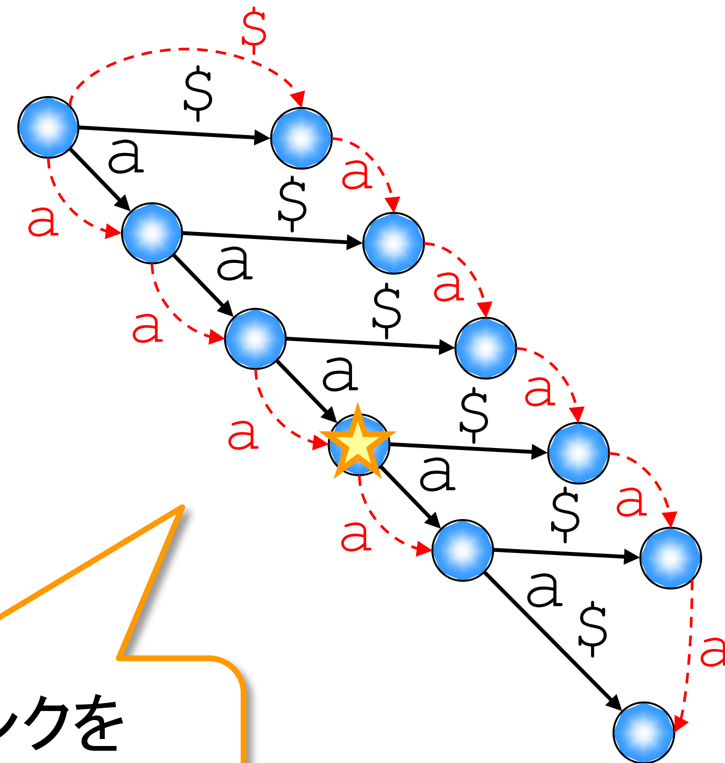
aaa\$ T_3

aa\$ T_4

a\$ T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1

aaaa$ T_2

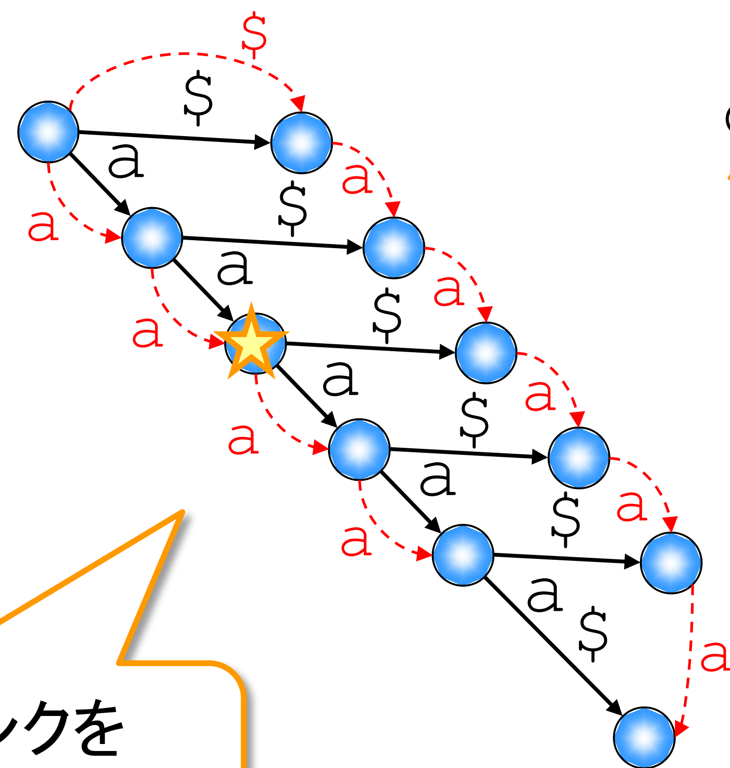
aaa$ T_3

aa$ T_4

a$ T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

$\Omega(N^{1.5})$ 時間の証明



c a a a a a $ T_1

△ a a a a $ T_2

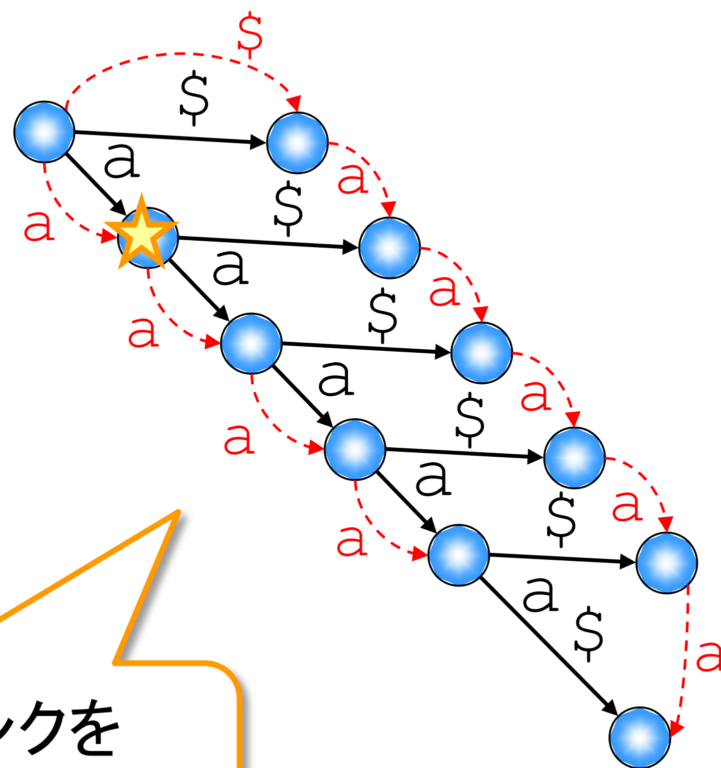
a a a $ T_3

a a $ T_4

a $ T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

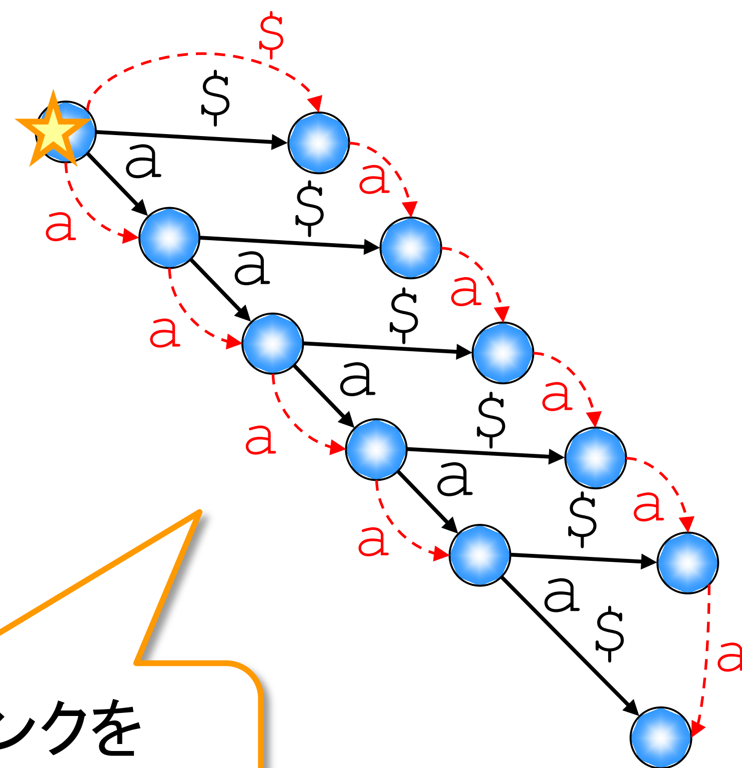
$\Omega(N^{1.5})$ 時間の証明



caaaaa$	T_1
△ aaaa$	T_2
aaa$	T_3
aa$	T_4
a$	T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

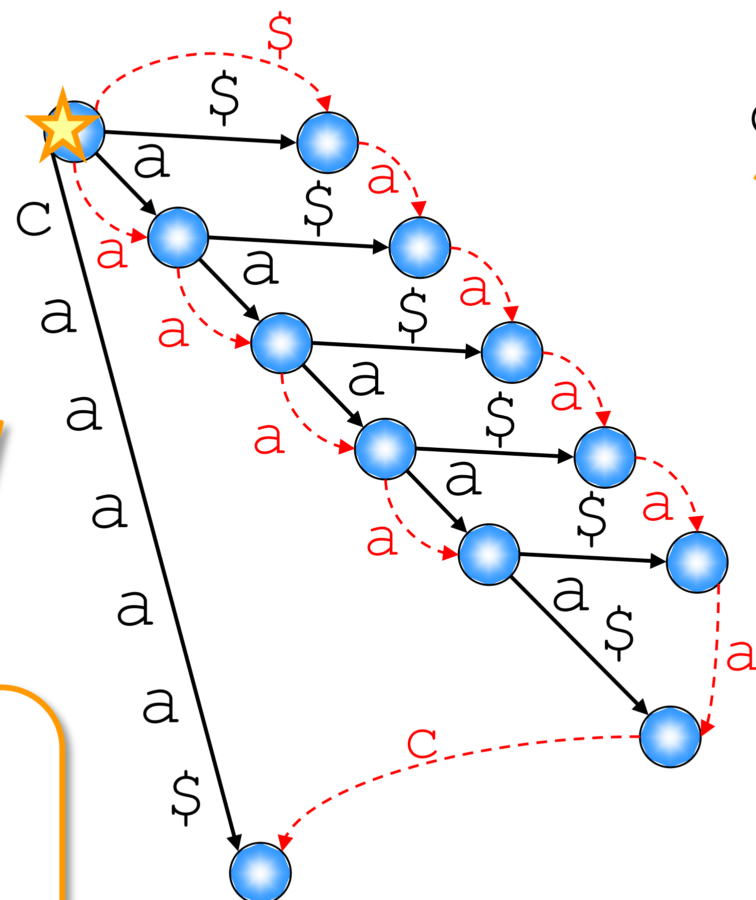
$\Omega(N^{1.5})$ 時間の証明



caaaaa$	T_1
aaaa$	T_2
aaa$	T_3
aa$	T_4
a$	T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1

aaaa\$ T_2

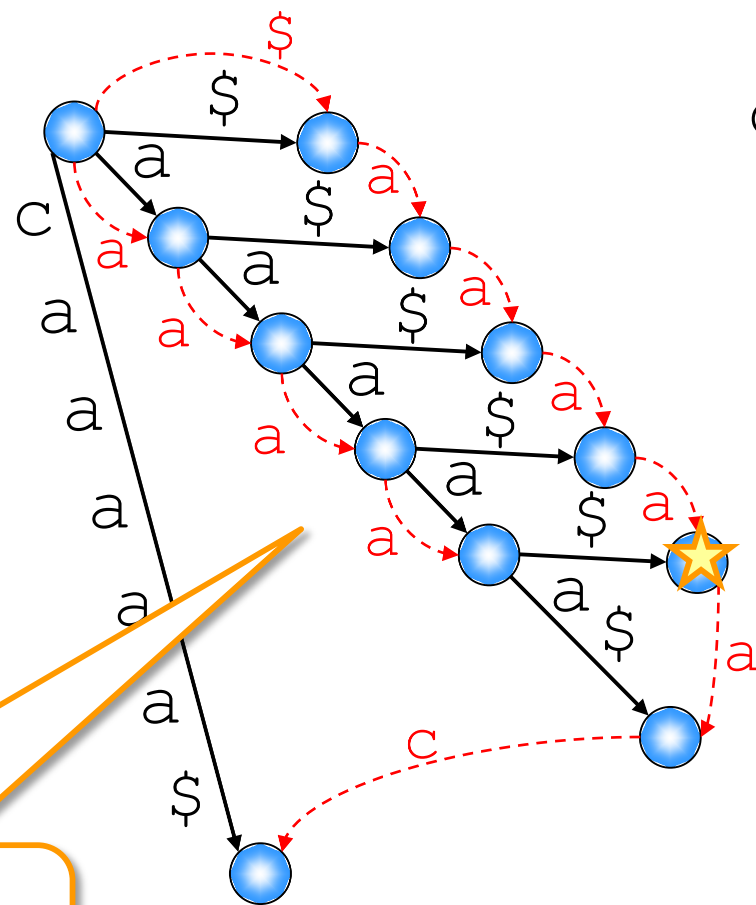
aaa\$ T_3

aa\$ T_4

a\$ T_5

根まで行っても c のリンクがなかったので，
文字追加後の T_1 を表す新たな葉を根から挿入

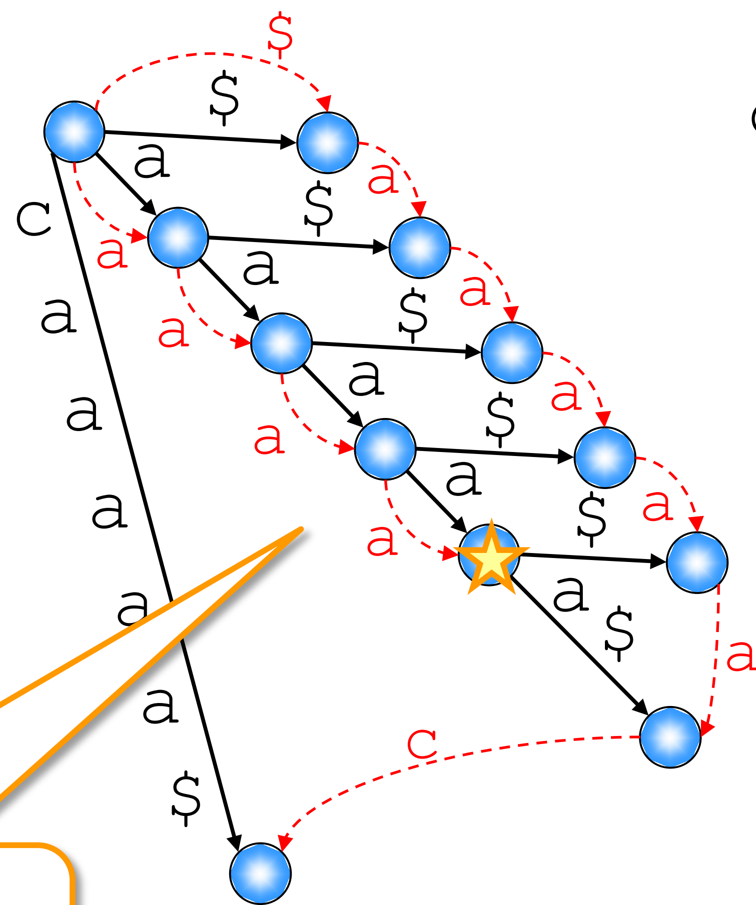
$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1
caaaa$ T_2
△ aaa$ T_3
aa$ T_4
a$ T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

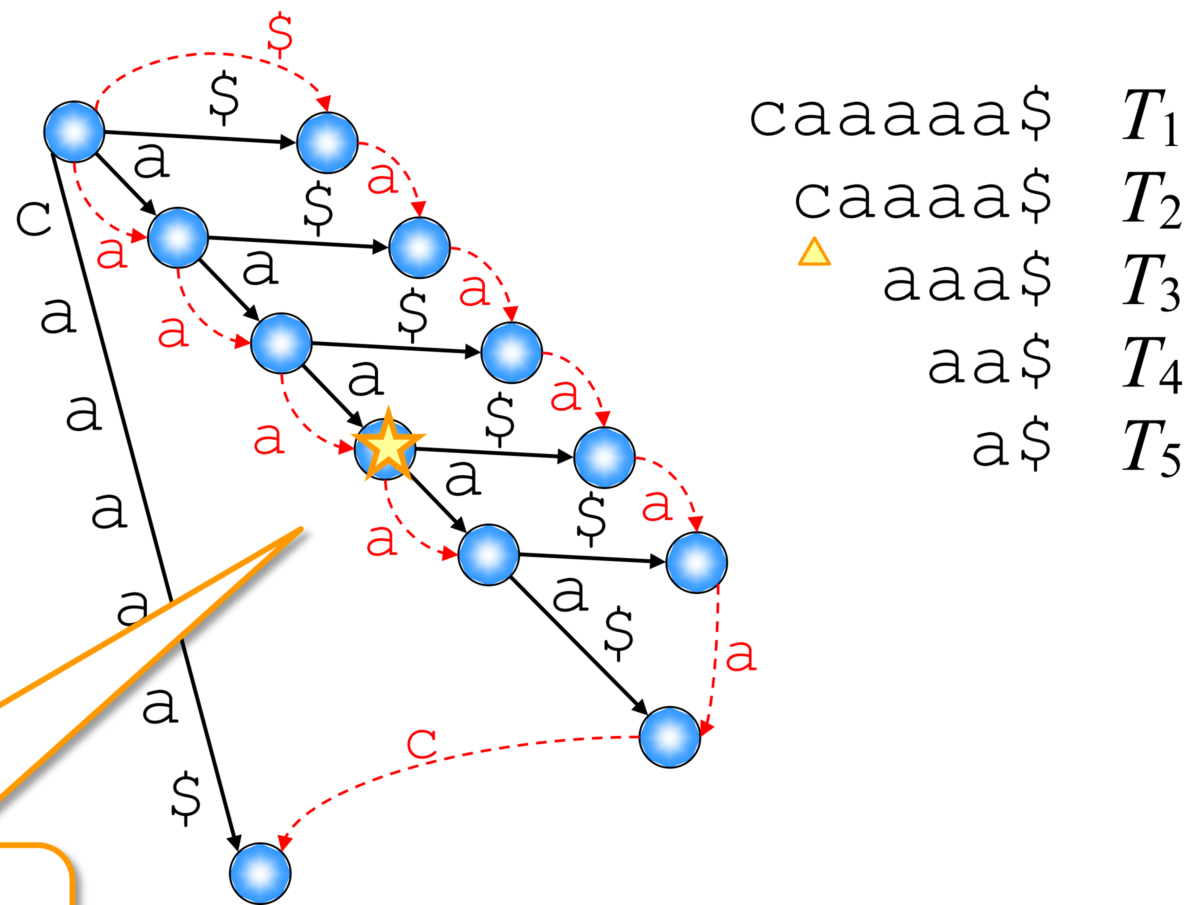
$\Omega(N^{1.5})$ 時間の証明




```
caaaaa$   T1
 caaaa$   T2
   aaa$   T3
    aa$   T4
     a$   T5
```

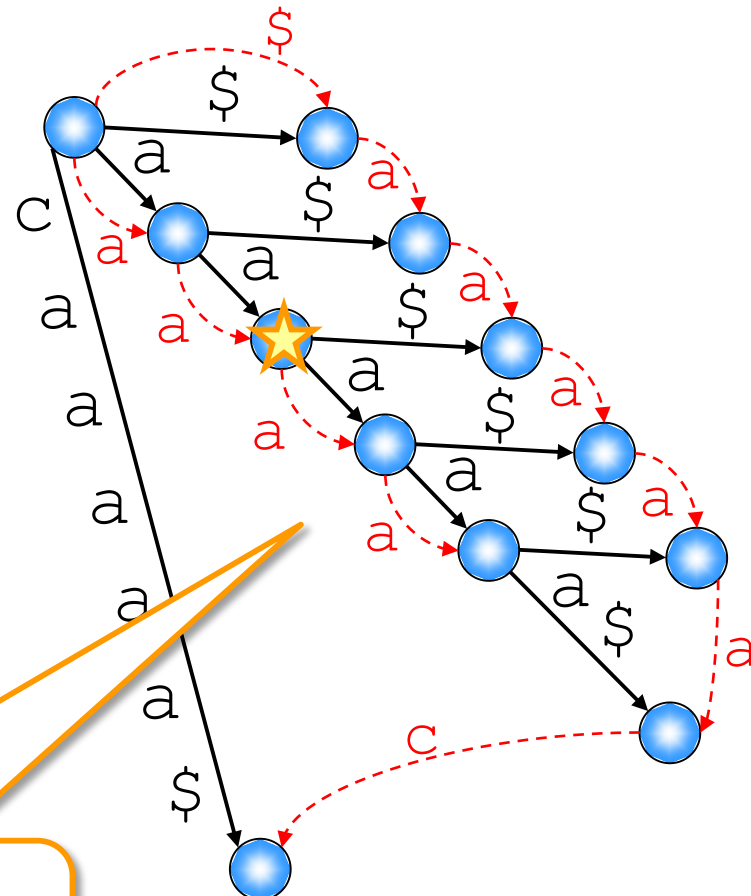
追加文字 c のリンクを
持つ最も深い頂点を探す

$\Omega(N^{1.5})$ 時間の証明



追加文字 c のリンクを
持つ最も深い頂点を探す

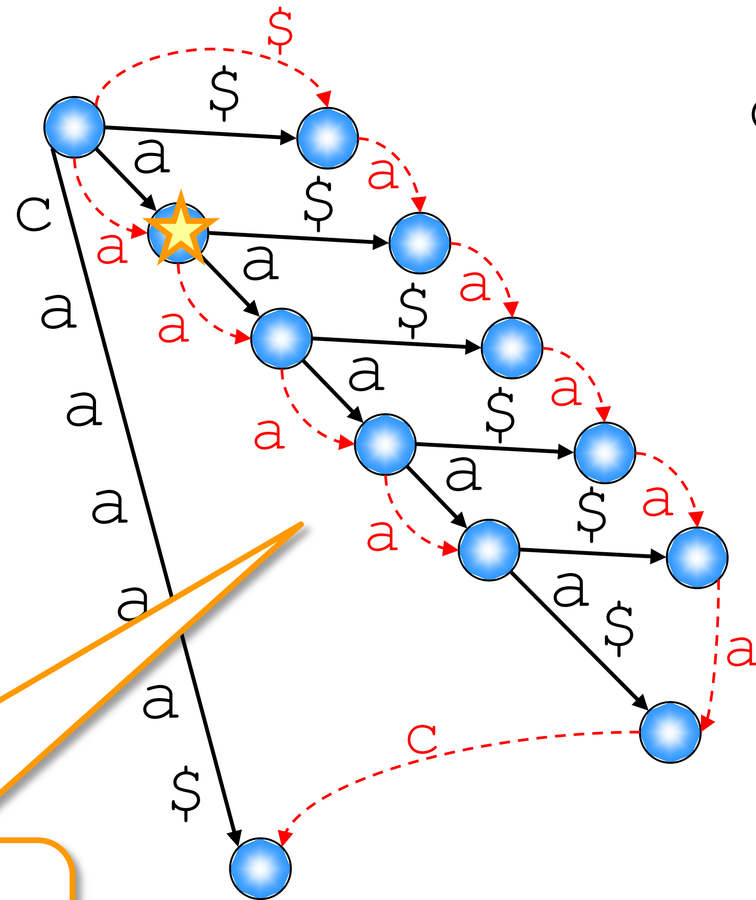
$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1
caaaa$ T_2
aaa$ T_3
aa$ T_4
a$ T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

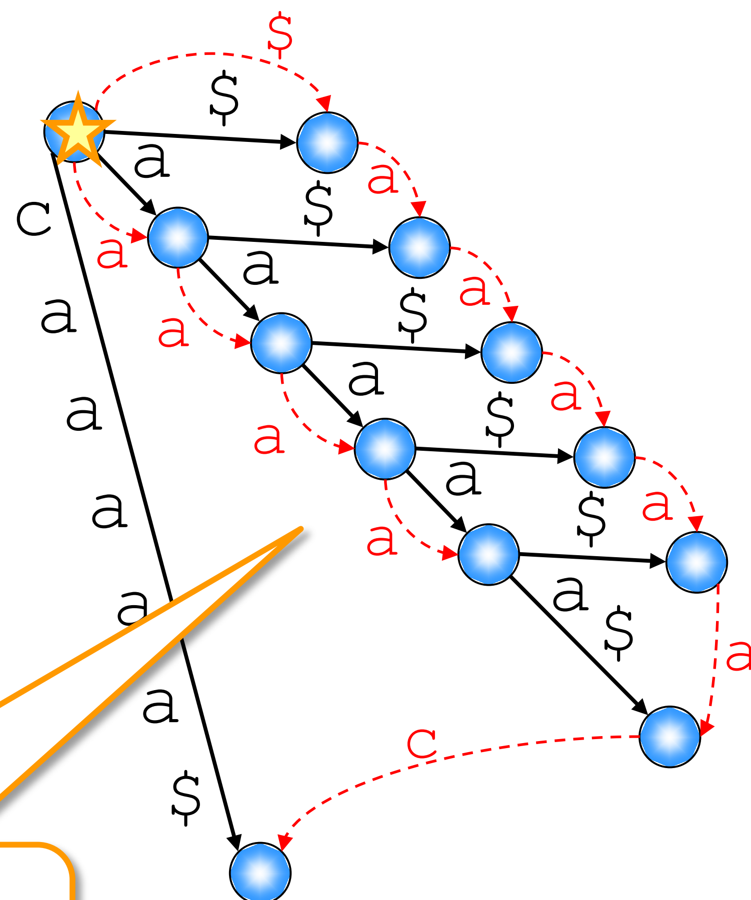
$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1
caaaa\$ T_2
aaa\$ T_3
aa\$ T_4
a\$ T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1

caaaa$ T_2

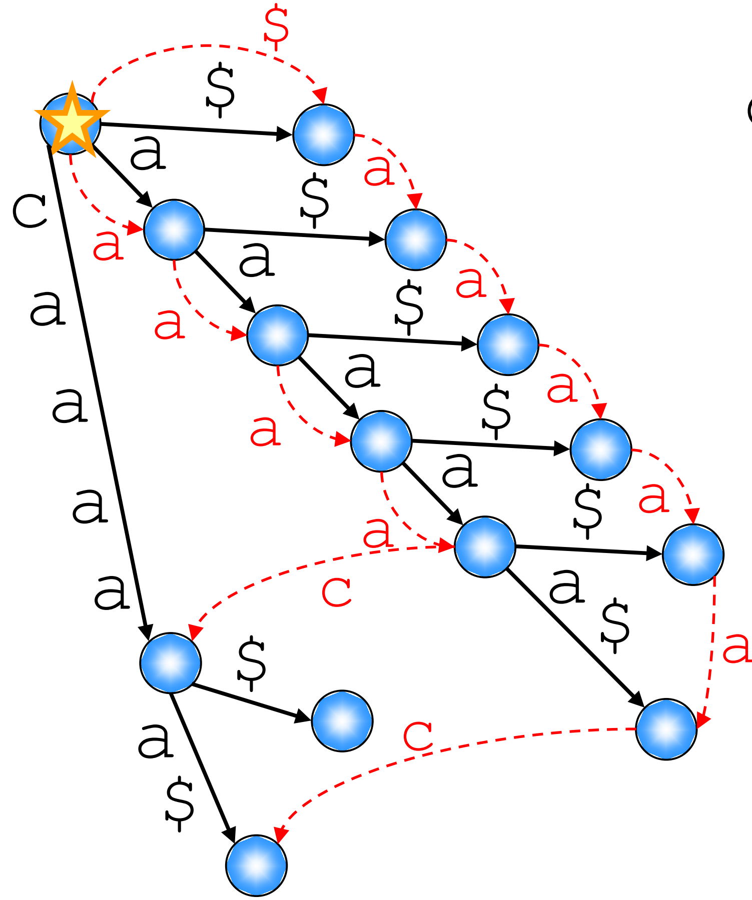
aaa$ T_3

aa$ T_4

a$ T_5

追加文字 c のリンクを
持つ最も深い頂点を探す

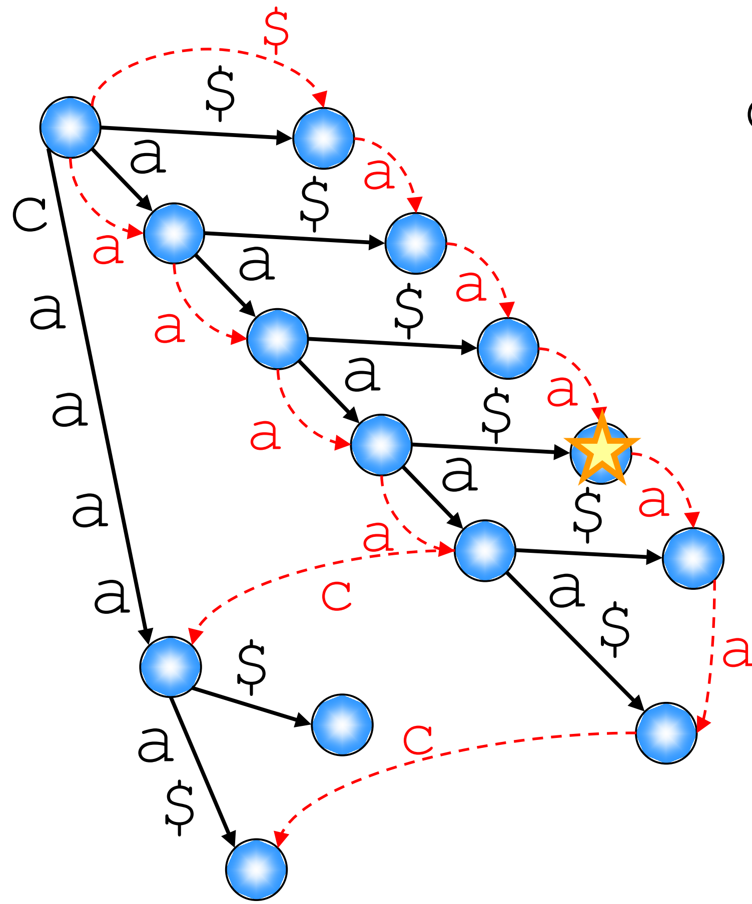
$\Omega(N^{1.5})$ 時間の証明



caaaaa$	T_1
caaaa$	T_2
△ aaa$	T_3
aa$	T_4
a$	T_5

※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1

caaaa\$ T_2

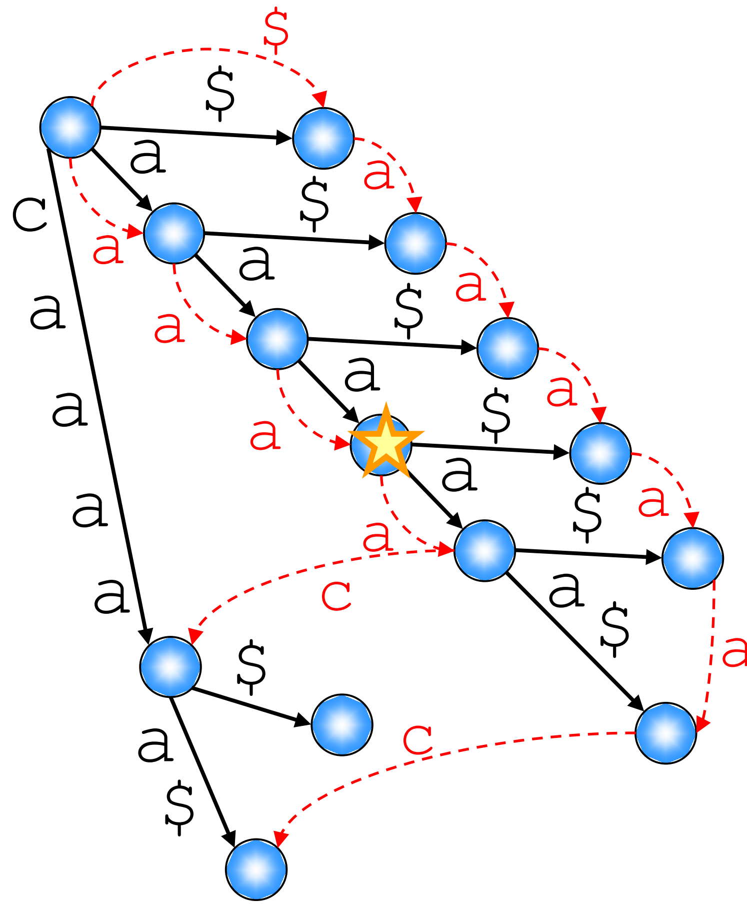
caaa\$ T_3

aa\$ T_4

a\$ T_5

※ このステップに関係ないリンクは一部省略

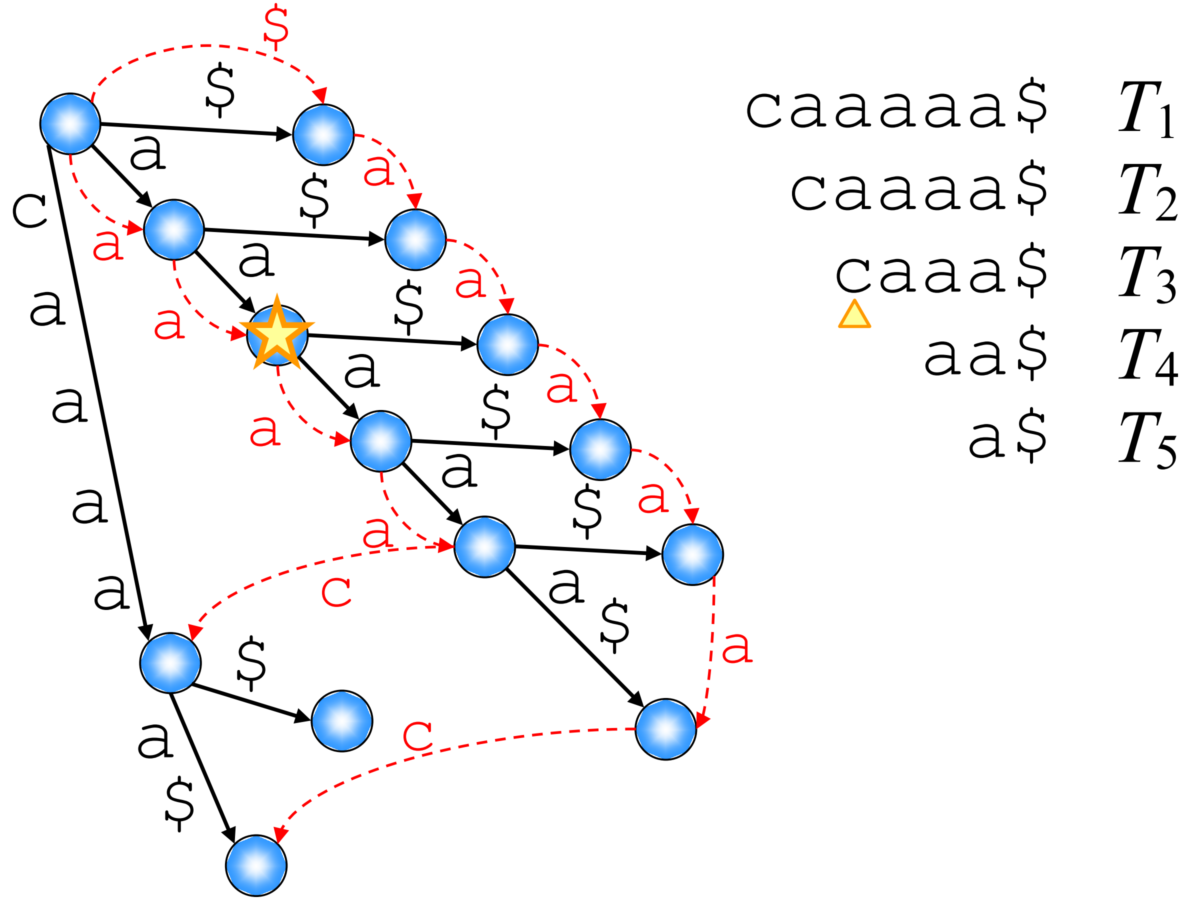
$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1
caaaa$ T_2
caaa$ T_3
aa$ T_4
a$ T_5

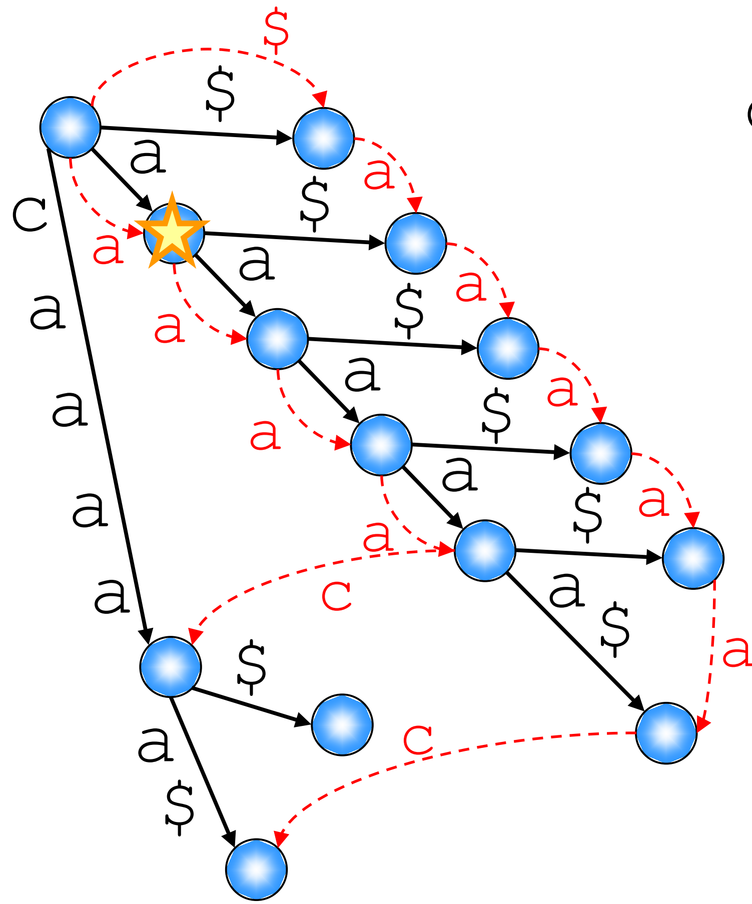
※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明



※ このステップに関係ないリンクは一部省略

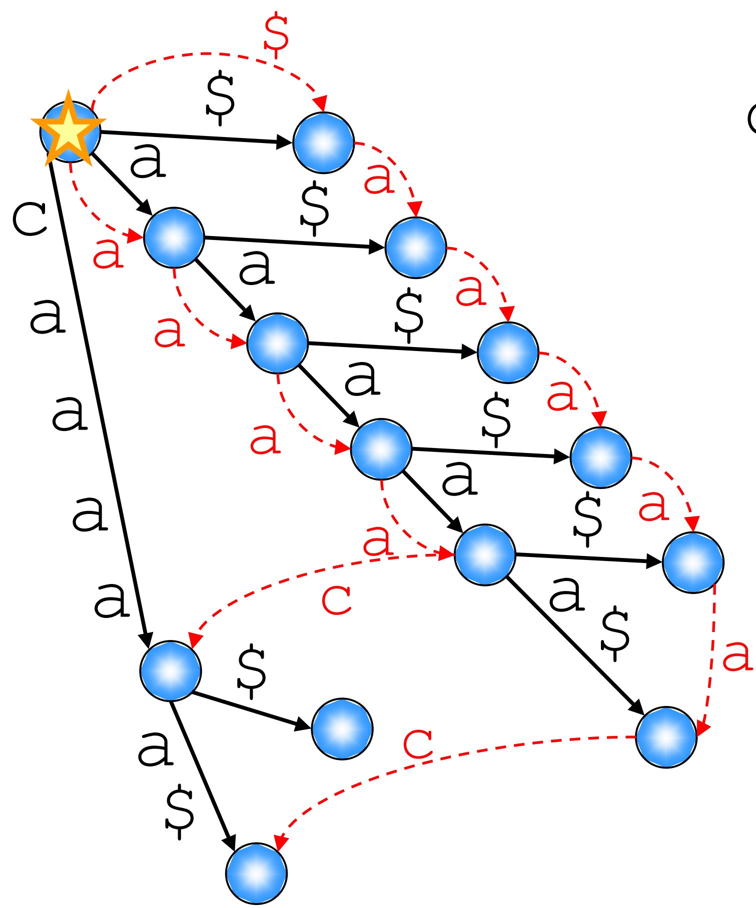
$\Omega(N^{1.5})$ 時間の証明



caaaaa$	T_1
caaaa$	T_2
caaa$	T_3
aa$	T_4
a$	T_5

※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1

caaaa$ T_2

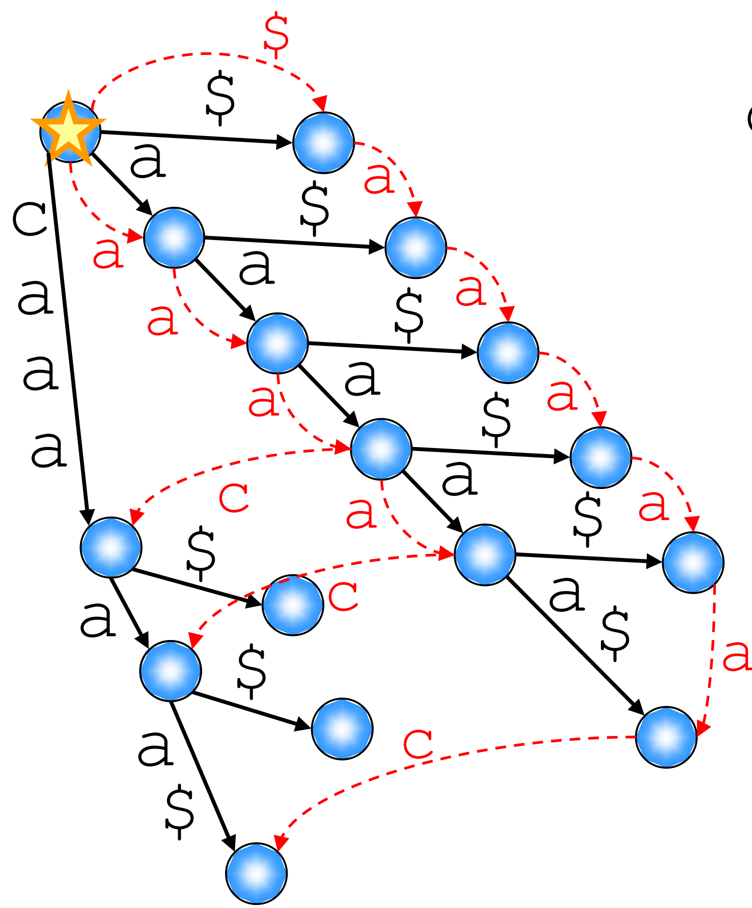
caaa$ T_3

aa$ T_4

a$ T_5

※ このステップに関係ないリンクは一部省略

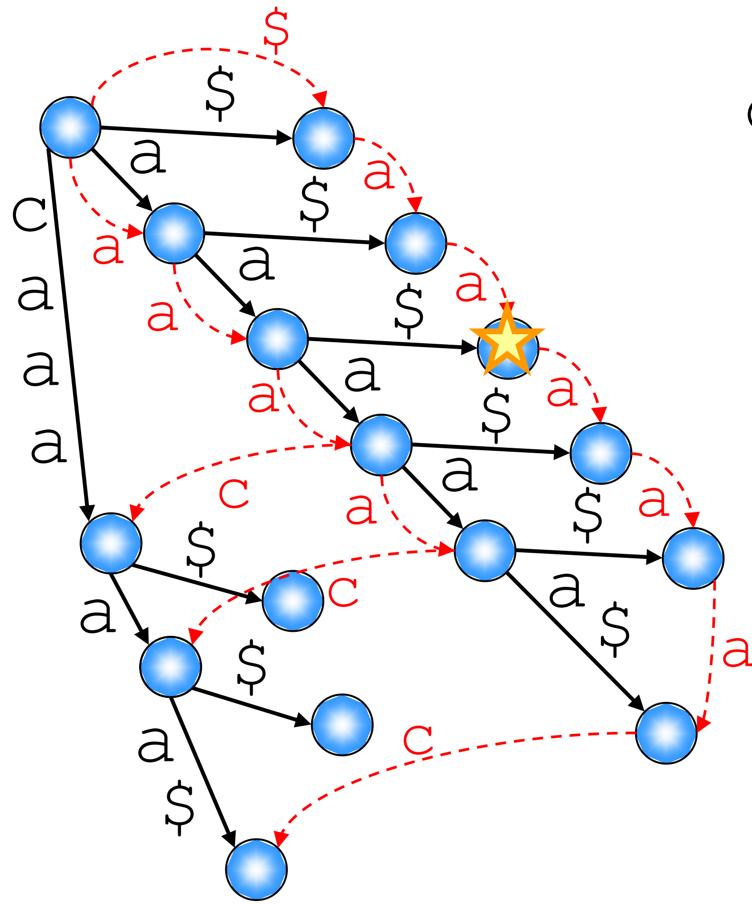
$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1
caaaa$ T_2
caaa$ T_3
aa$ T_4
a$ T_5

※ このステップに関係ないリンクは一部省略

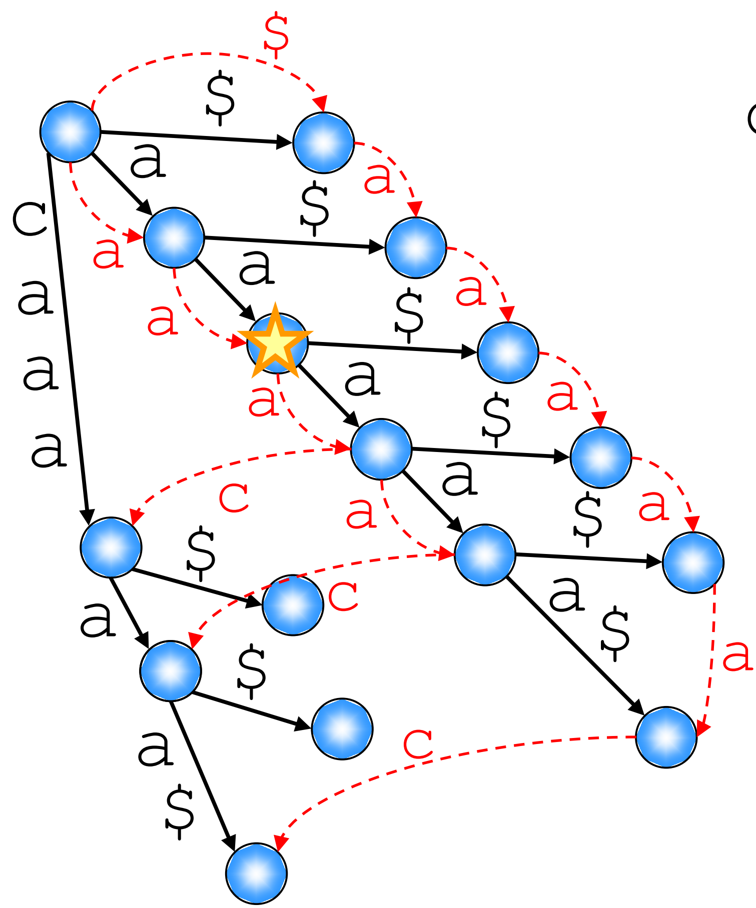
$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1
caaaa\$ T_2
caaa\$ T_3
aa\$ T_4
a\$ T_5

※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1

caaaa$ T_2

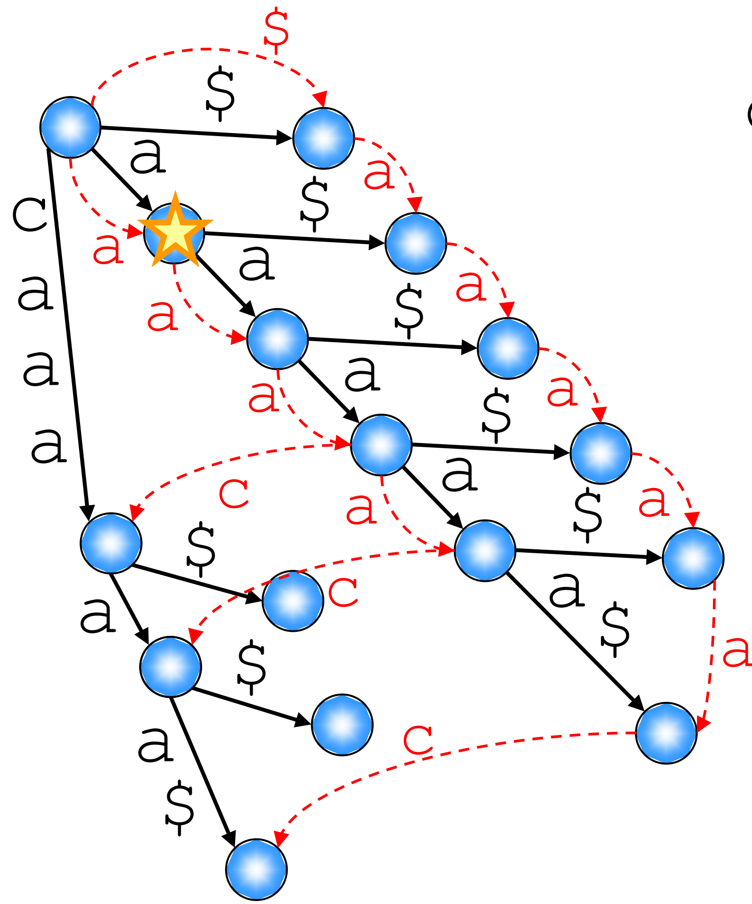
caaa$ T_3

aa$ T_4

a$ T_5

※ このステップに関係ないリンクは一部省略

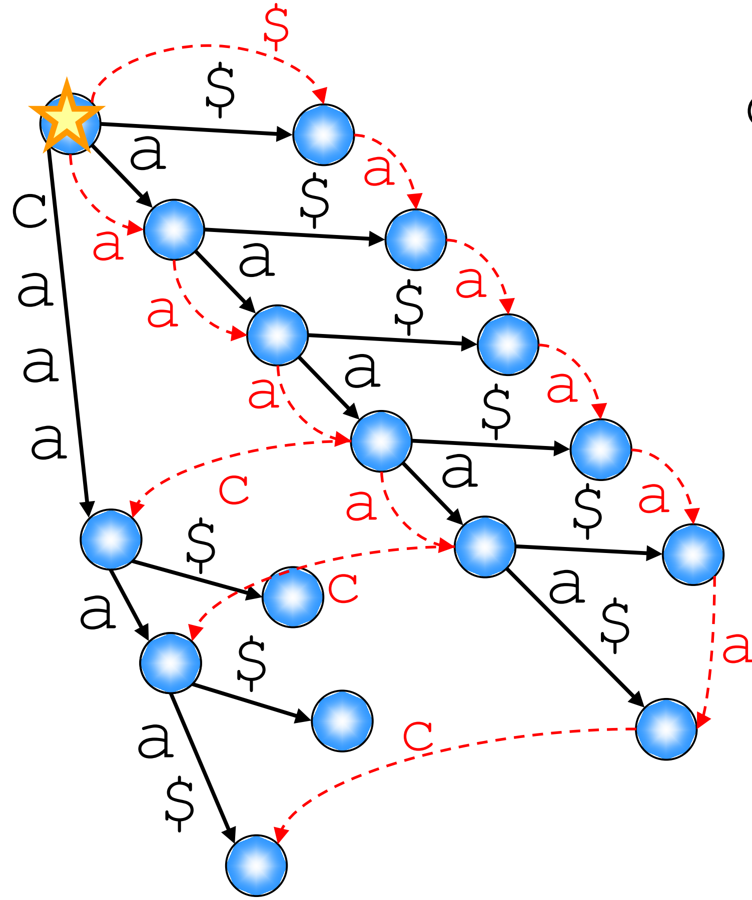
Ω($N^{1.5}$) 時間の証明



caaaaa\$ T_1
caaaa\$ T_2
caaa\$ T_3
aa\$ T_4
a\$ T_5

※ このステップに関係ないリンクは一部省略

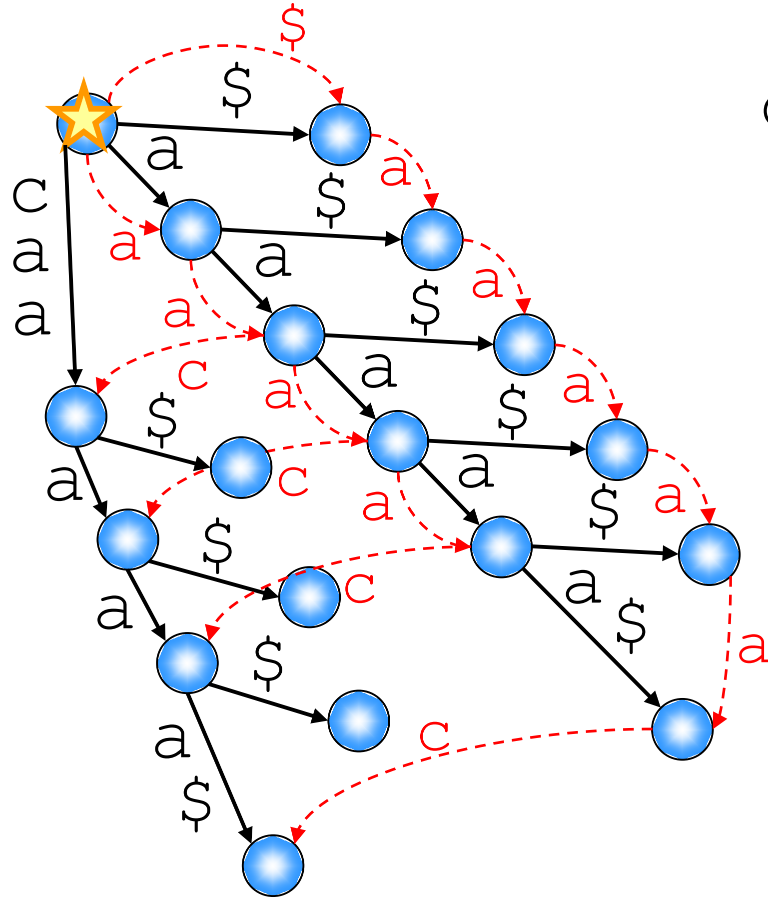
$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1
caaaa\$ T_2
caaa\$ T_3
aa\$ T_4
a\$ T_5

※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1

caaaa\$ T_2

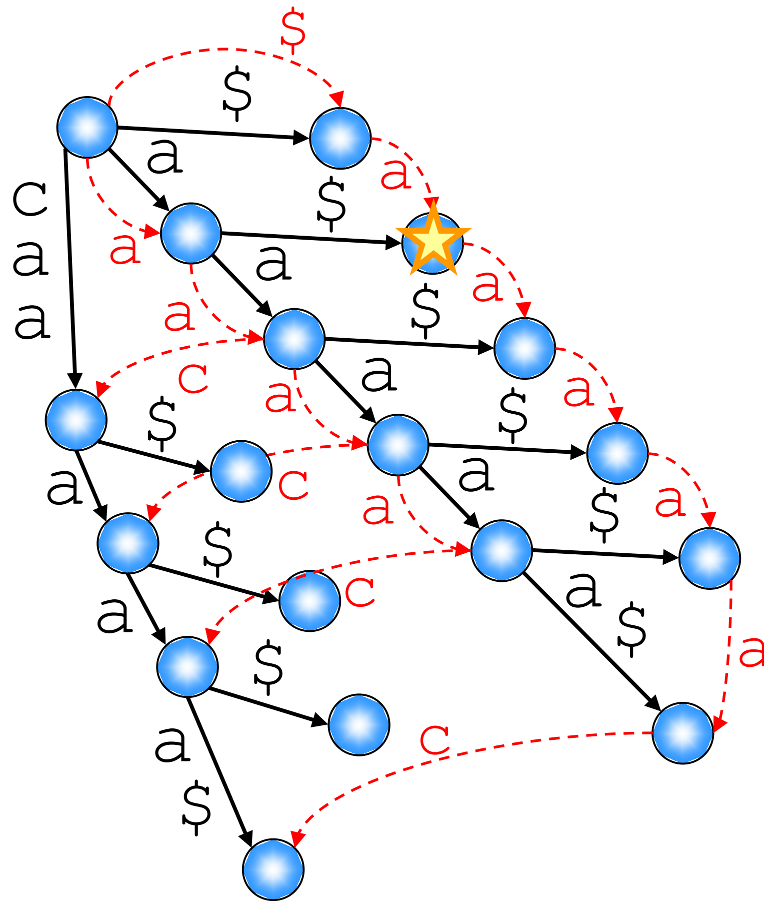
caaa\$ T_3

caa\$ T_4

a\$ T_5

※ このステップに関係ないリンクは一部省略

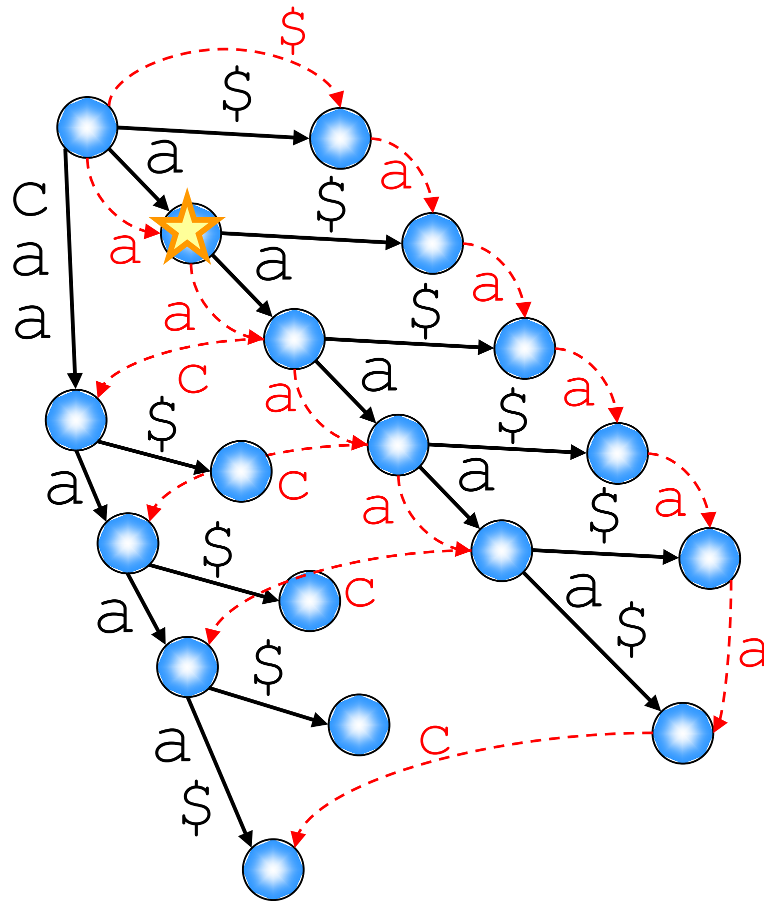
$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1
caaaa\$ T_2
caaa\$ T_3
caa\$ T_4
ca\$ T_5

※ このステップに関係ないリンクは一部省略

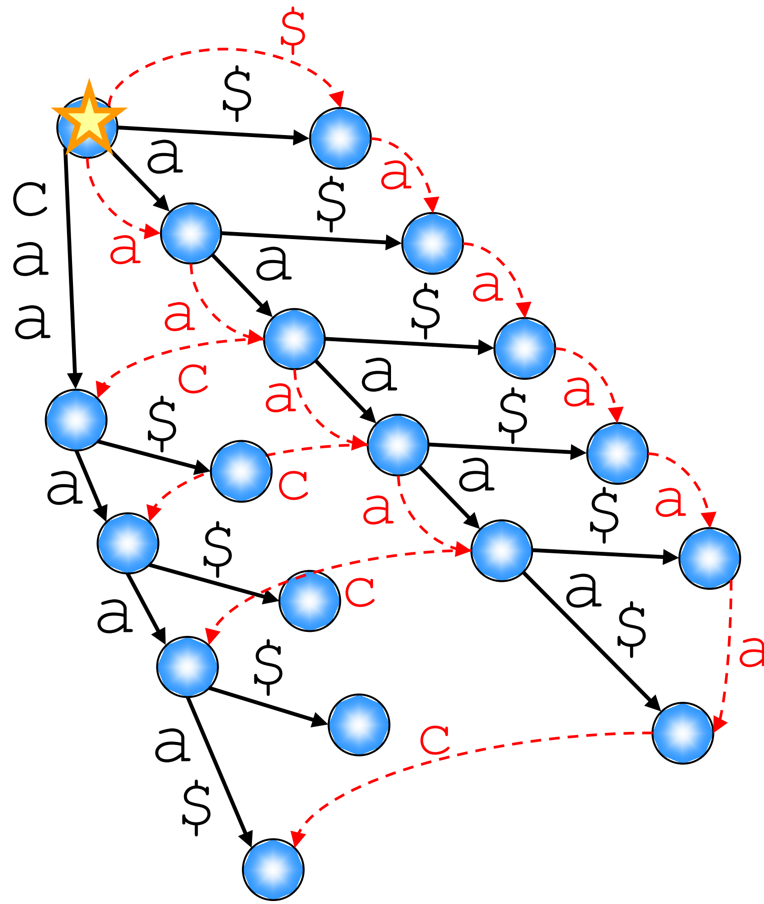
$\Omega(N^{1.5})$ 時間の証明



caaaaa\$ T_1
caaaa\$ T_2
caaa\$ T_3
caa\$ T_4
ca\$ T_5

※ このステップに関係ないリンクは一部省略

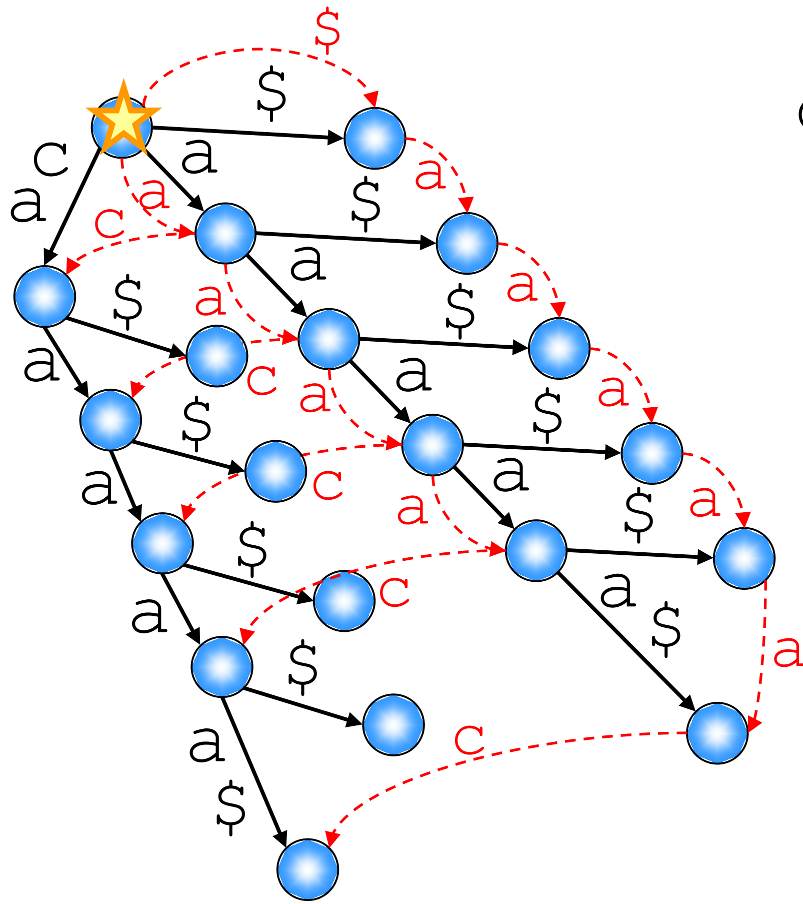
$\Omega(N^{1.5})$ 時間の証明



caaaaa$	T_1
caaaa$	T_2
caaa$	T_3
caa$	T_4
ca$	T_5

※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明



caaaaa$ T_1

caaaa$ T_2

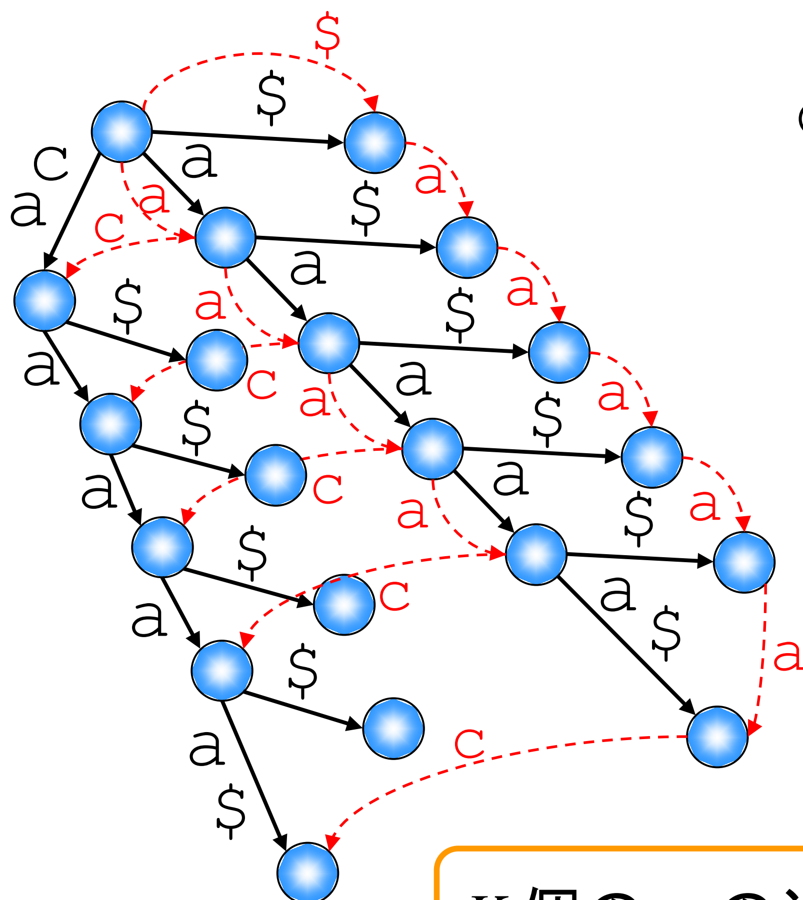
caaa$ T_3

caa$ T_4

ca$ T_5

※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明



K

```
caaaaa$
caaaa$
caaa$
caa$
ca$
```

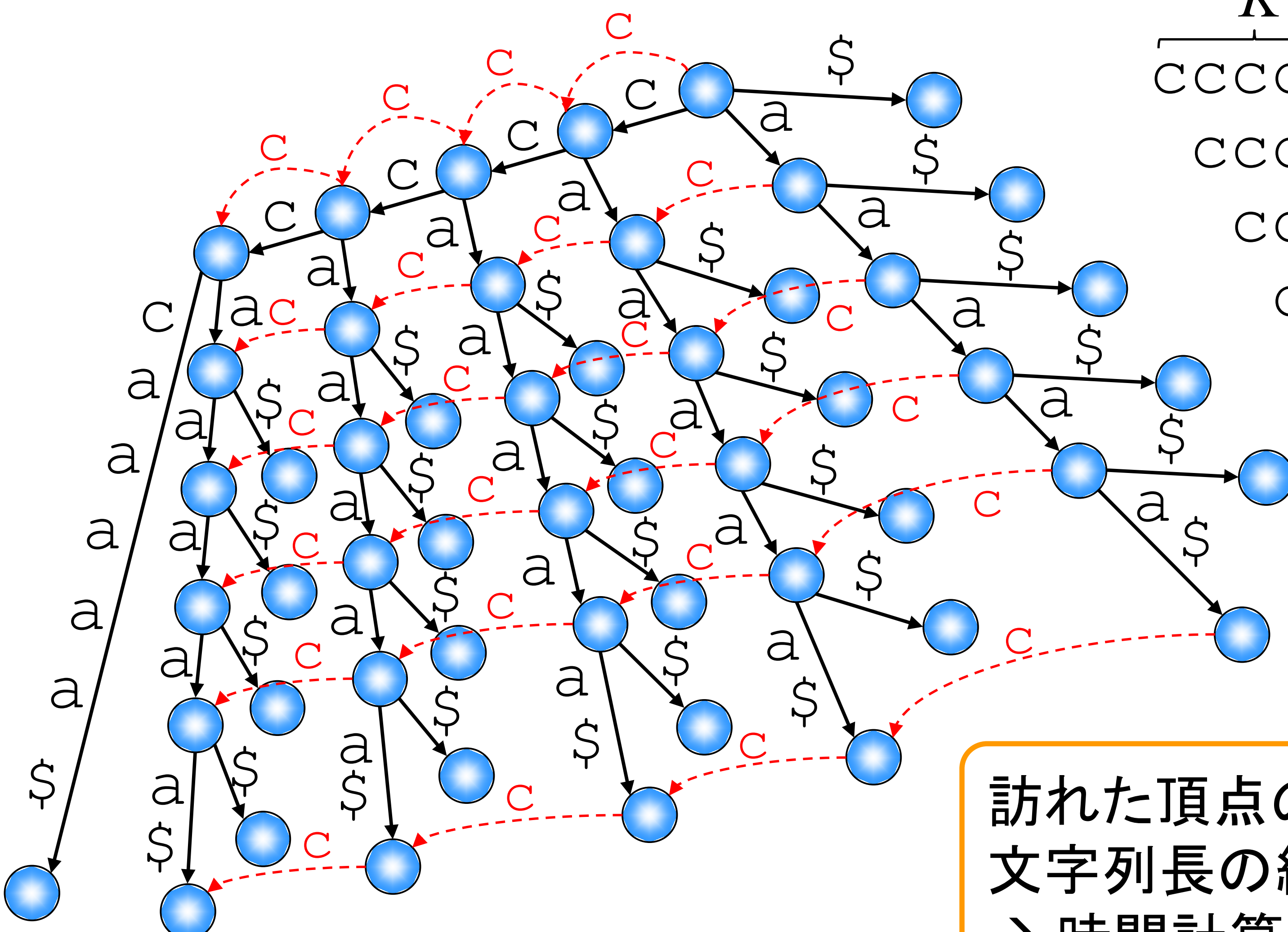
K

K 個の c の追加に対して訪れた頂点数は $\Omega(K^2)$

※ このステップに関係ないリンクは一部省略

$\Omega(N^{1.5})$ 時間の証明

先頭への c の追加を
K ステップ繰り返す



$$\left.\begin{array}{r}\overbrace{\texttt{ccccc}}^{K}\overbrace{\texttt{aaaaa}}^{K}\texttt{\$}\\ \texttt{ccccc}\texttt{aaaa}\texttt{\$}\\ \texttt{ccccc}\texttt{aaa}\texttt{\$}\\ \texttt{ccccc}\texttt{aa}\texttt{\$}\\ \texttt{ccccc}\texttt{a}\texttt{\$}\end{array}\right\} K$$

訪れた頂点の総数 = $\Omega(K^3)$
文字列長の総和 $N = \Theta(K^2)$
→ 時間計算量 = $\Omega(N^{1.5})$

<u>右左</u>オンライン構築〜完全版

定理 2

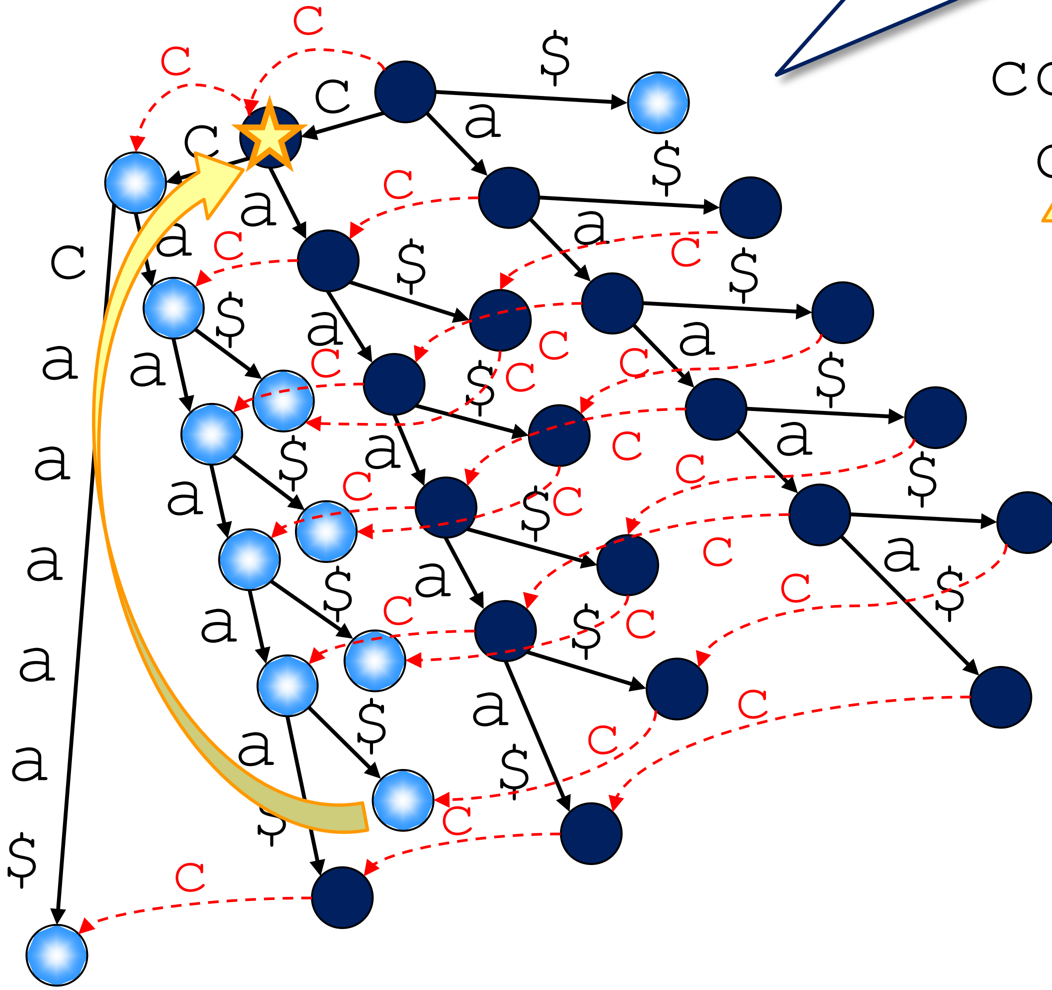
<u>**先頭に**</u>文字を追加していく**複数文字列**の接尾辞木を $O(N \log \sigma)$時間 $O(N)$領域で構築するアルゴリズムが存在する.

【テクニックの概要】

- 1文字追加につき $\Omega(K)$ 個訪れていた頂点を <u>**$O(1)$ 時間でジャンプ**</u>できればよい.
- このジャンプを木の<u>**最近マーク祖先 (NMA) 問題**</u>に帰着.
- NMA のデータ構造を愚直に用いると $O(\sigma N)$ 領域必要だが, 注意深く選んだ頂点集合にのみ用いると <u>**$O(N)$ 領域**</u>に収まる.

NMA による高速化

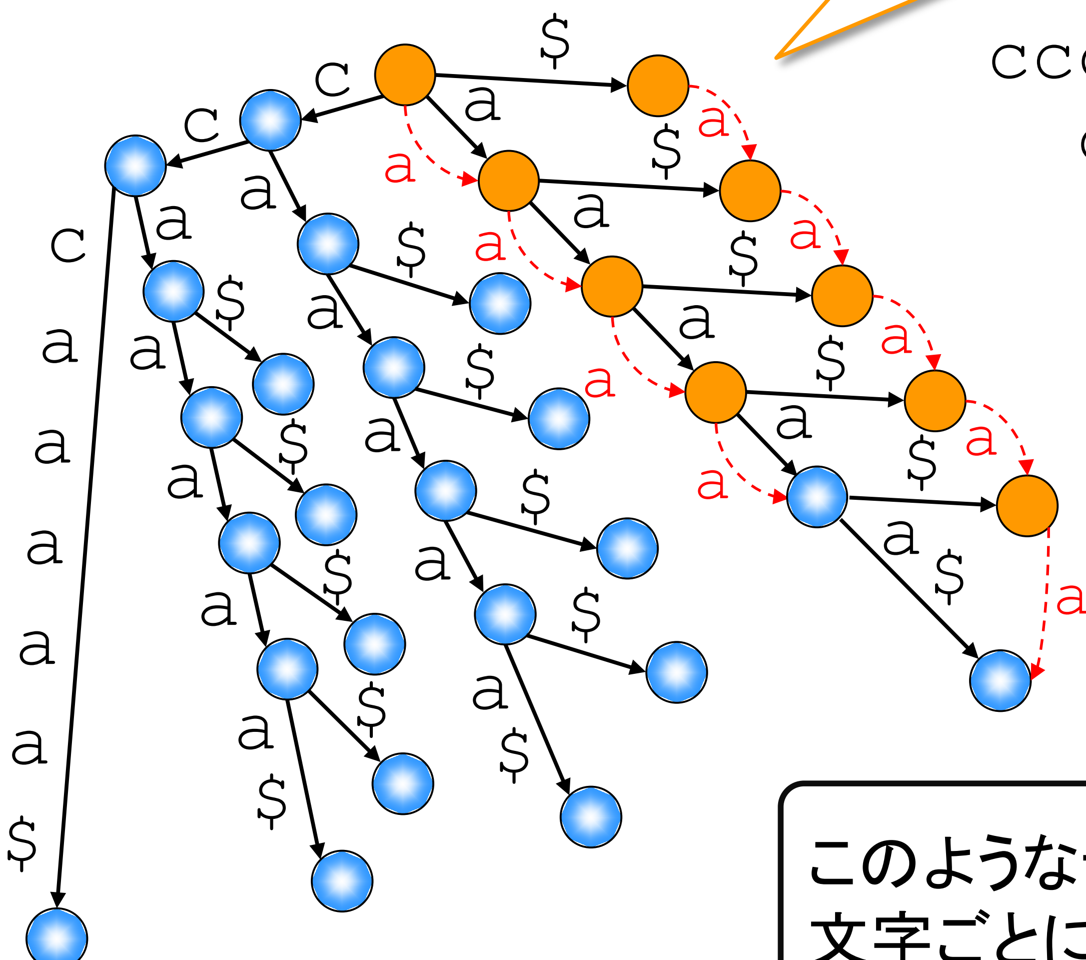
頂点 v がマーク頂点 ●
⇔ v は c のリンクを持つ



cccaaaaaa$	T_1
cccaaaaa$	T_2
△ ccaaaa$	T_3
ccaaa$	T_4
ccaa$	T_5

NMA による高速化

頂点 v がマーク頂点 ●
⇔ v は a のリンクを持つ



cccaaaaaa$	T_1
ccaaaaa$	T_2
ccaaa$	T_3
ccaa$	T_4
cca$	T_5

このようなデータ構造を
<u>文字ごとに</u>構築する

<u>左右</u>オンライン構築

定理 4

Ukkonen のアルゴリズムをそのまま
複数文字列に適用すると <u>$\Omega(N^2)$ 時間かかる</u>.

定理 5

<u>**末尾に**</u>文字を追加していく**複数文字列**の接尾辞木を
$O(N \log \sigma)$時間 $O(N)$領域で構築するアルゴリズムが
存在する.

※ ただし，定理5の接尾辞木では葉のラベルを陽に管理しない.
（葉に達する場合のパターン照合は別の方法で行う）

本研究にまつわるヒストリー

2015/5	研究開始（高木，稲永，有村）
2016/7	文字列処理に関する国際会議 CPM で発表
2017/1	Dany から３人にメールが届き， CPM の論文に致命的なエラーが見つかる
2017/2～3	Dany と Diptarama が研究に参加
2017/7	すべての問題を解決!
2017/12/20	第１版の執筆が完了し，共著者にメール
2017/12/21	Dany が急逝
2018/3	Algorithmica に投稿
2019/10	Algorithmica に採択/掲載

Meeting with Dany @北海道大学 (2017/6）

【Dany Breslauer 氏の功績に敬意を表します】