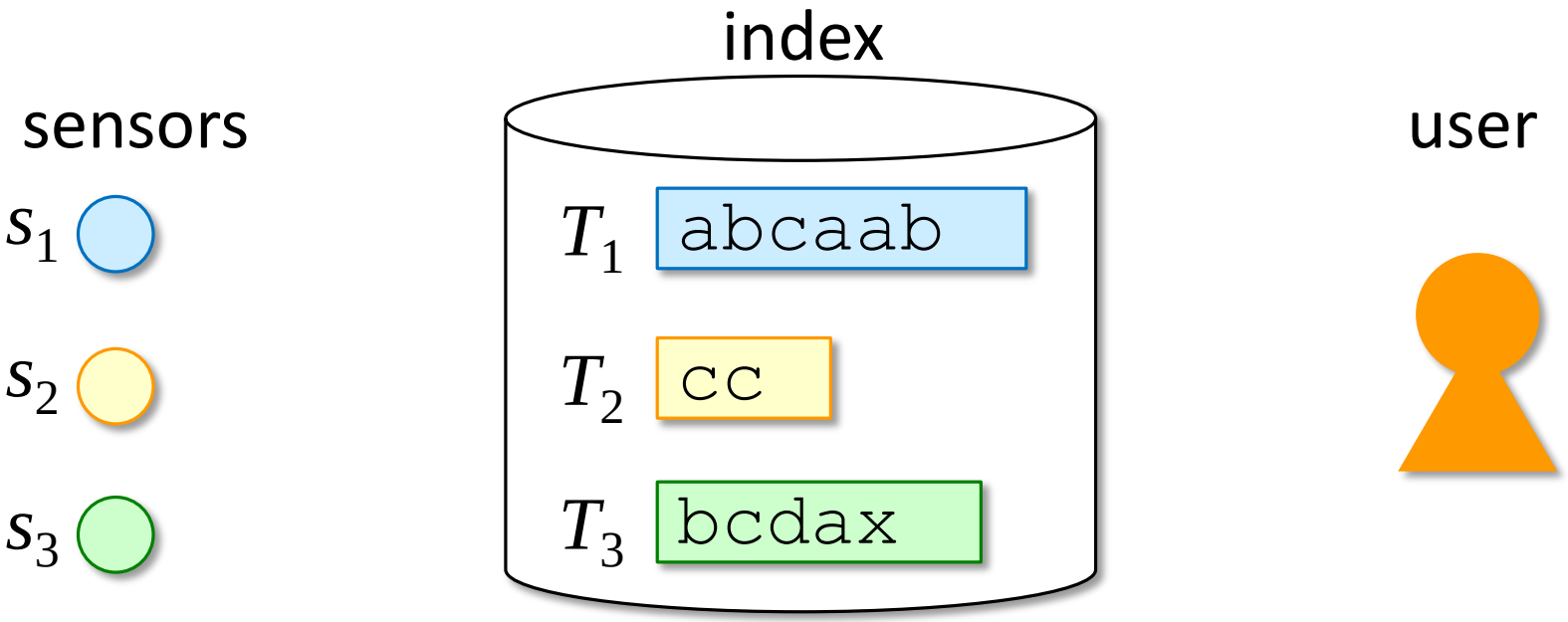
PWA 2018

Fully-online suffix tree and DAWG construction for multiple texts

Shunsuke Inenaga
Kyushu University, Japan

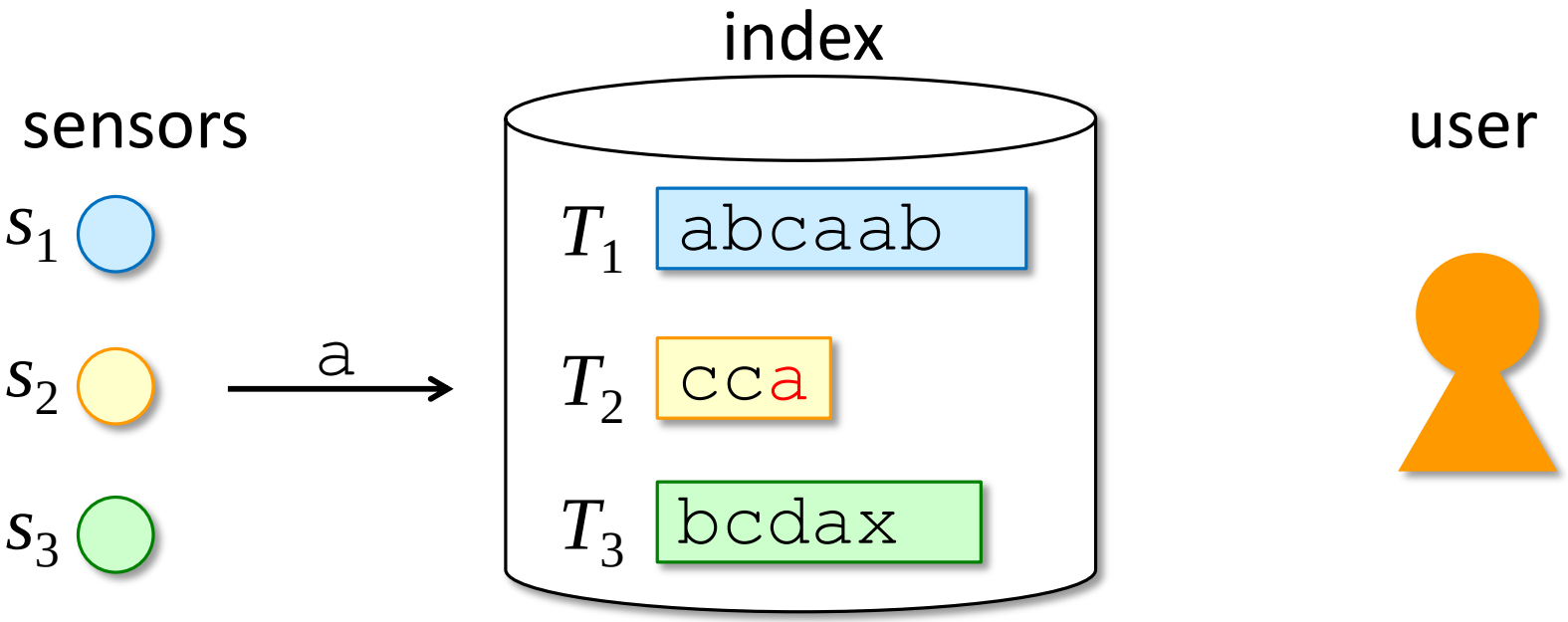
Fully-online index for multi texts

- Goal: Indexing multiple texts in *fully-online manner* where each text can grow *any time*.
- Motivation: Indexing multi online/streaming data.
  - Sensing data, trajectory data, twitter, etc.



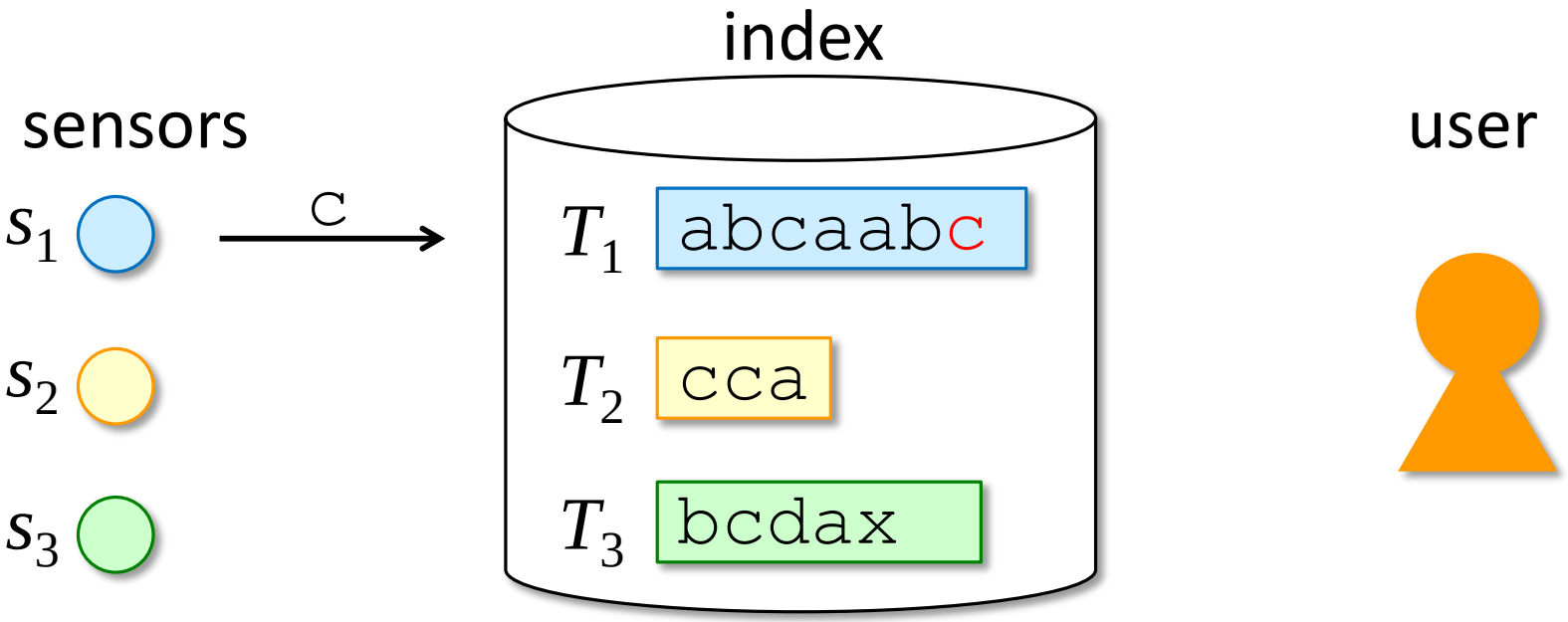
Fully-online index for multi texts

- Goal: Indexing multiple texts in *fully-online manner* where each text can grow *any time*.
- Motivation: Indexing multi online/streaming data.
  - Sensing data, trajectory data, twitter, etc.



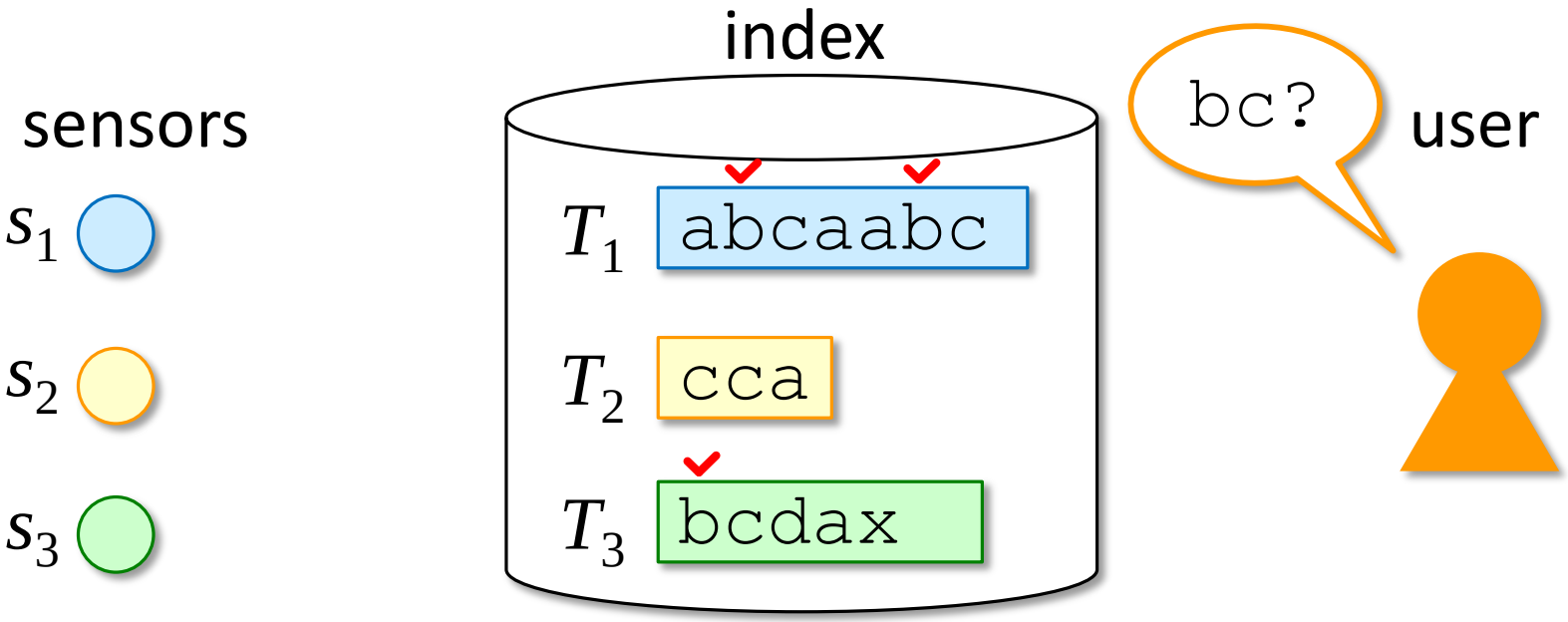
Fully-online index for multi texts

- Goal: Indexing multiple texts in *fully-online manner* where each text can grow *any time*.
- Motivation: Indexing multi online/streaming data.
  - Sensing data, trajectory data, twitter, etc.



Fully-online index for multi texts

- Goal: Indexing multiple texts in *fully-online manner* where each text can grow *any time*.
- Motivation: Indexing multi online/streaming data.
  - Sensing data, trajectory data, twitter, etc.



Original team

… May, 2015

We could extend Ukkonen's algorithm
to fully-online multiple texts.

Takuya Takagi

Hiroki Arimura

Me

Original team

It’s difficult to maintain active points and leaf edge labels for multiple texts…

Takuya Takagi

Hiroki Arimura

Me

Original team

What about Weiner's algorithm?

Takuya Takagi

Hiroki Arimura

Me

Original team

Hmm, it seems that Weiner's right-to-left algorithm can be directly extended to fully-online multiple texts…!

Takuya Takagi

Hiroki Arimura

Me

Original team

Great!
As a bonus, we could also obtain
left-to-right fully-online DAWG construction!

Takuya Takagi

Hiroki Arimura

Me

Original team

Also, DAWG can tell us how to maintain active points for Ukkonen's left-to-right suffix tree.

Takuya Takagi

Hiroki Arimura

Me

Original team

… July, 2015

Excellent!!

Let’s leave maintenance of leaf edge labels as open question and write paper!

Takuya Takagi

Hiroki Arimura

Me

Our CPM 2016 paper…

Claim 1

Suffix tree of multiple texts of total length N can be built in *right-to-left* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

Claim 2

DAWG (suffix automaton) of multiple texts of total length N can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

σ is the alphabet size

Our CPM 2016 paper…

Claim 3

Suffix tree of multiple texts of total length N *without leaf edge labels* can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space, with the aid of DAWG.

σ is the alphabet size

Danny read our CPM 2016 paper

… January, 2017

Hi, I read your paper, and I am afraid that your results might be wrong…

Danny Breslauer

Oh really?

Takuya Takagi Hiroki Arimura Me

Danny read our CPM 2016 paper

Yes, because you overlooked some operations that can take ***super linear*** time for fully-online multiple texts.

Danny Breslauer

OMG…

Takuya Takagi

Hiroki Arimura

Me

Our CPM 2016 paper was WRONG!

Claim 1

Suffix tree of multiple texts of total length N can be built in *right-to-left* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

Claim 2

DAWG (suffix automaton) of multiple texts of total length N can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

σ is the alphabet size

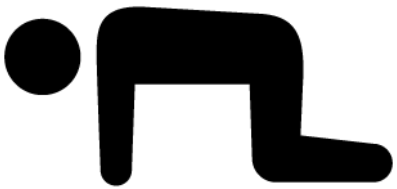
Our CPM 2016 paper was WRONG!

Claim 3

Suffix tree of multiple texts of total length N *without leaf edge labels* can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space, with the aid of DAWG.

σ is the alphabet size

All main claims were wrong…



New team!

… February, 2017

Let’s fix the paper!

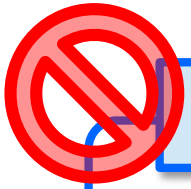
Takuya Takagi

Hiroki Arimura

Me

Danny Breslauer

Right-to-left suffix tree



Claim 1

Suffix tree of multiple texts of total length N can be built in *right-to-left* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

σ is the alphabet size

Theorem 1

Weiner's right-to-left suffix tree algorithm for fully-online multiple texts needs to visit $\Omega(N^{1.5})$ nodes, and this bound is tight ($O(N^{1.5})$ in the worst case).

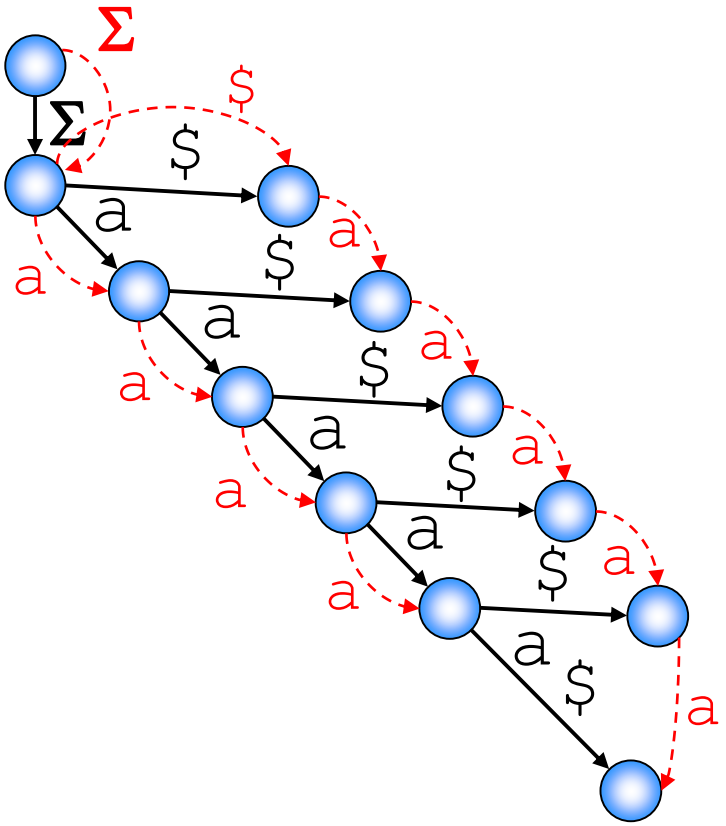
Fully-online Weiner visits $\Omega(N^{1.5})$ nodes

aaaaa$$_1$ T_1
aaaa$$_2$ T_2
aaa$$_3$ T_3
aa$$_4$ T_4
a$$_5$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes

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

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



aaaaa$ T_1

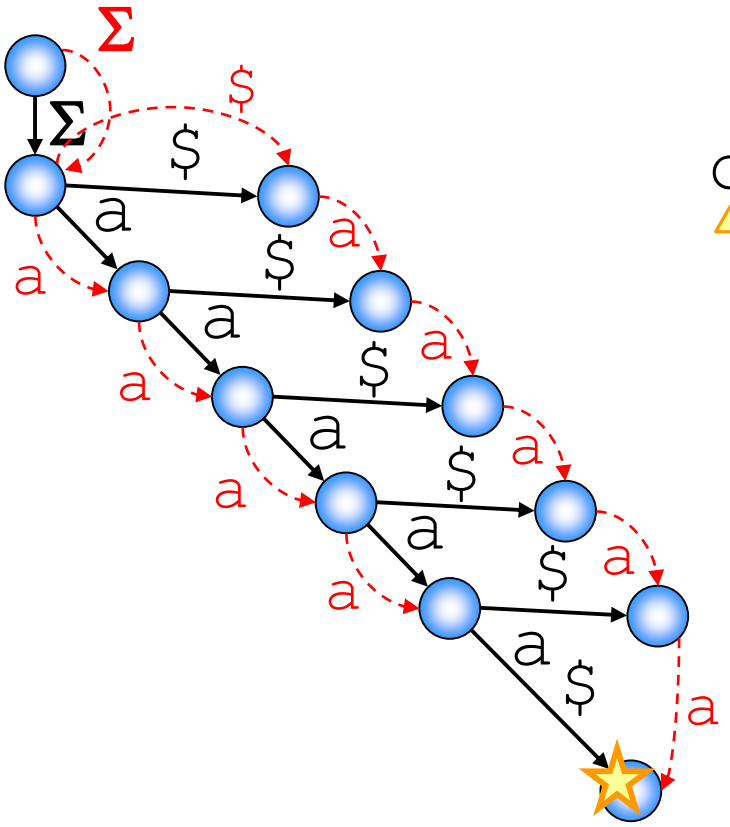
aaaa$ T_2

aaa$ T_3

aa$ T_4

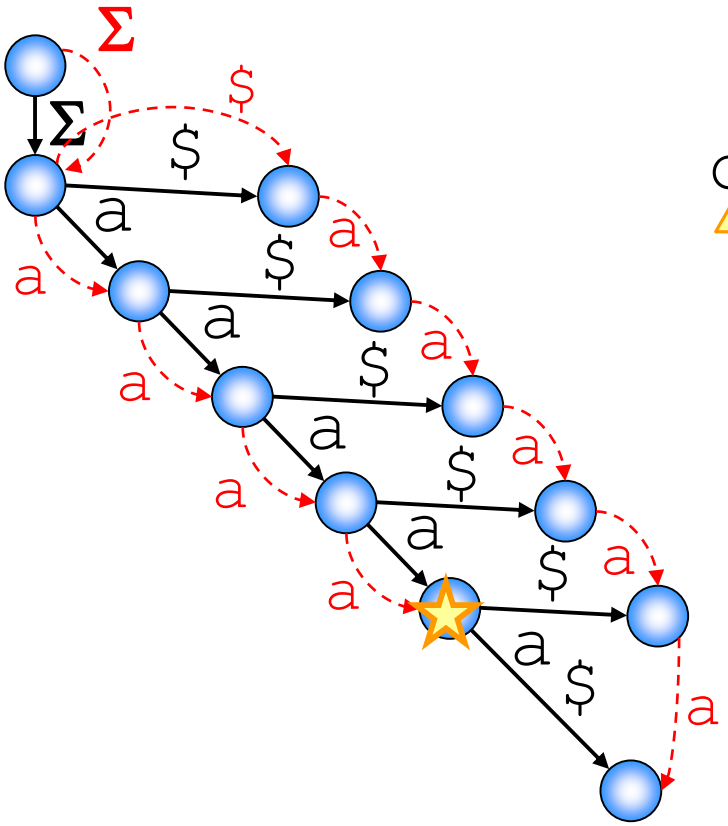
a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



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

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

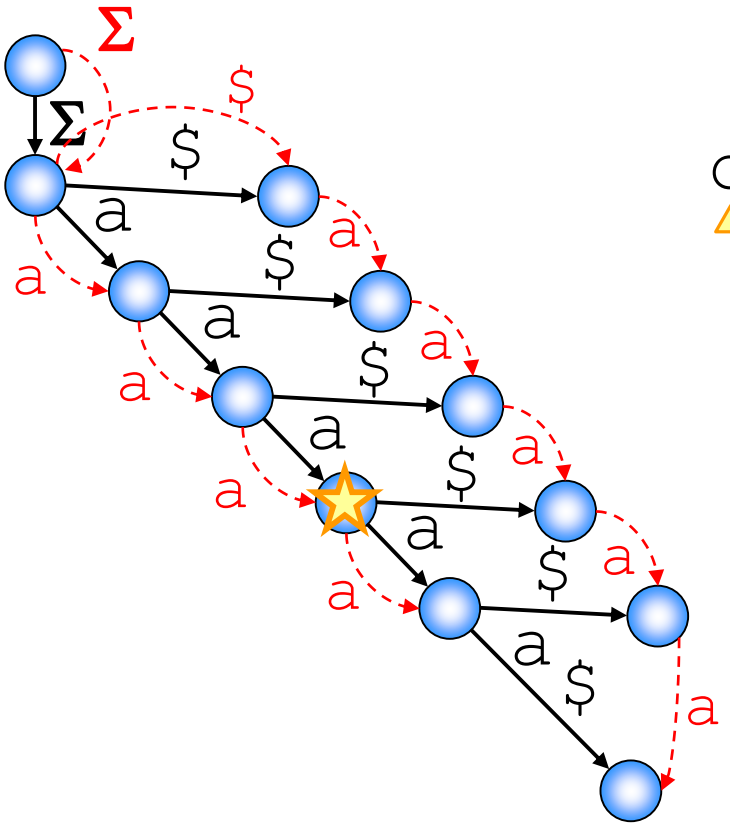
aaaa$ T_2

aaa$ T_3

aa$ T_4

a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

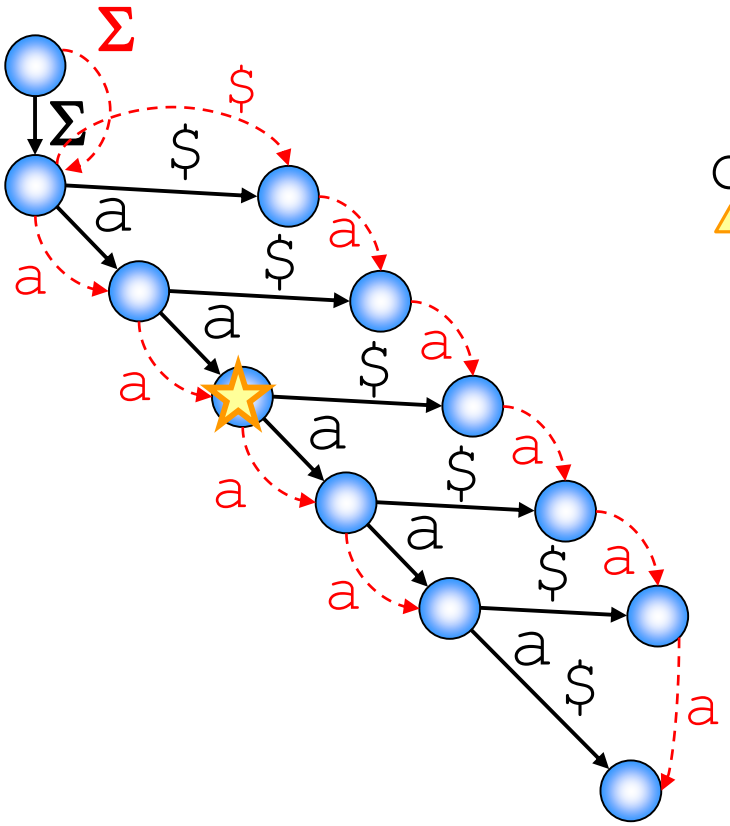
aaaa$ T_2

aaa$ T_3

aa$ T_4

a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

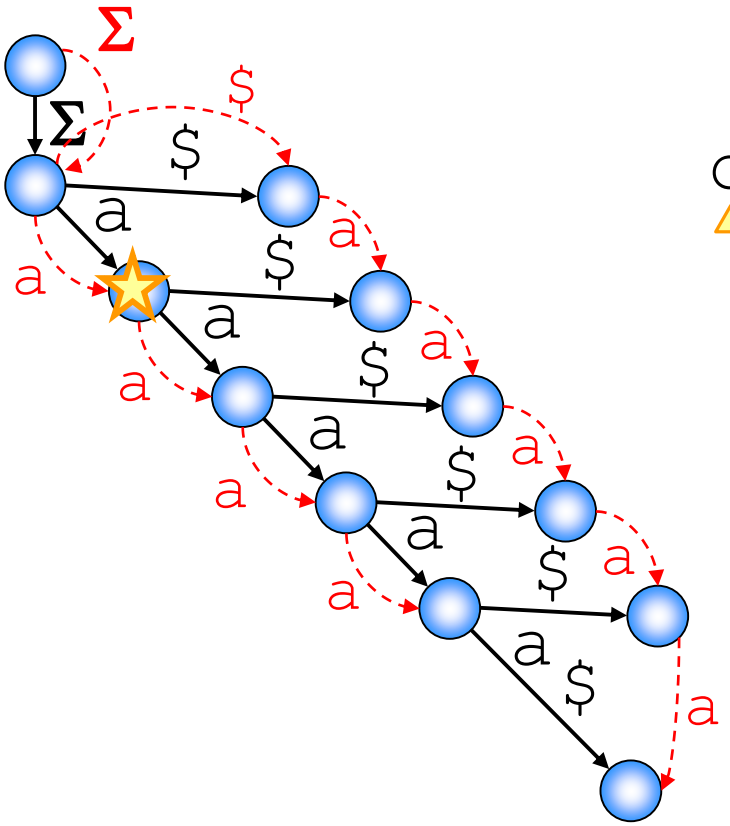
aaaa$ T_2

aaa$ T_3

aa$ T_4

a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

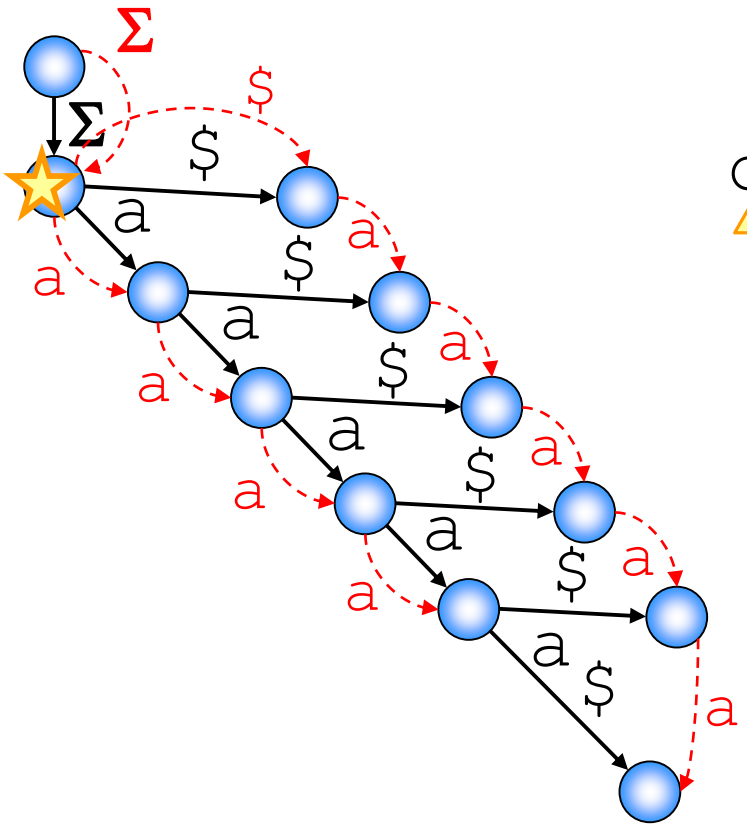
aaaaa$ T_2

aaaa$ T_3

aaa$ T_4

aa$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

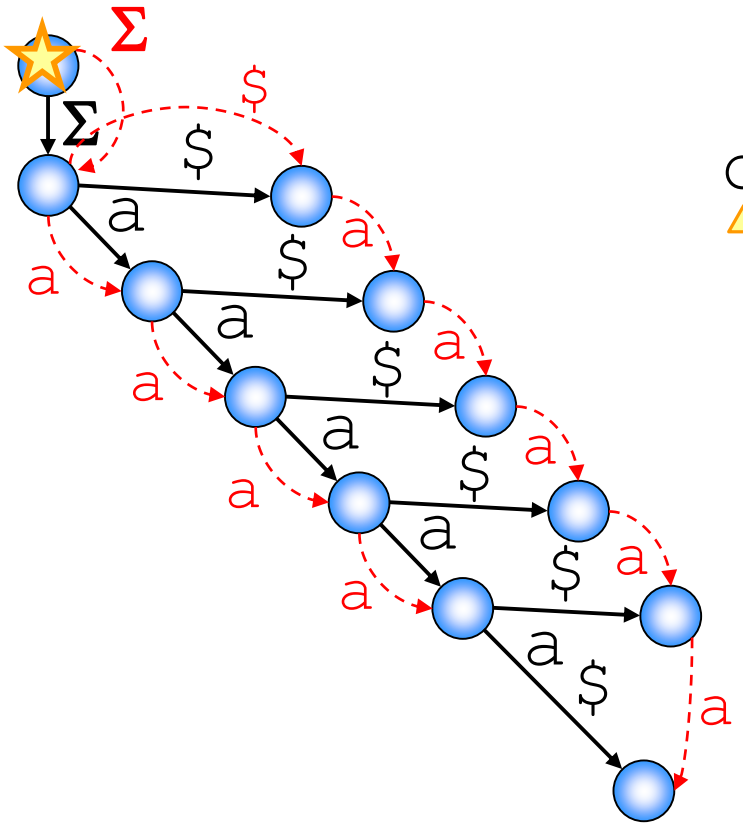
aaaaa$ T_2

aaaa$ T_3

aaa$ T_4

aa$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

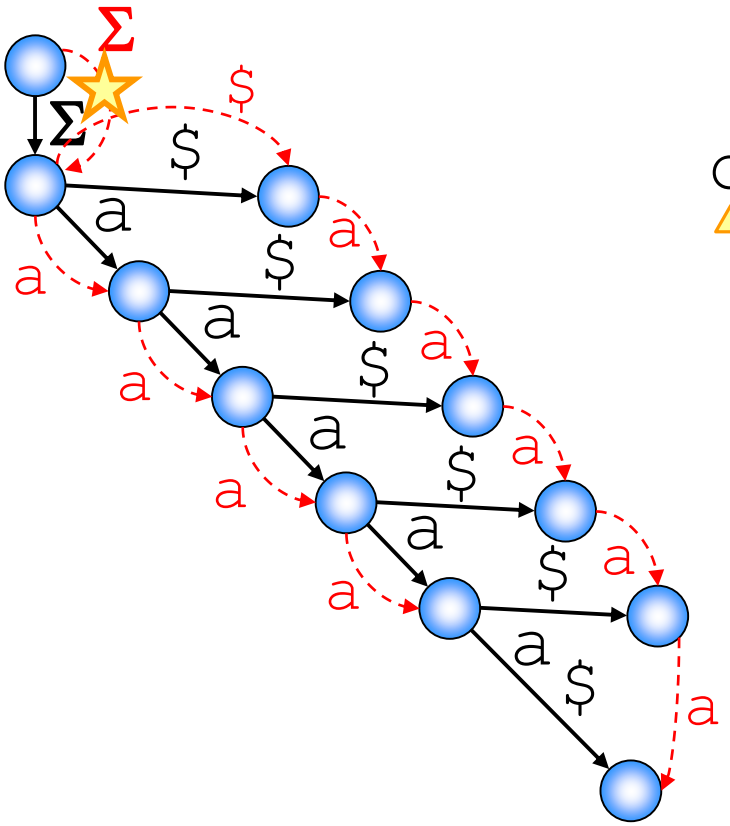
aaaa$ T_2

aaa$ T_3

aa$ T_4

a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

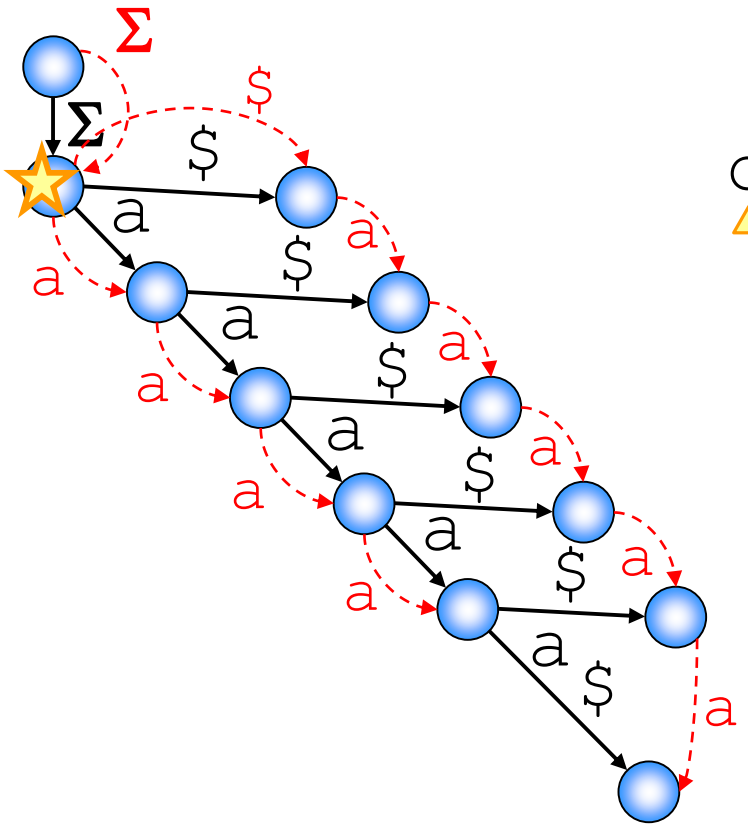
aaaa$ T_2

aaa$ T_3

aa$ T_4

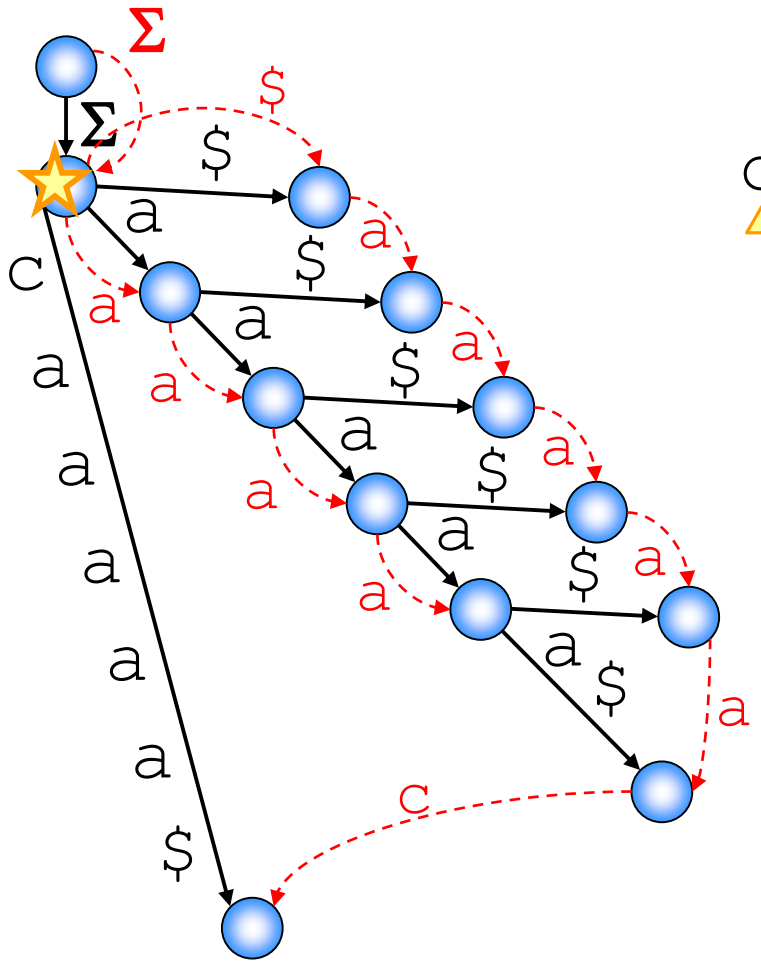
a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



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

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaaa$ T_1

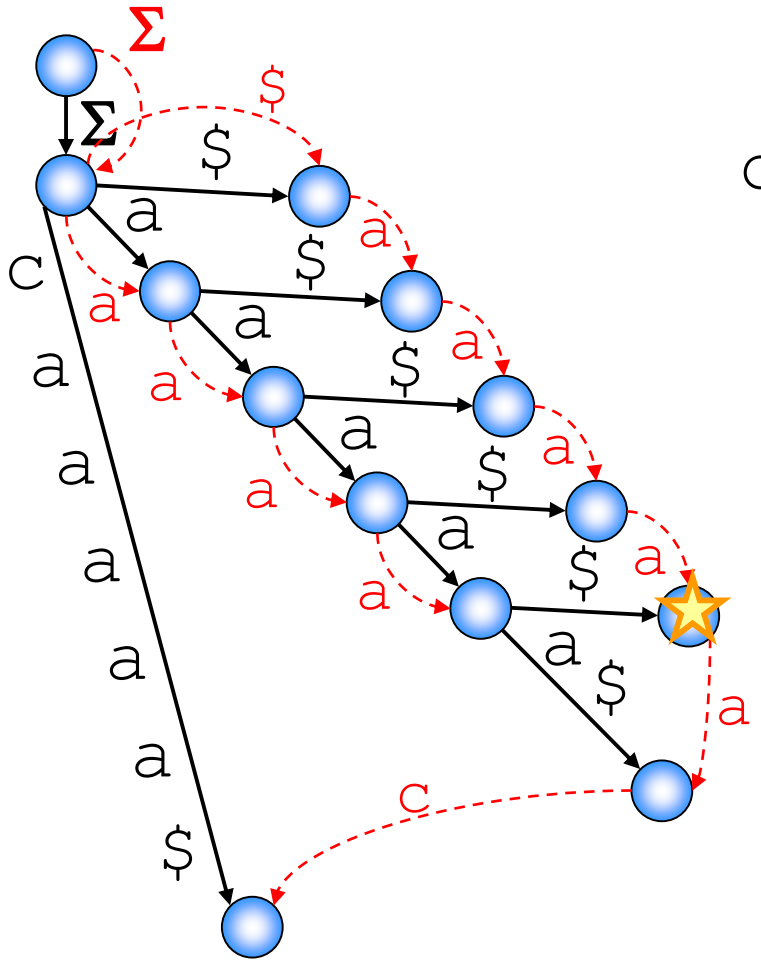
aaaa$ T_2

aaa$ T_3

aa$ T_4

a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

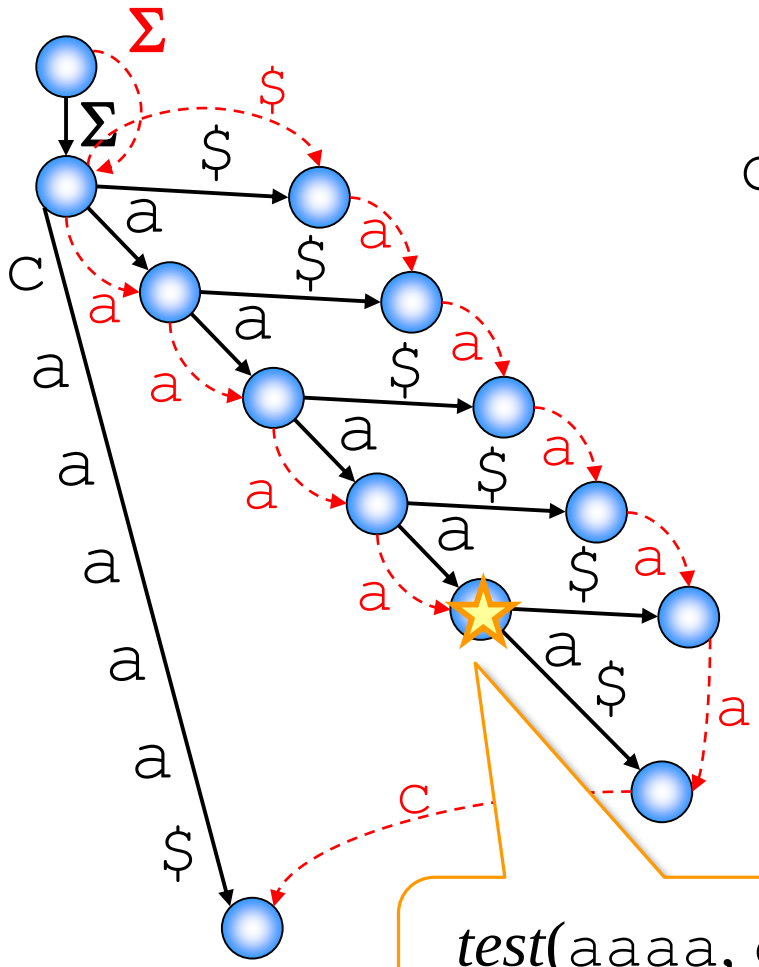
caaaa$ T_2

aaa$ T_3

aa$ T_4

a$ T_5

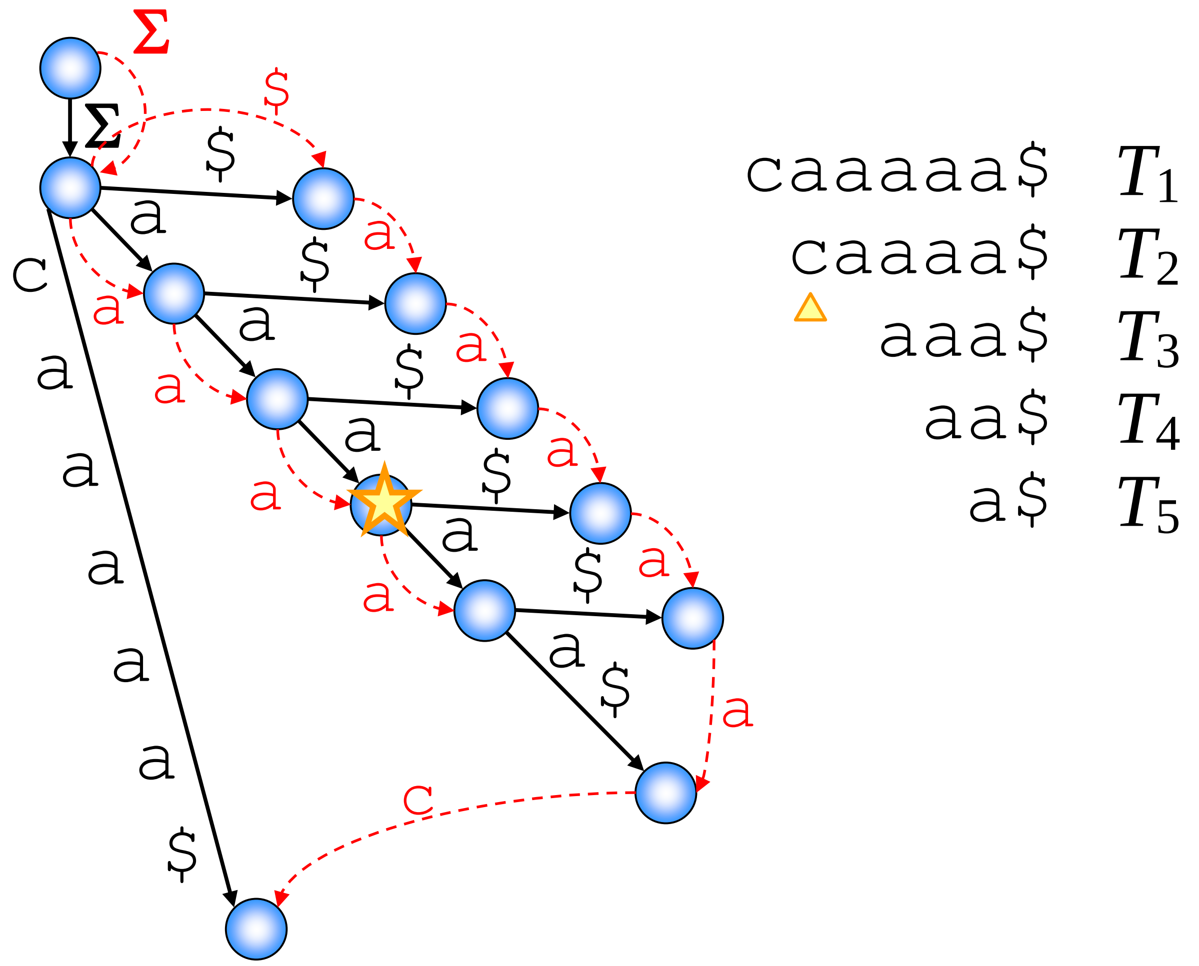
Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



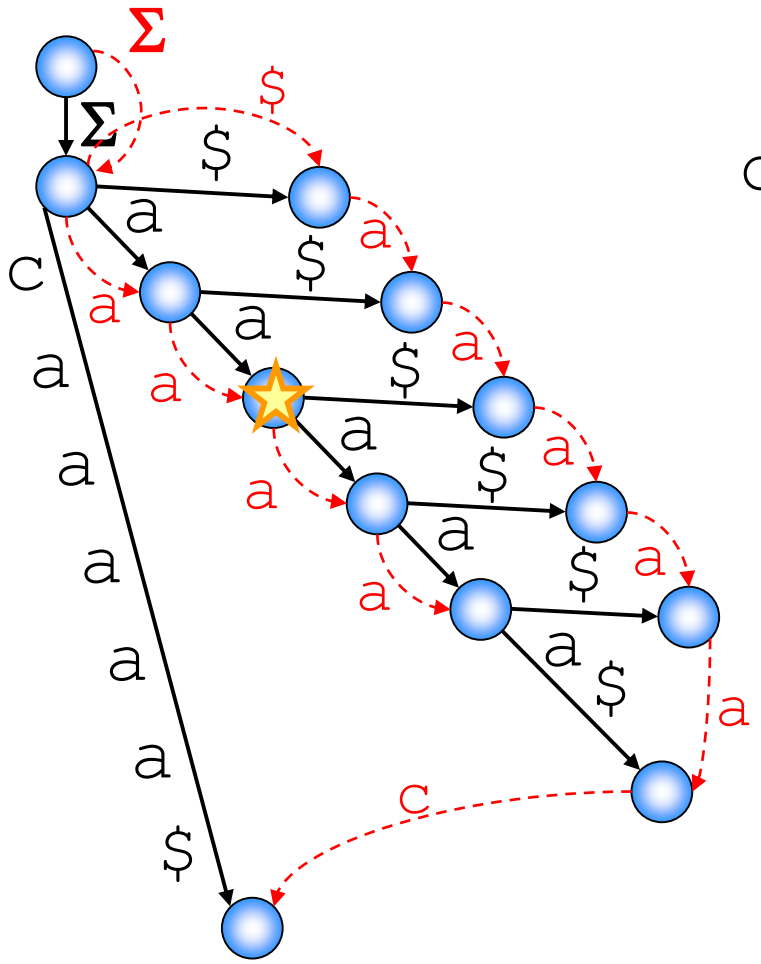
caaaaa$ T_1
caaaa$ T_2
aaa$ T_3
aa$ T_4
a$ T_5

test(aaaa, c) = true
⇔ caaaa exists

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

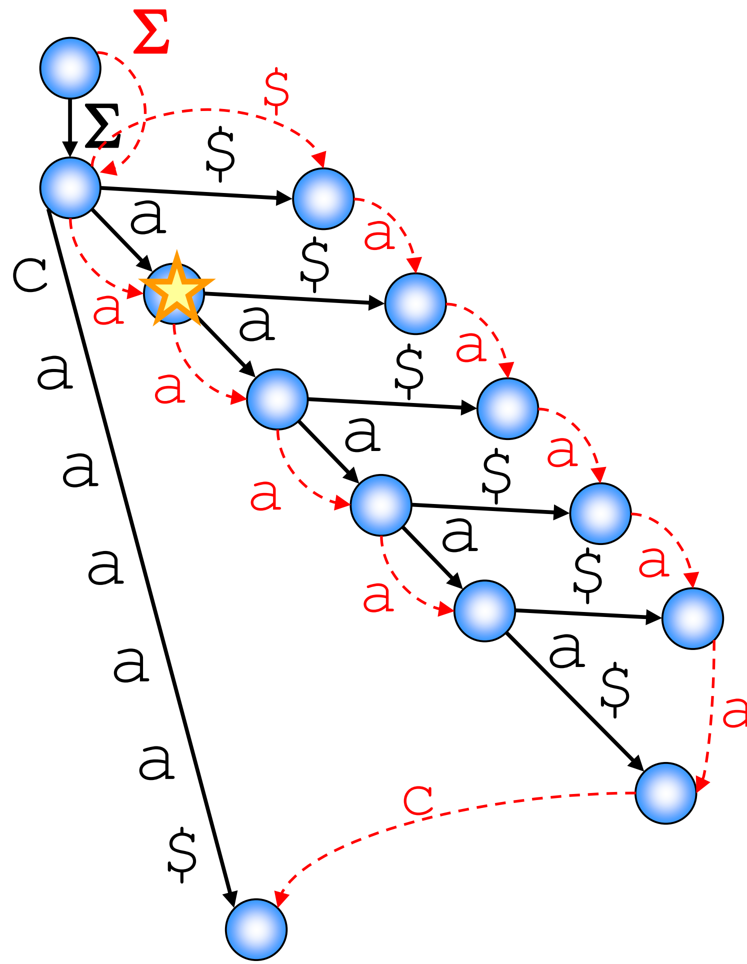
caaaa$ T_2

aaa$ T_3

aa$ T_4

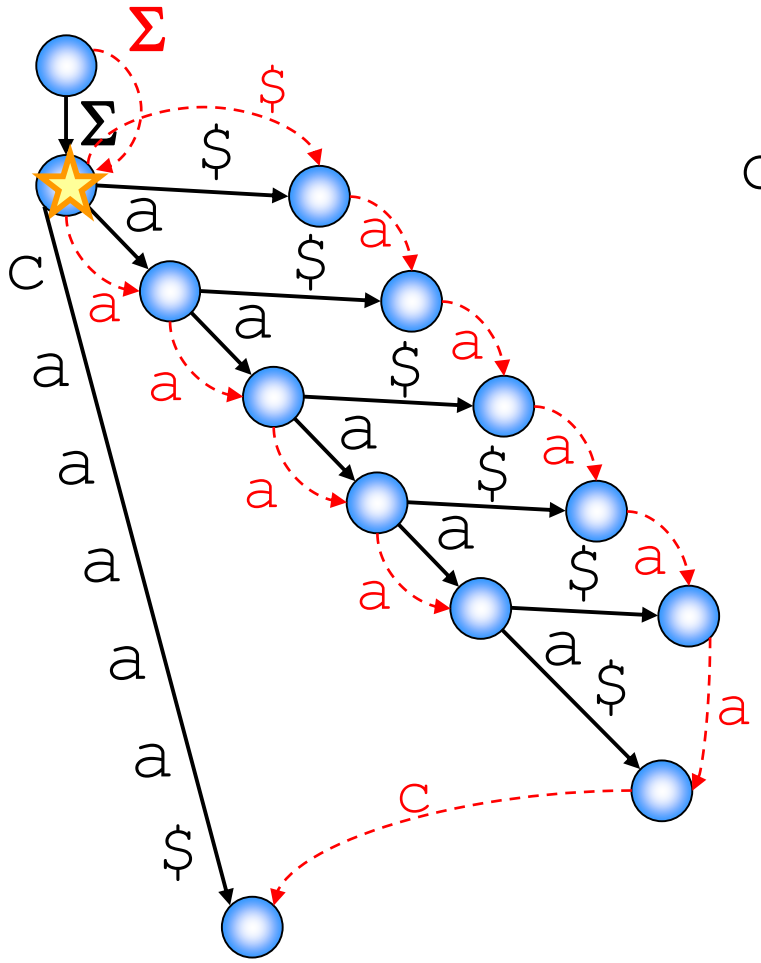
a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



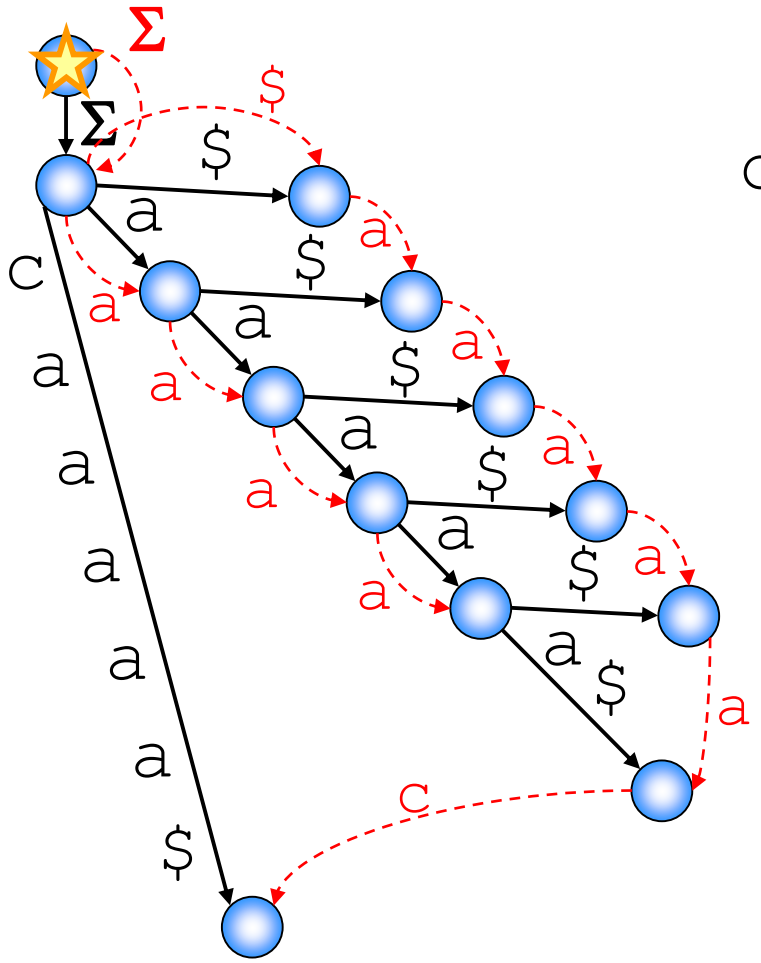
`caaaaa$`	T_1
`caaaa$`	T_2
△ `aaa$`	T_3
`aa$`	T_4
`a$`	T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



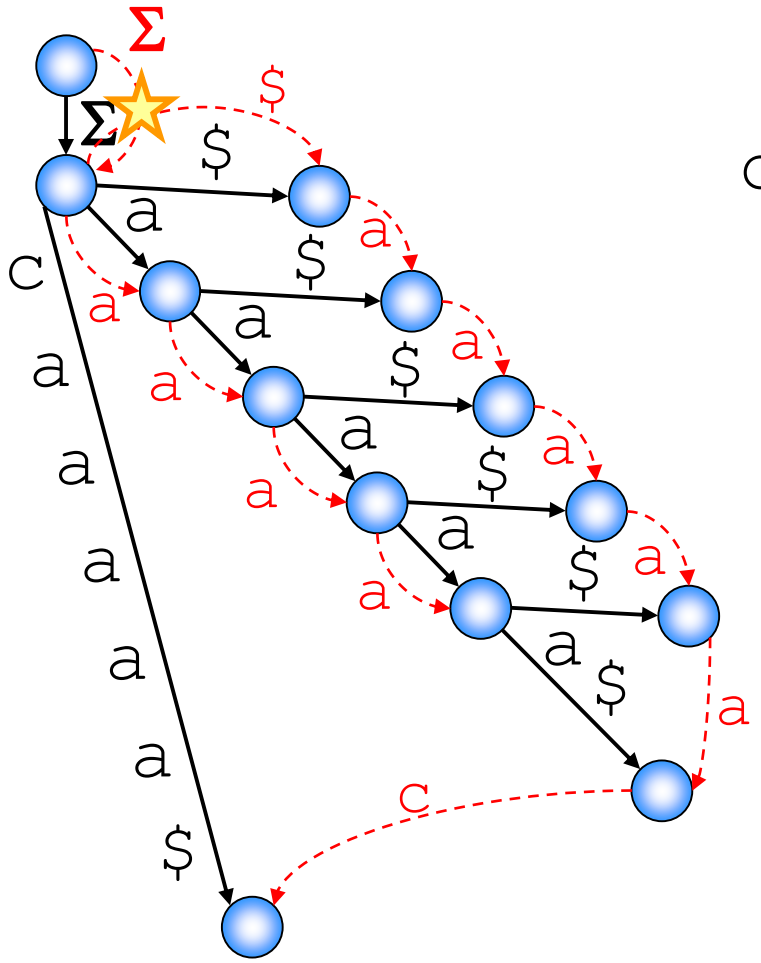
caaaaa$ T_1
caaaa$ T_2
aaa$ T_3
aa$ T_4
a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



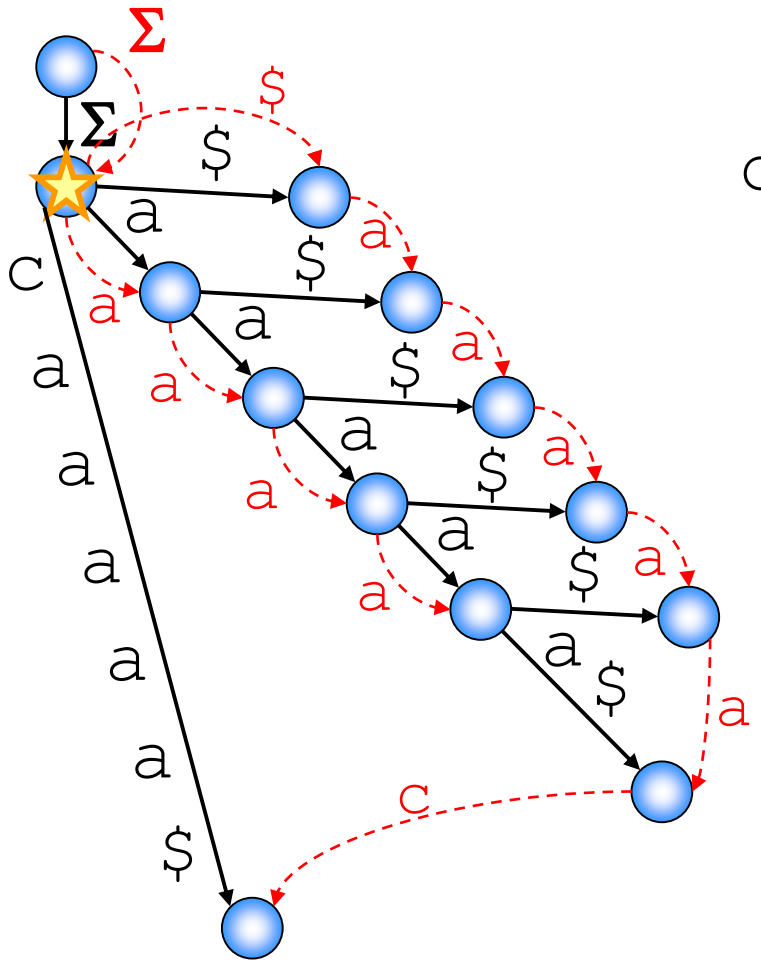
caaaaa$ T_1
caaaa$ T_2
aaa$ T_3
aa$ T_4
a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



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

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

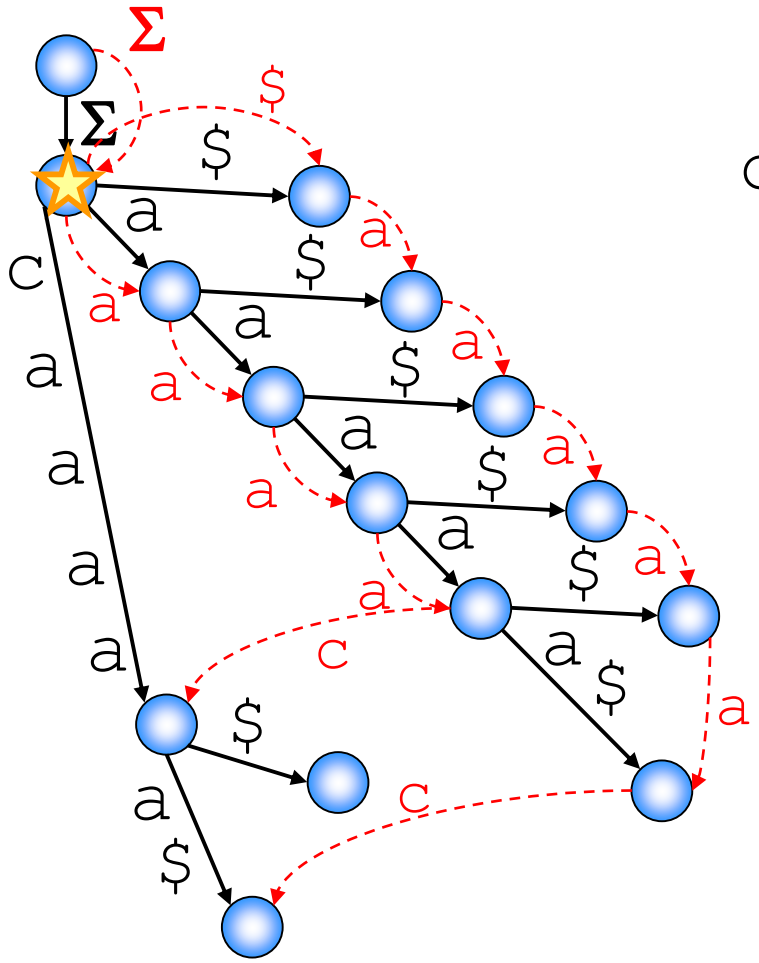
caaaa$ T_2

aaa$ T_3

aa$ T_4

a$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

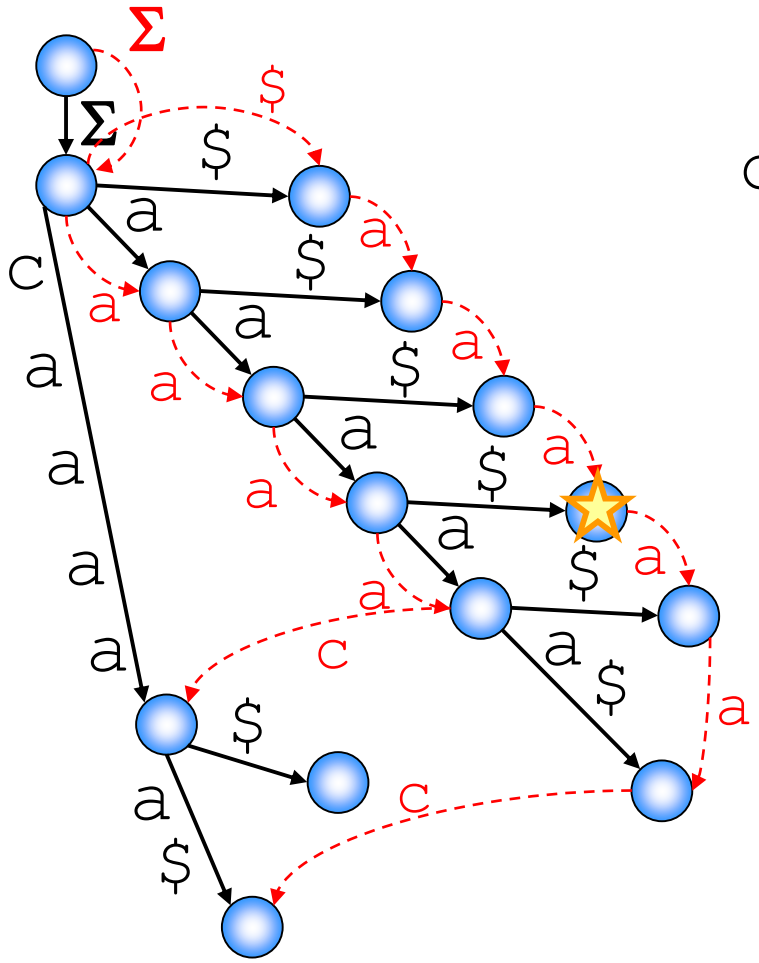
△ aaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

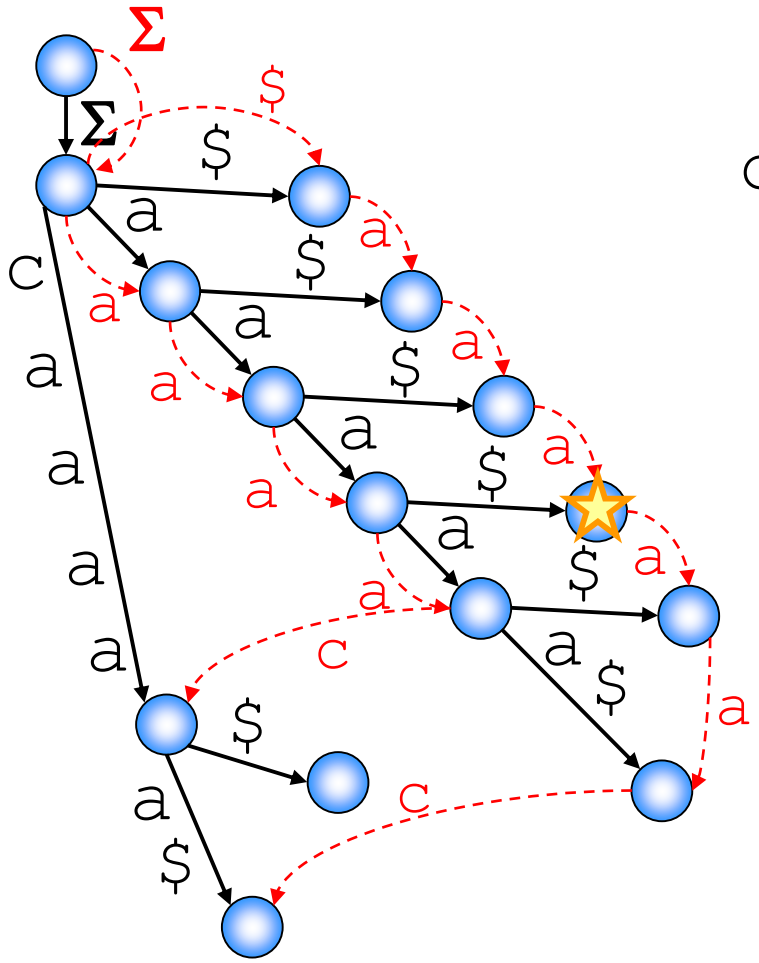
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

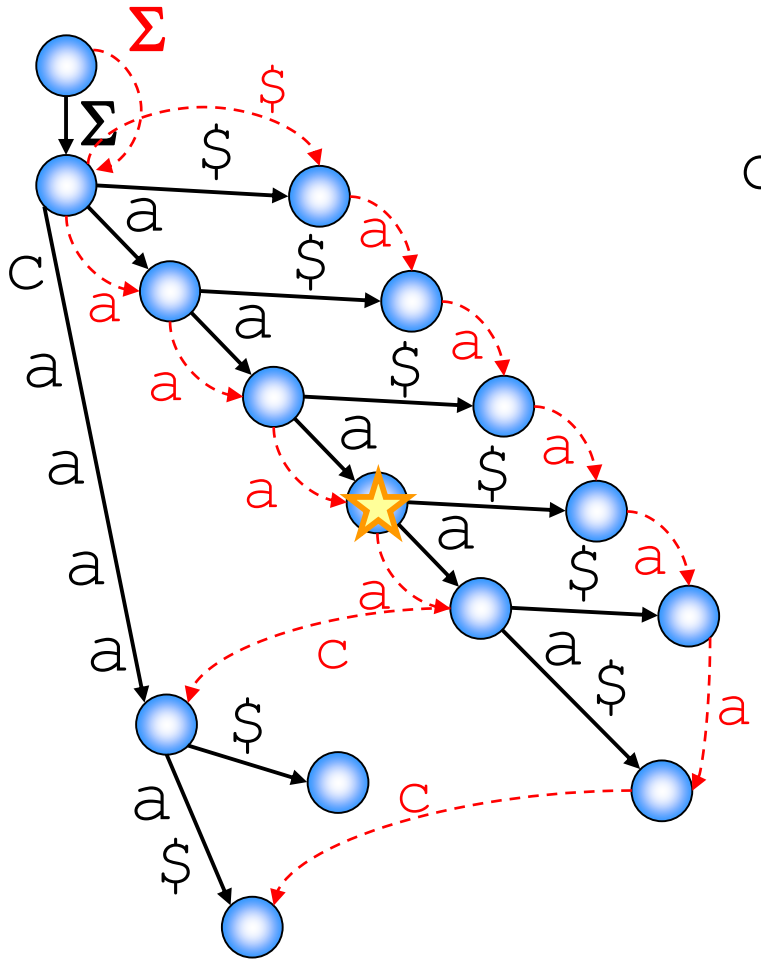
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

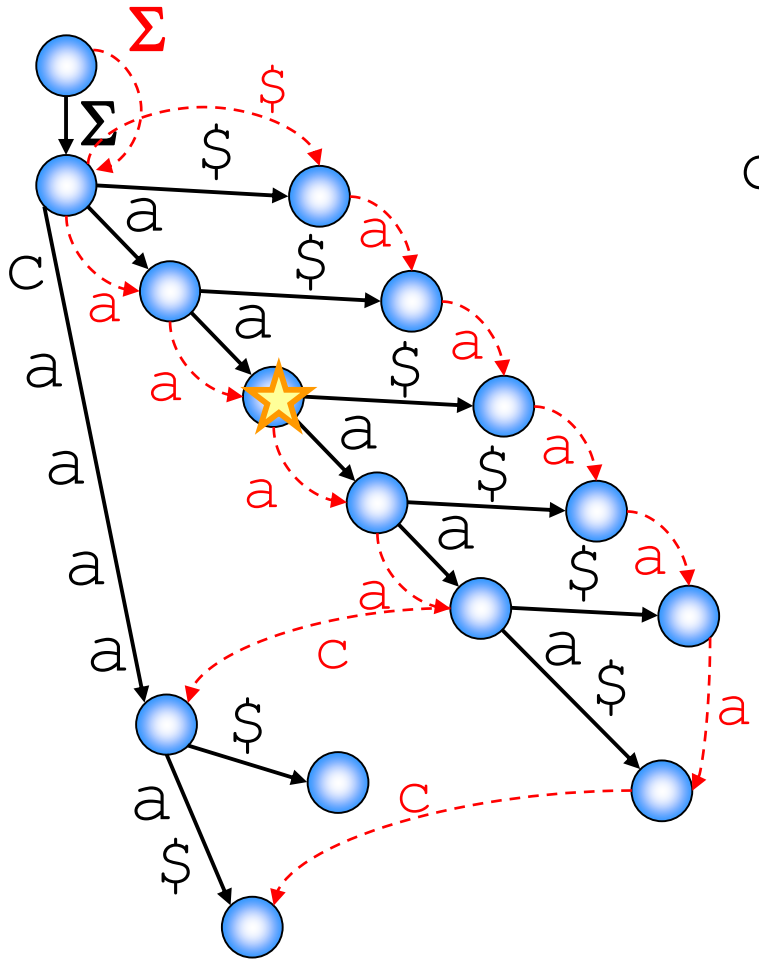
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

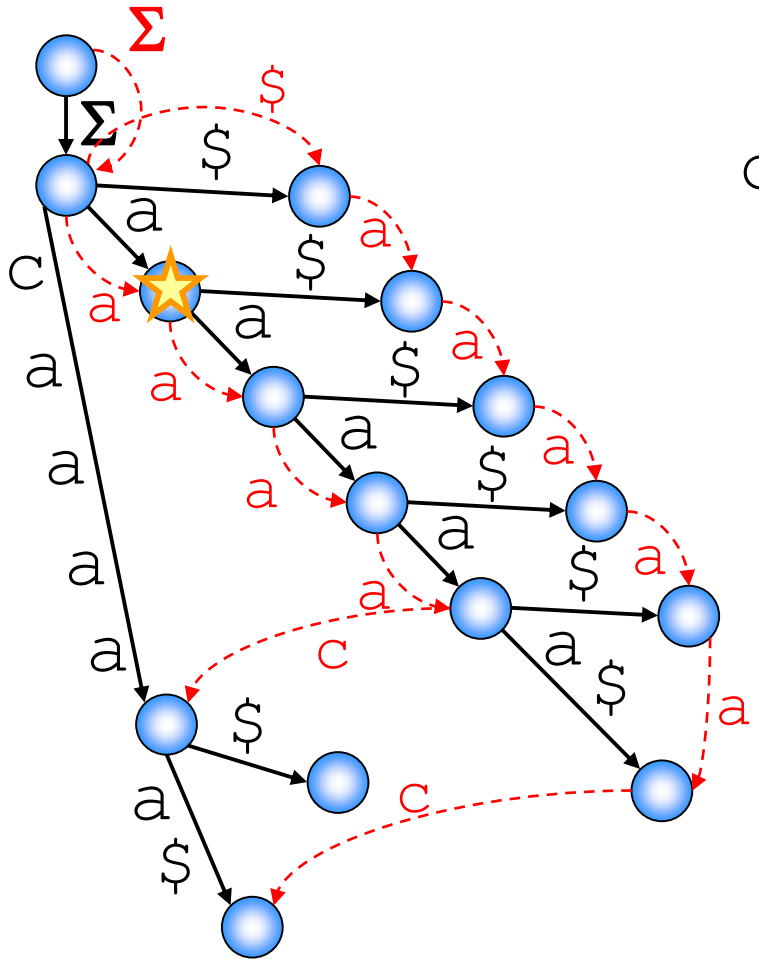
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

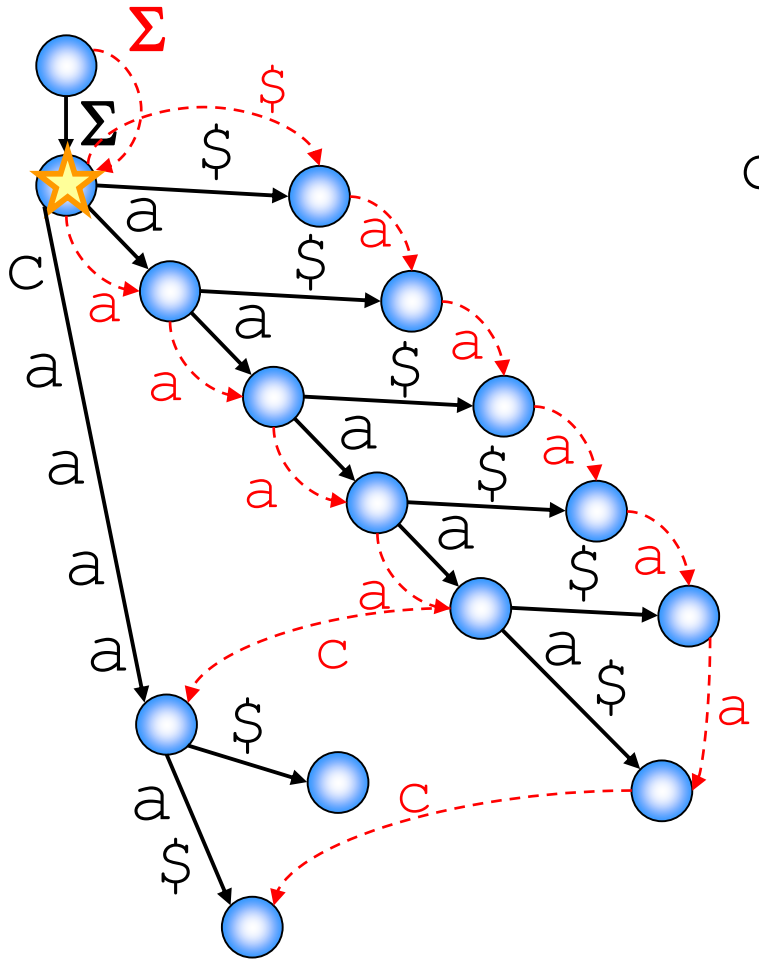
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

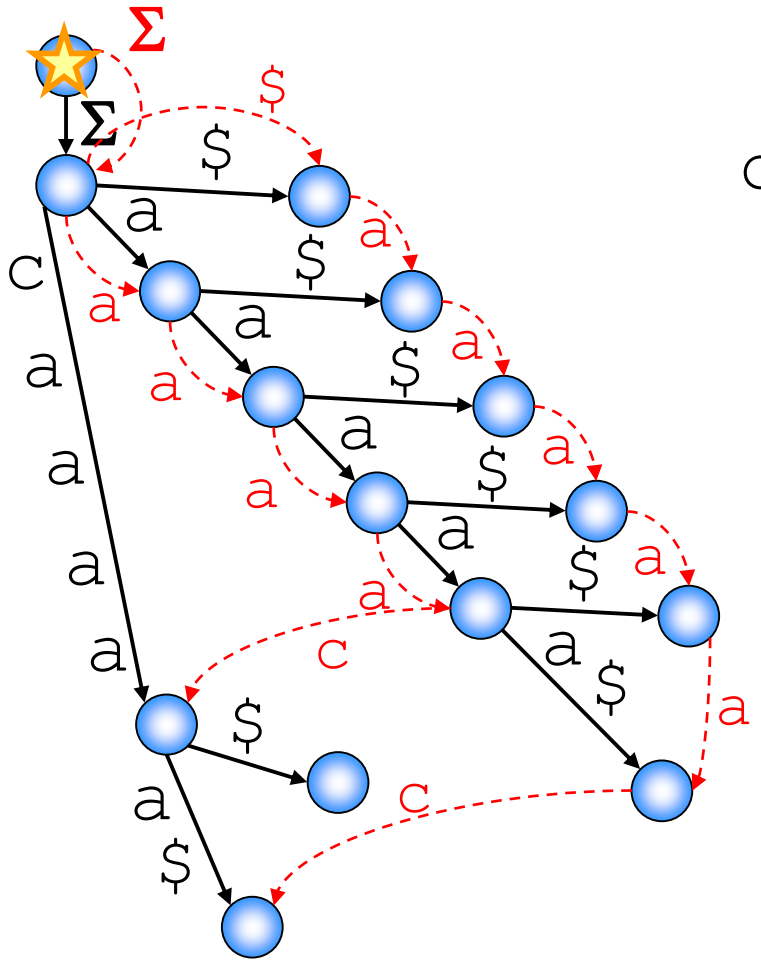
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

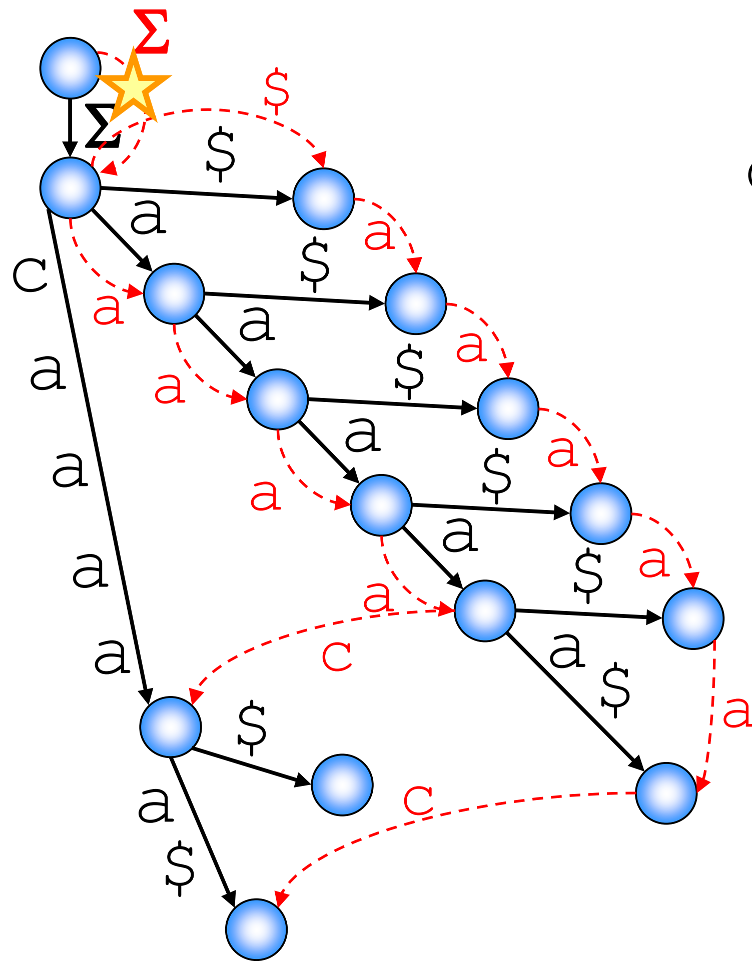
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

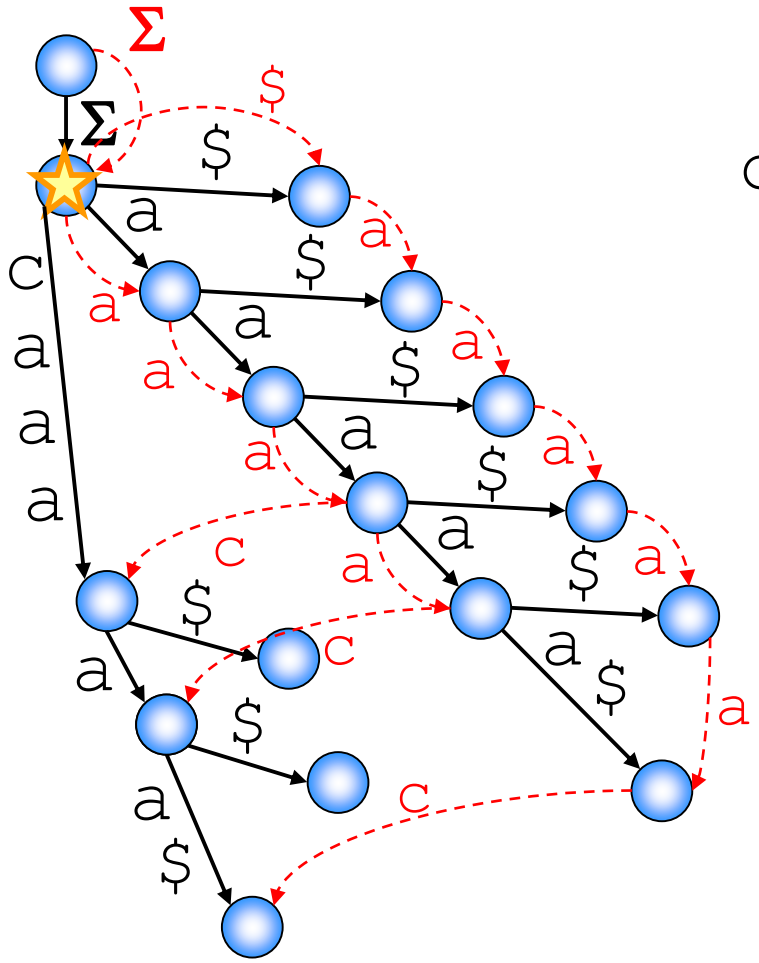
Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



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

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

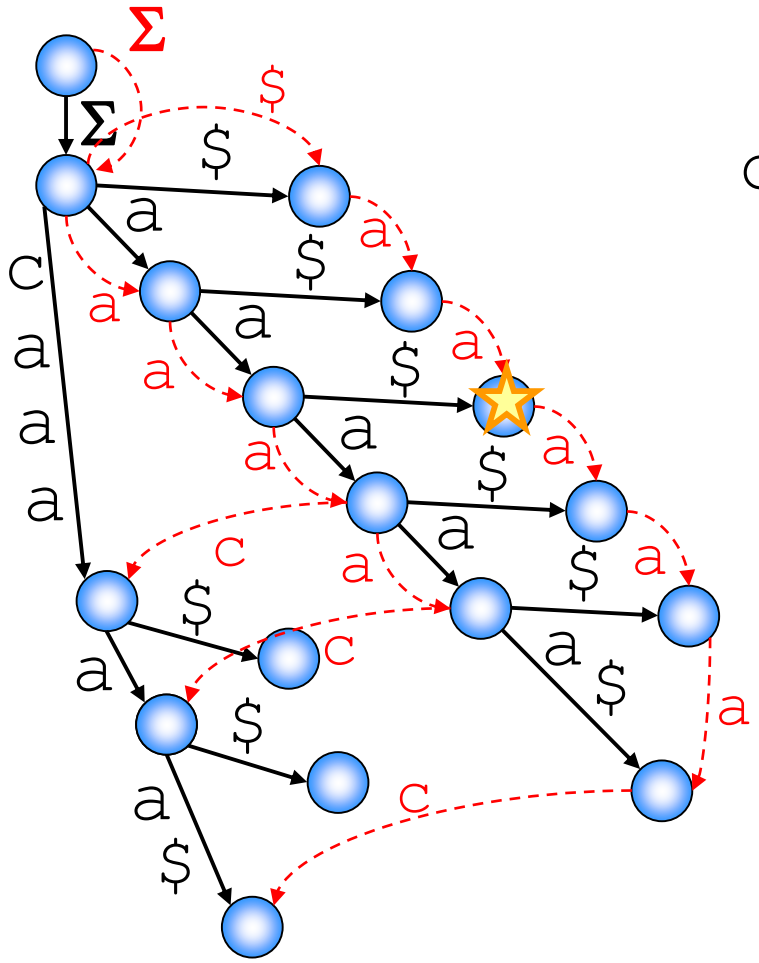
caaa$ T_3

aa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

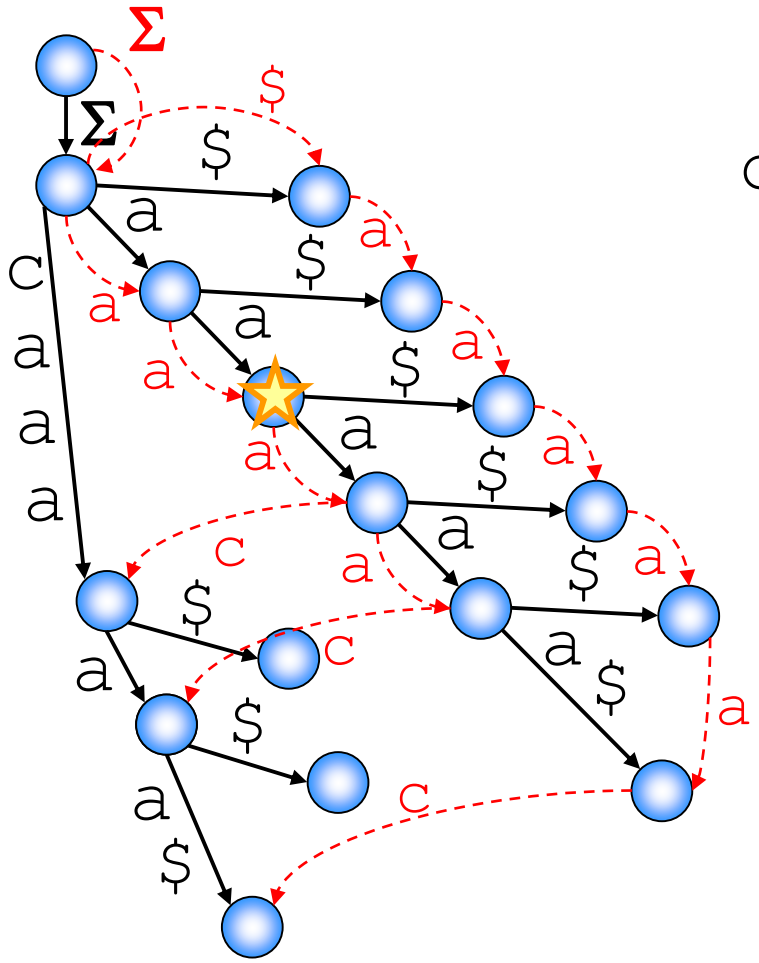
caaa$ T_3

caa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

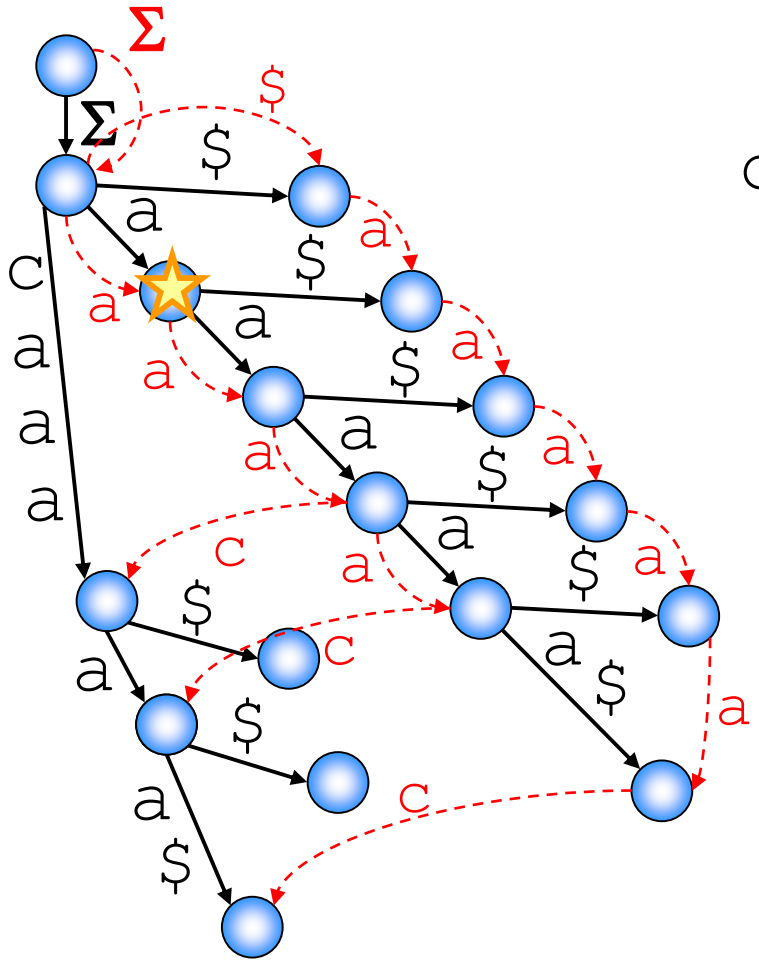
caaa$ T_3

caa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

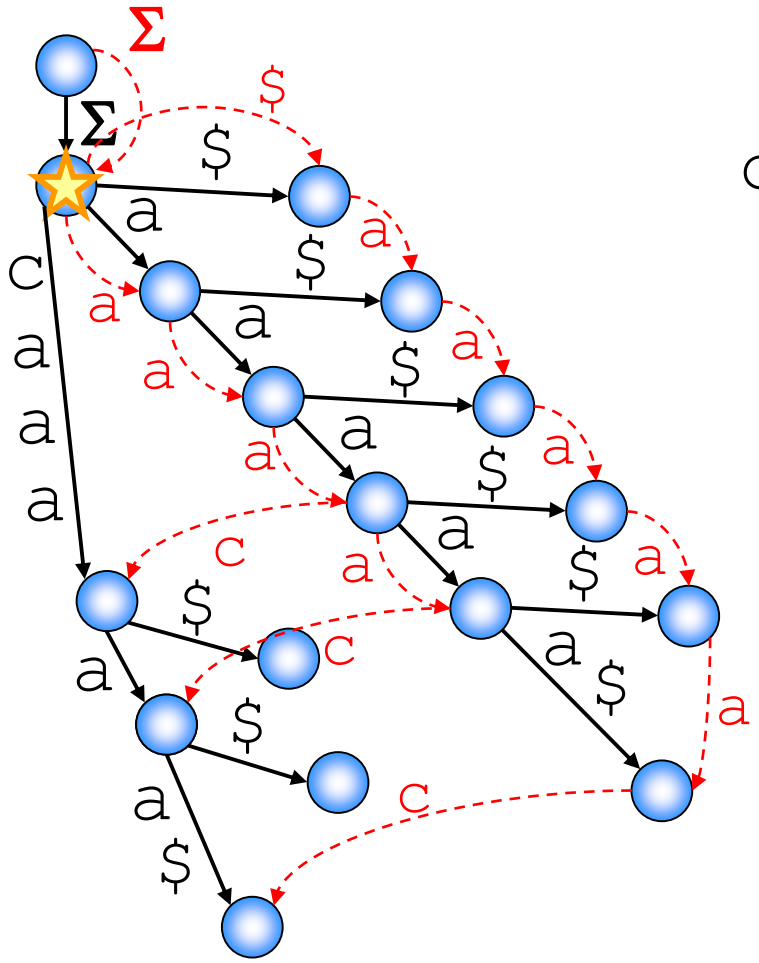
caaa$ T_3

caa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

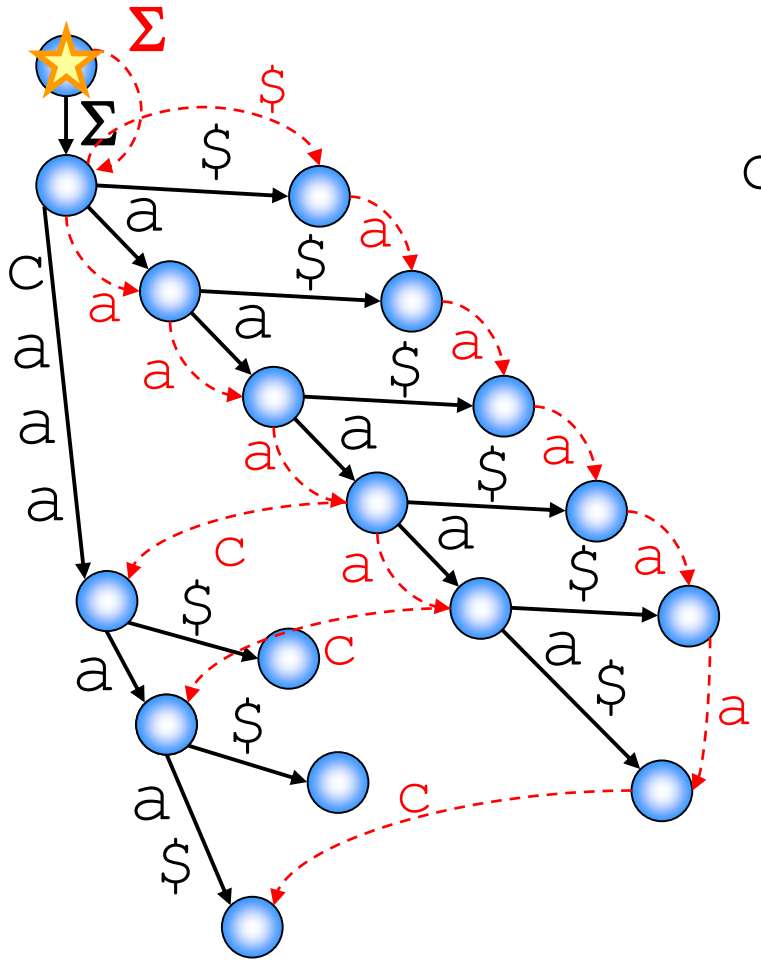
caaa$ T_3

caa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

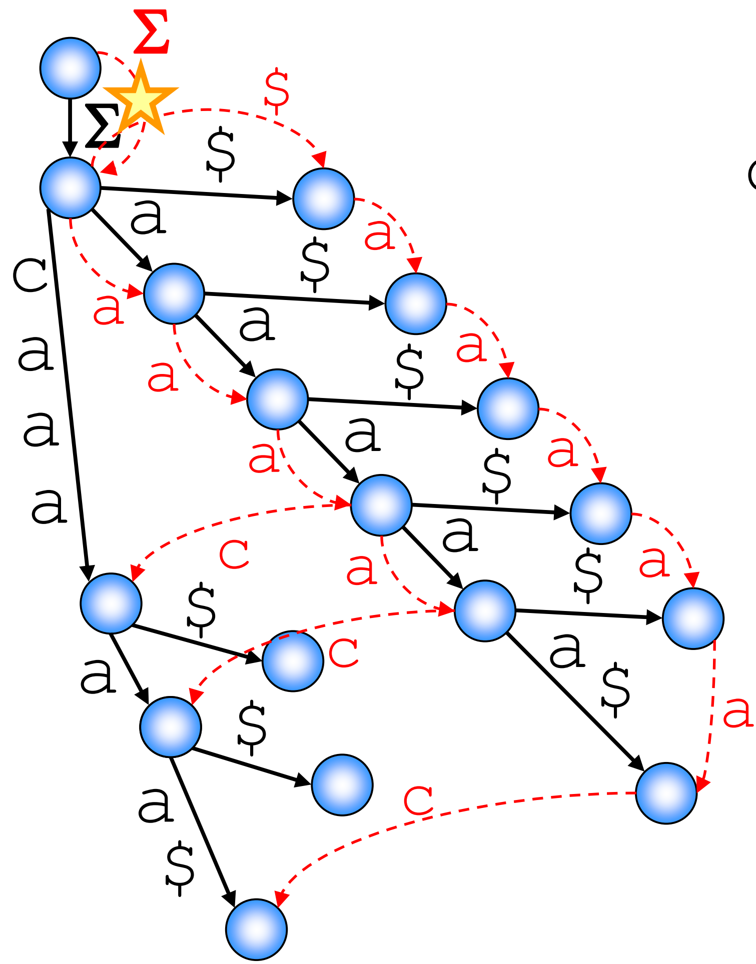
caaa$ T_3

caa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa\$ T_1

caaaa\$ T_2

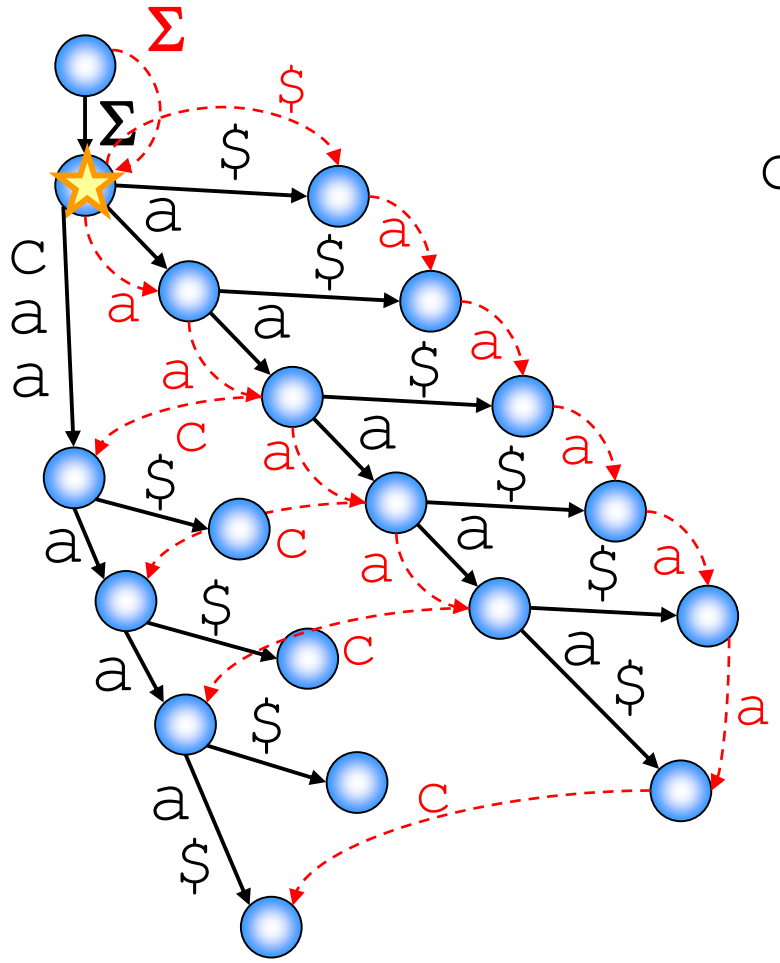
caaa\$ T_3

caa\$ T_4

a\$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

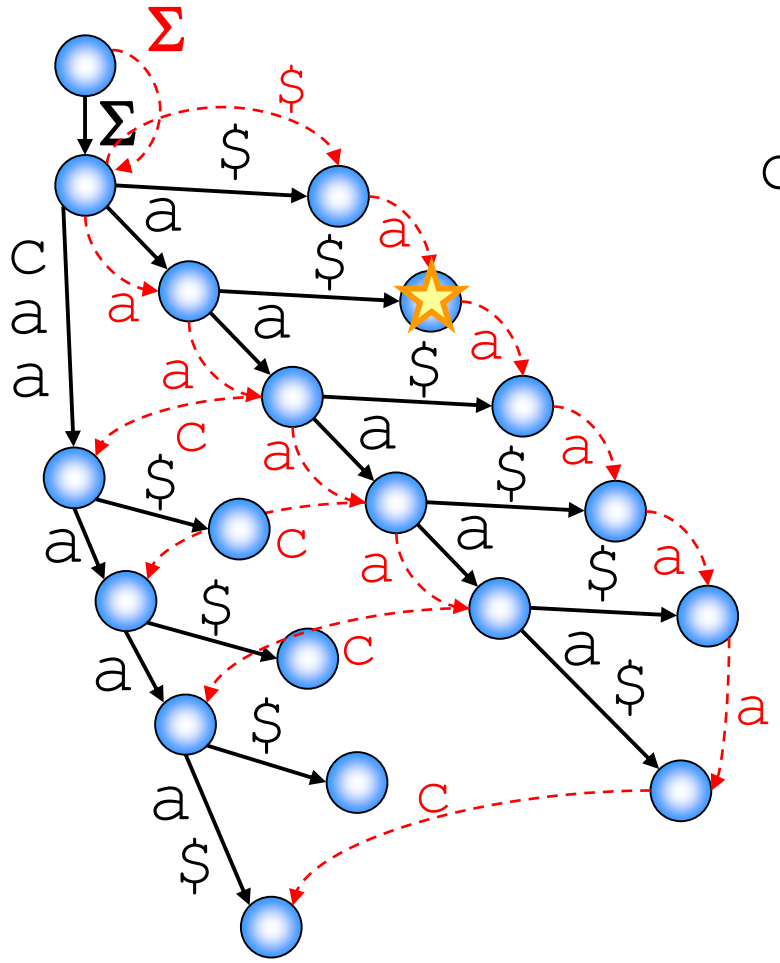
caaa$ T_3

caa$ T_4

a$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

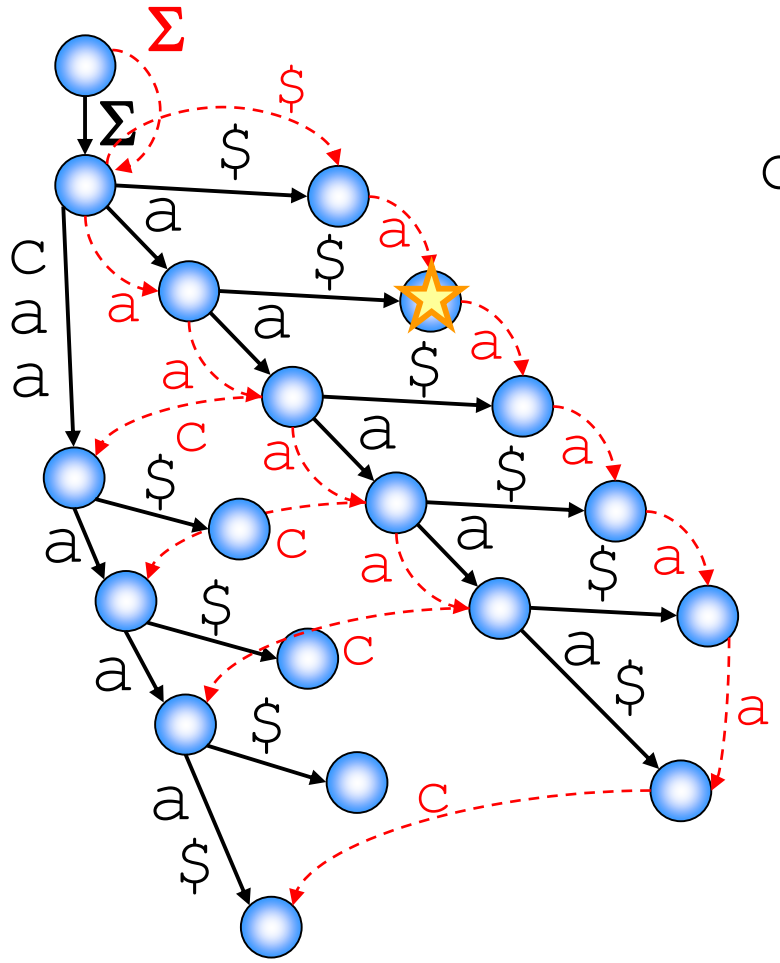
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

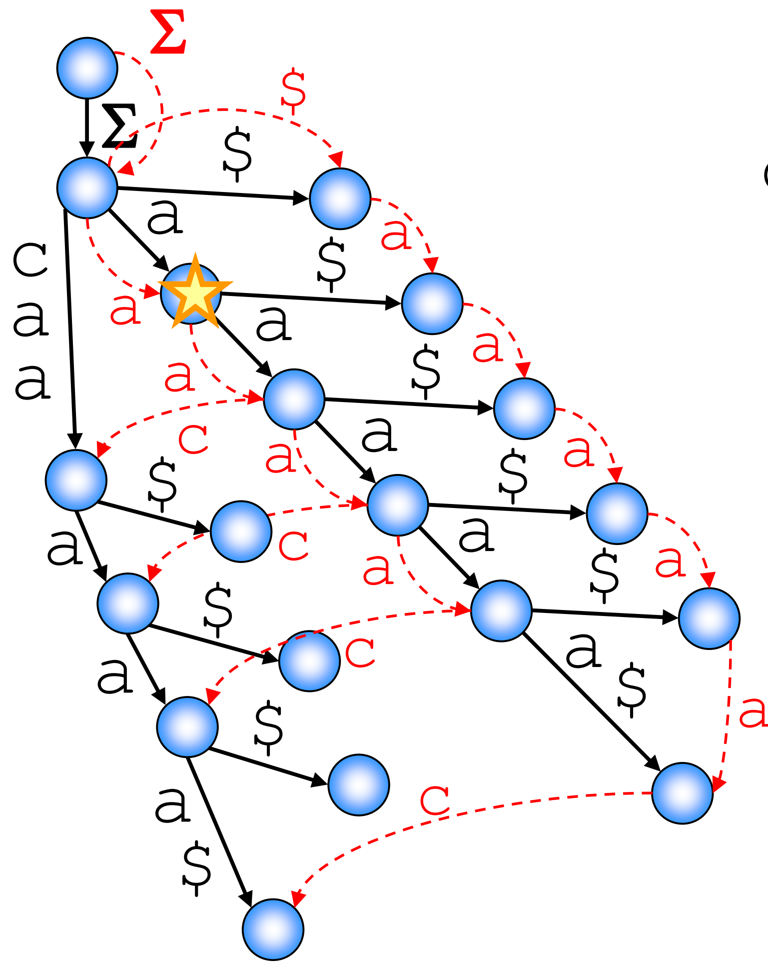
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

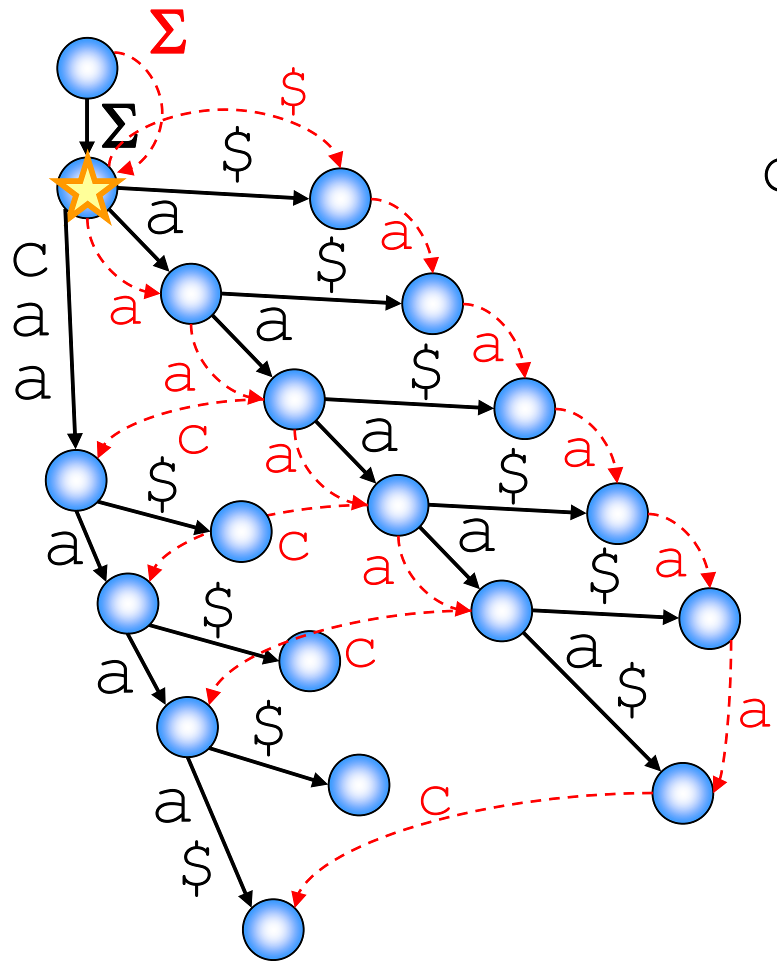
Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



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

Note: some Weiner links are omitted for simplicity

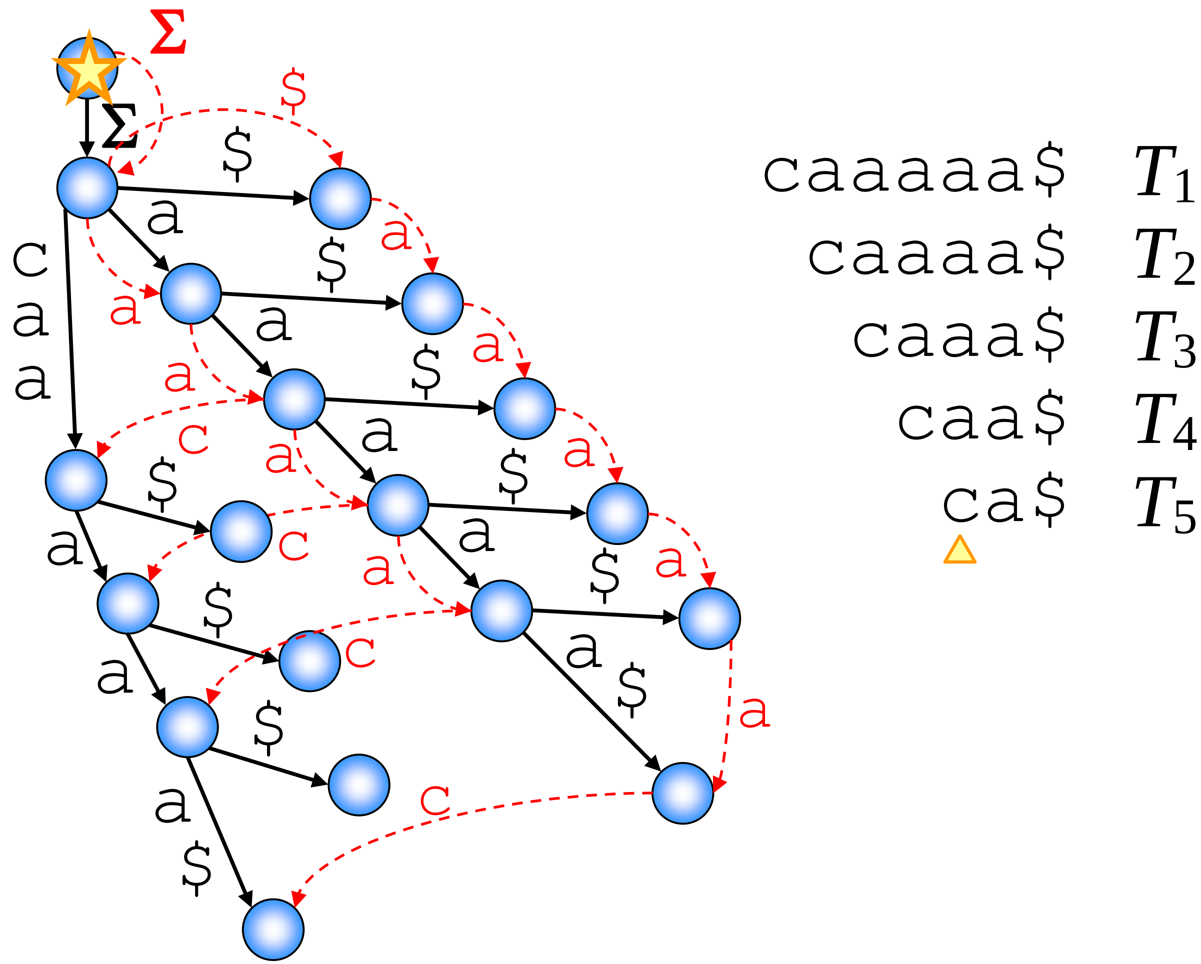
Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



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

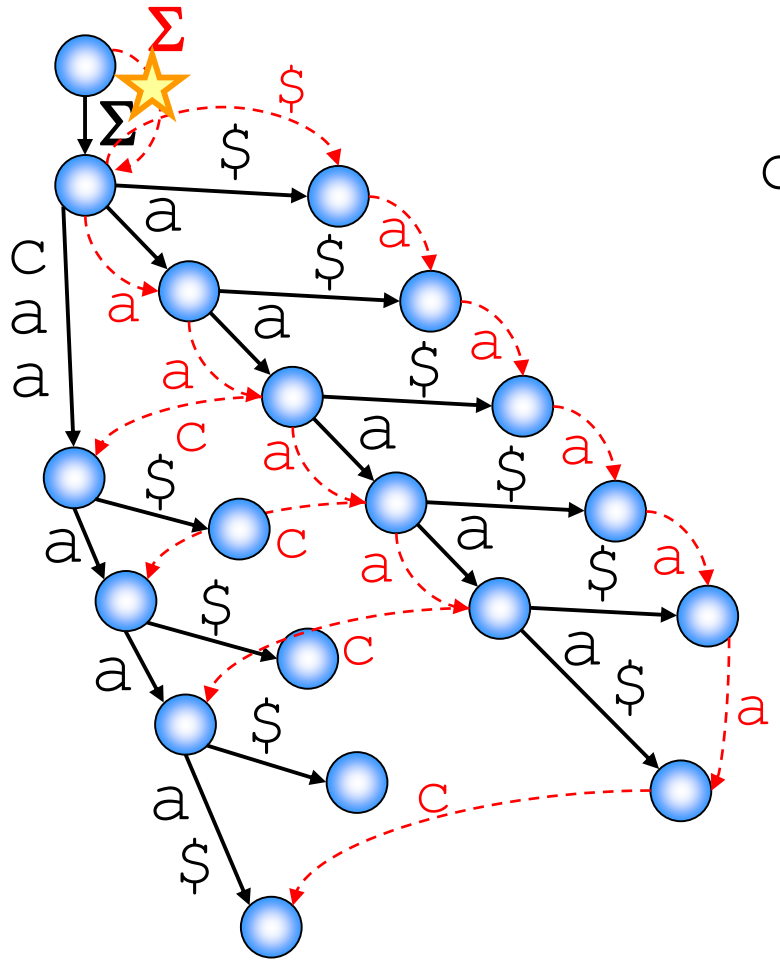
Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

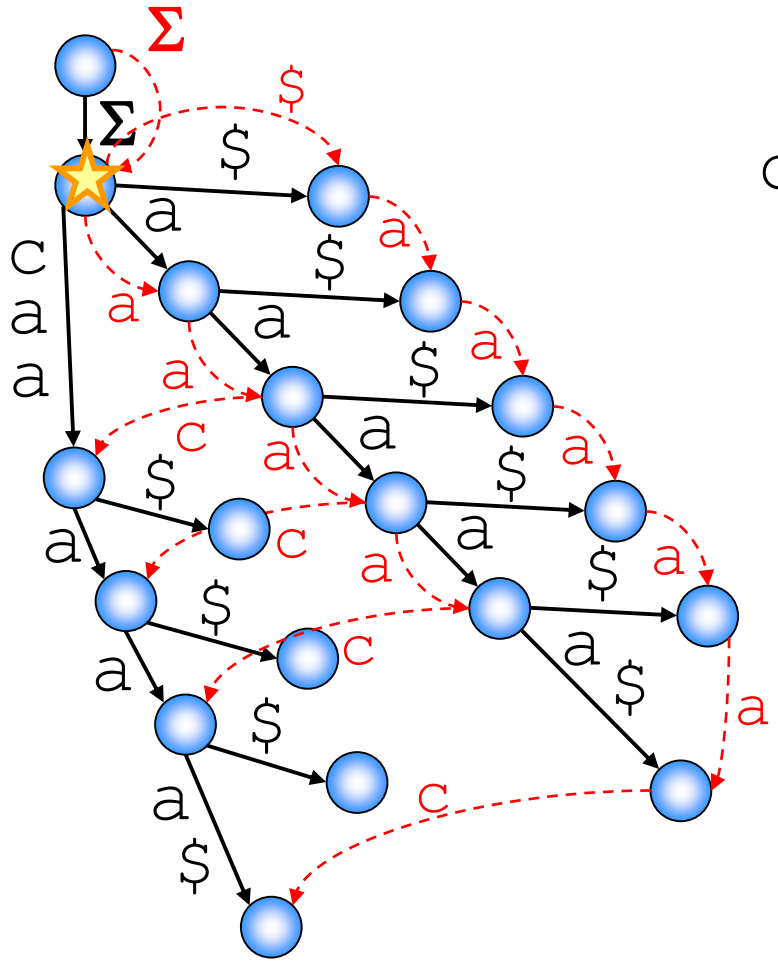
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

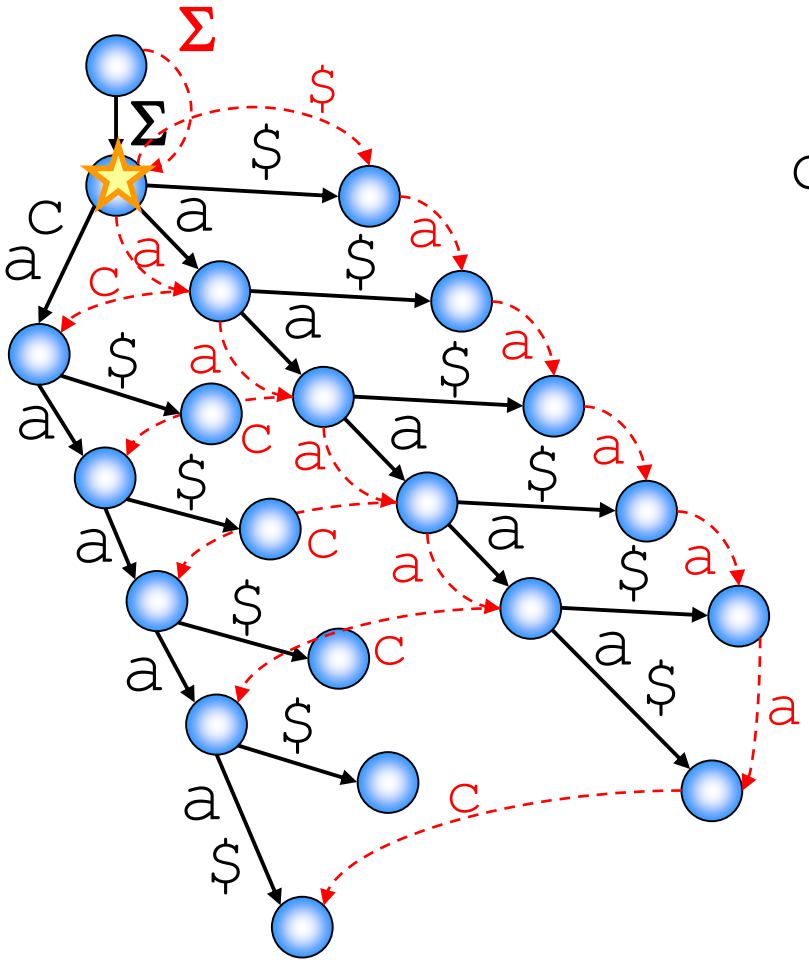
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



caaaaa$ T_1

caaaa$ T_2

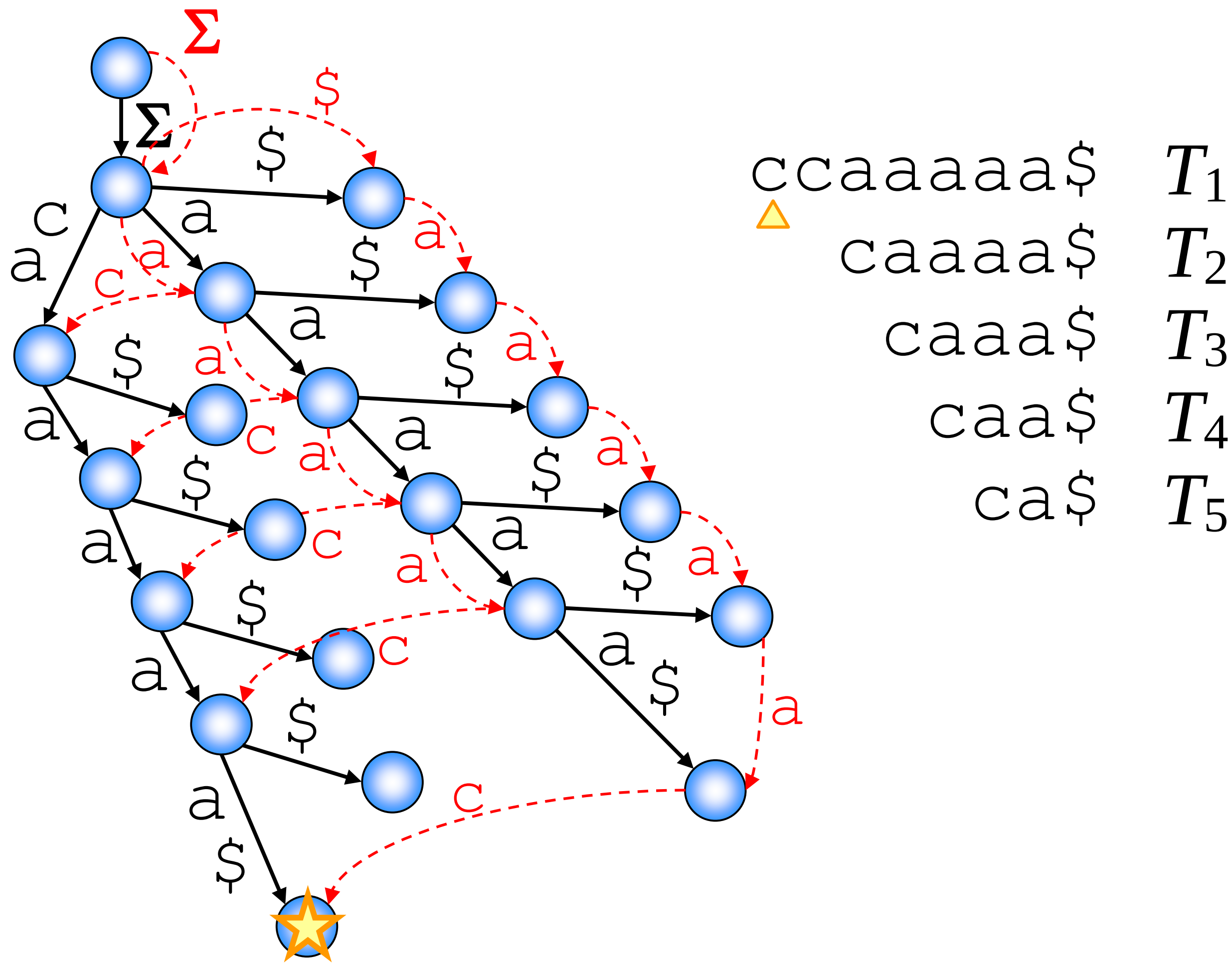
caaa$ T_3

caa$ T_4

ca$ T_5

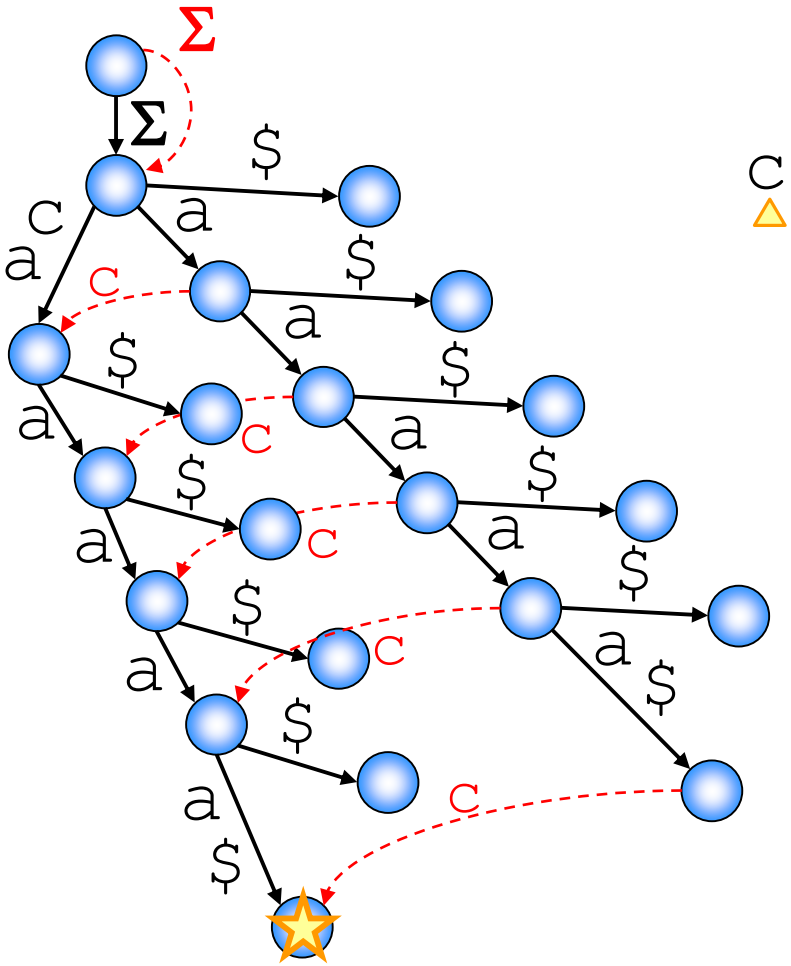
Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

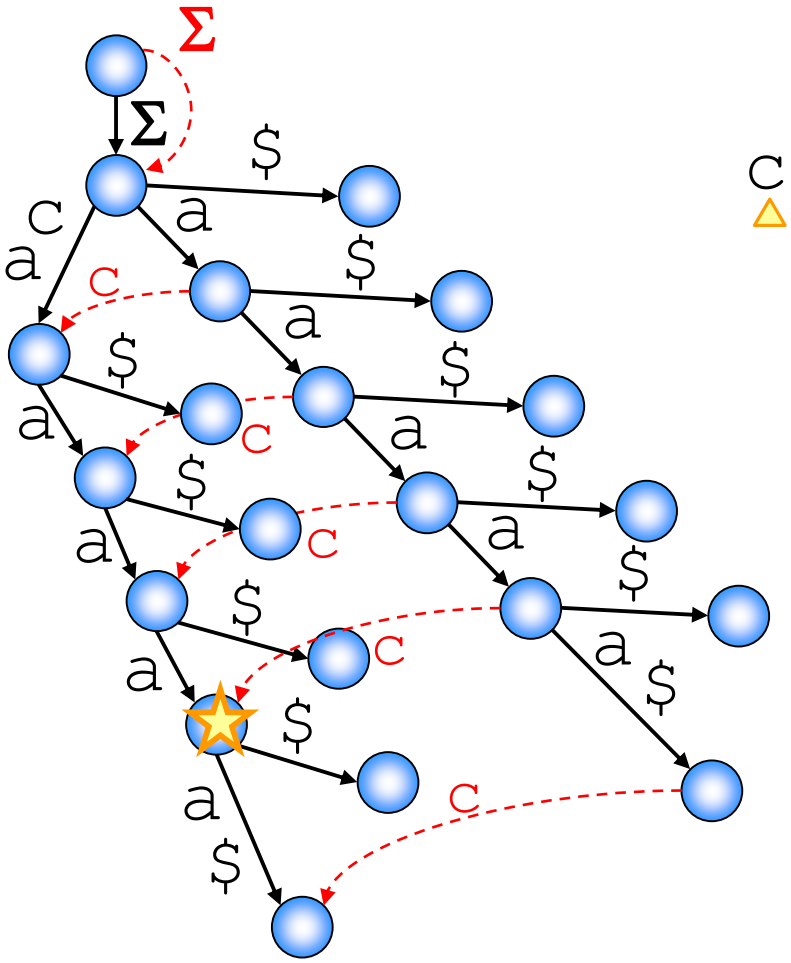
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

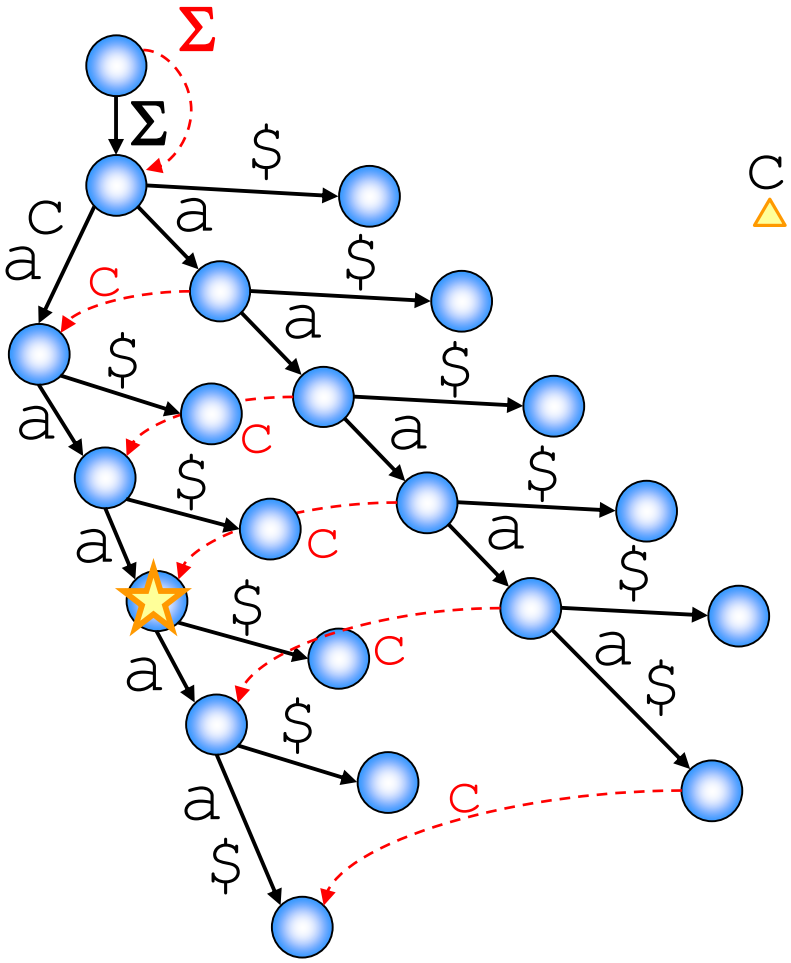
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

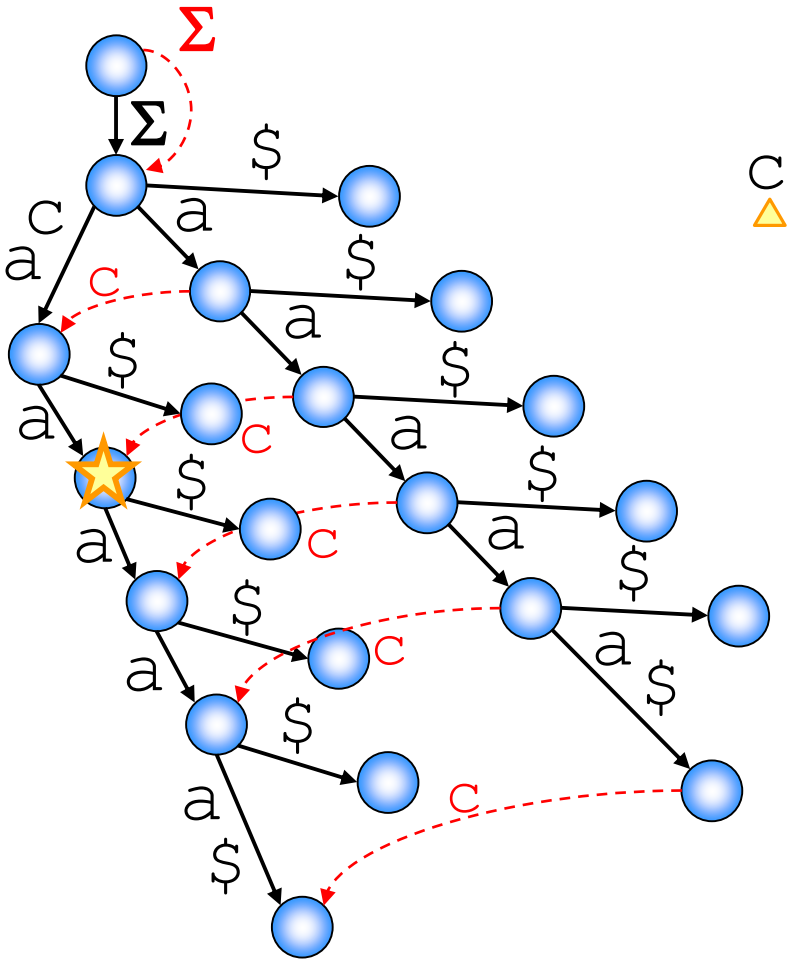
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

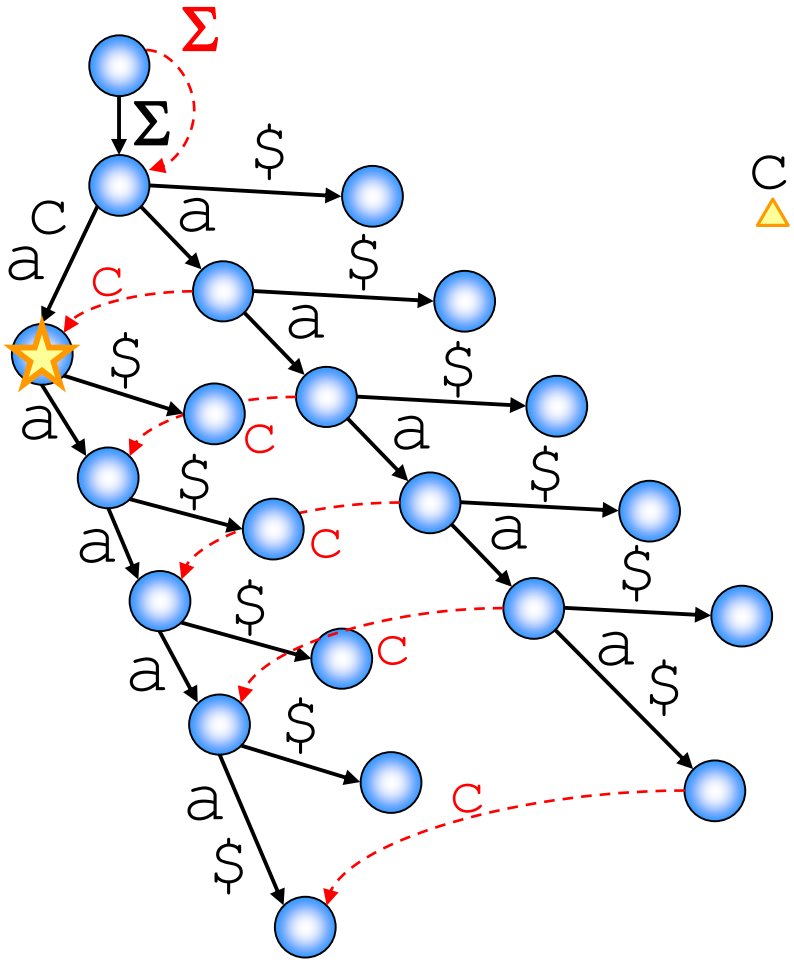
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

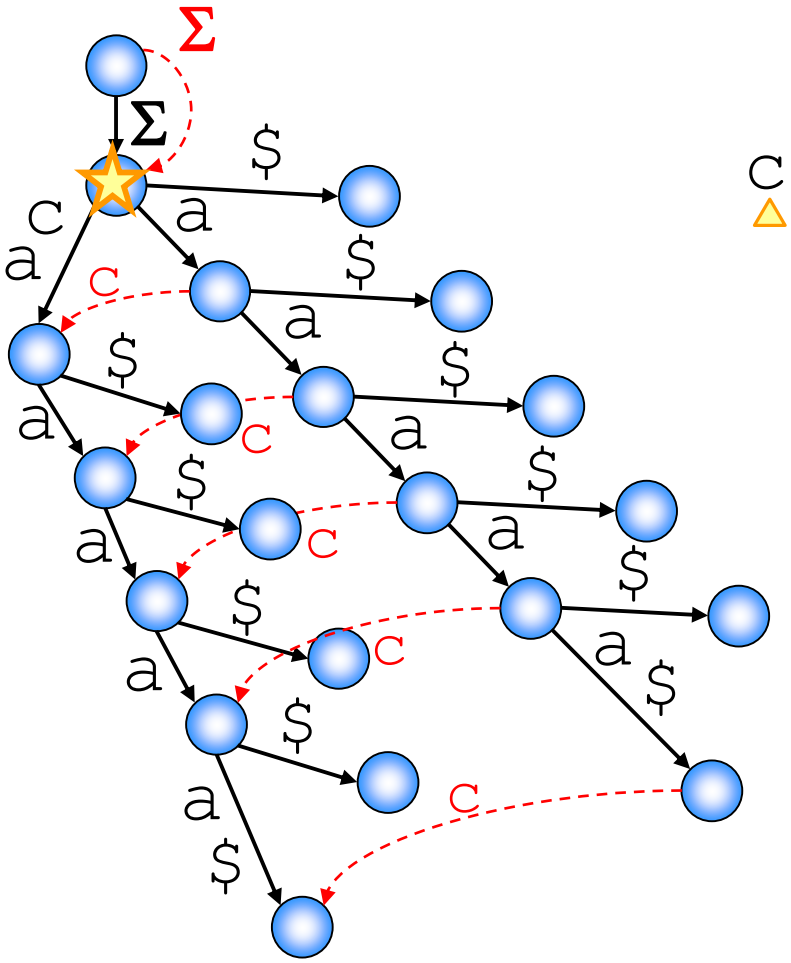
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

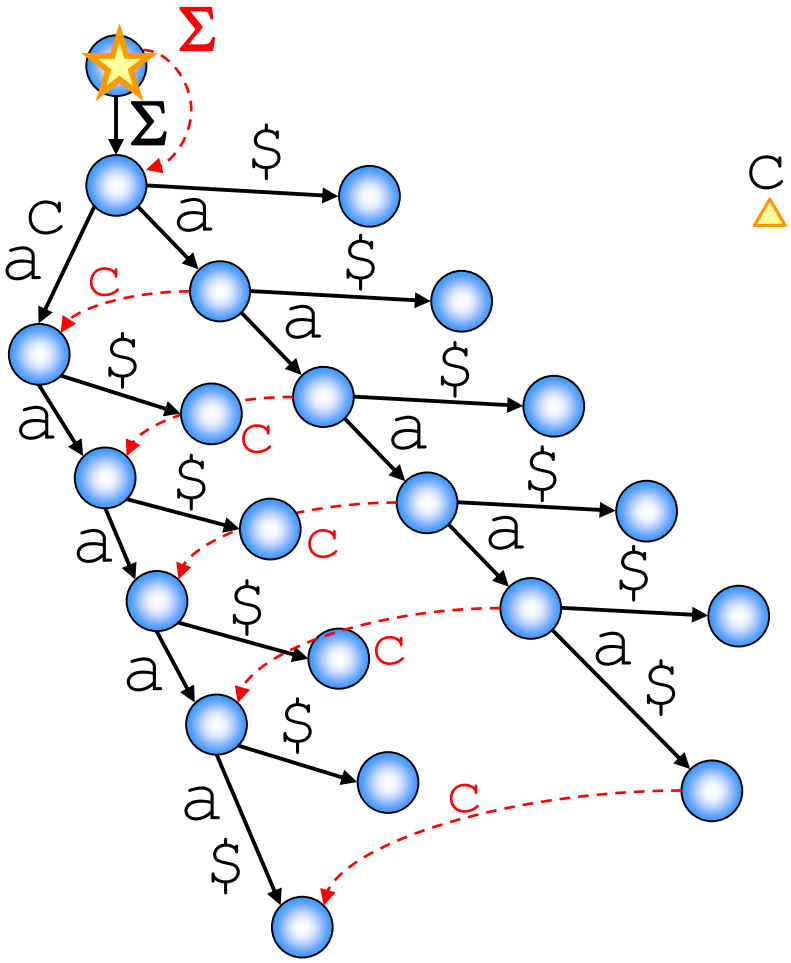
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

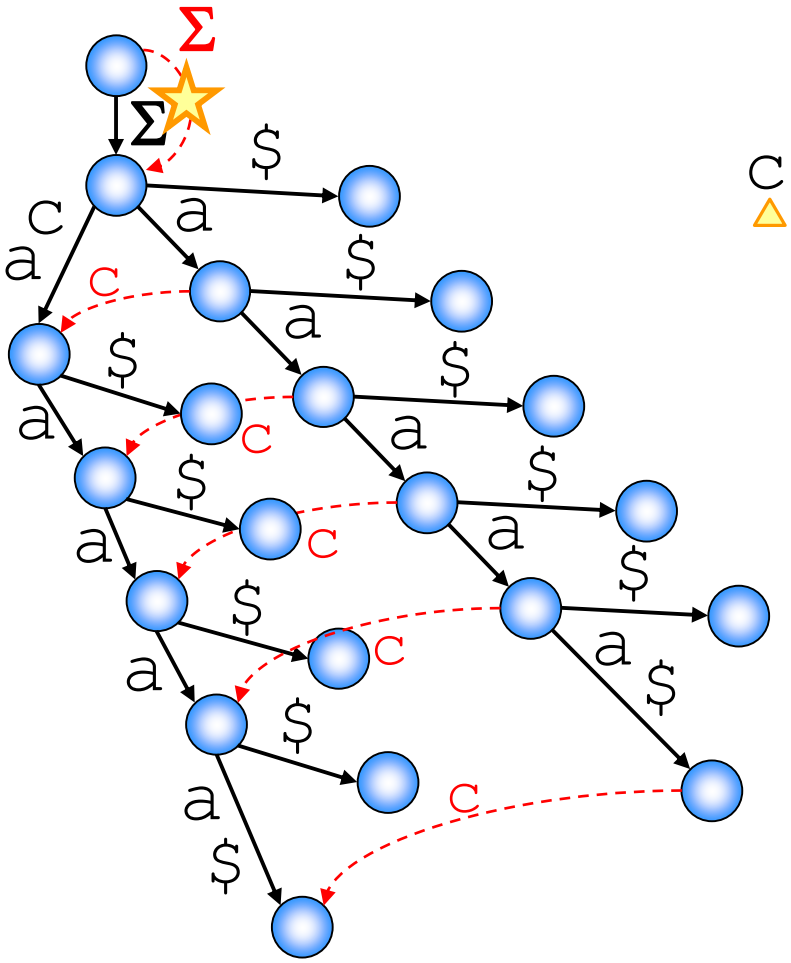
caaaaa$ T_2

caaa$ T_3

caa$ T_4

ca$ T_5

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

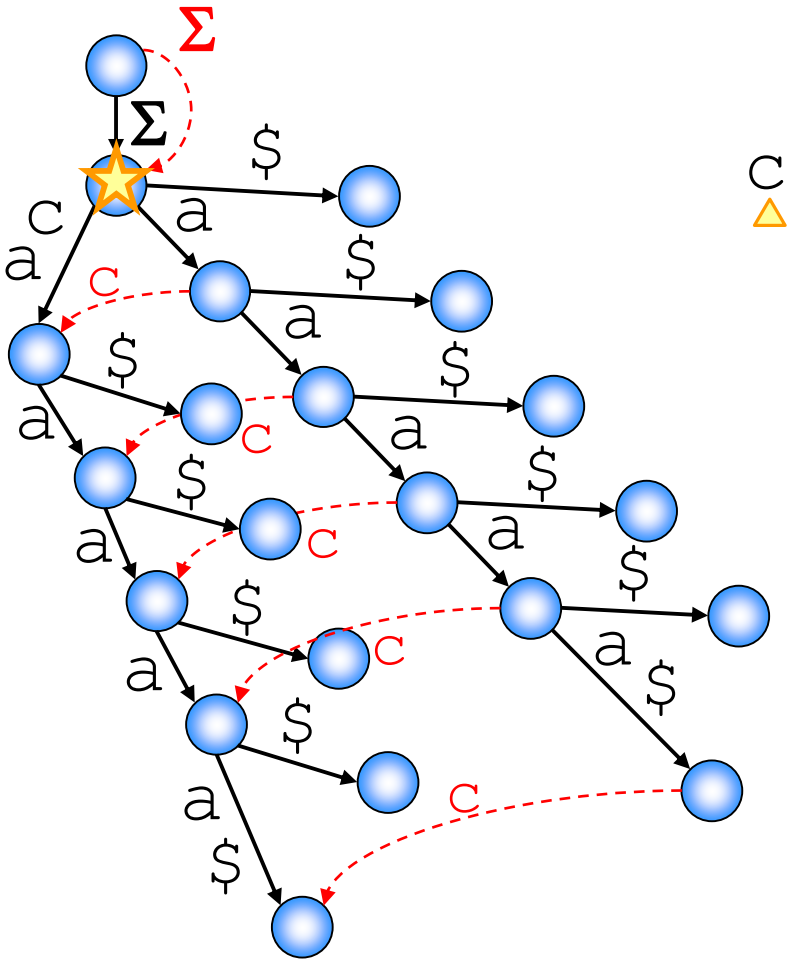
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

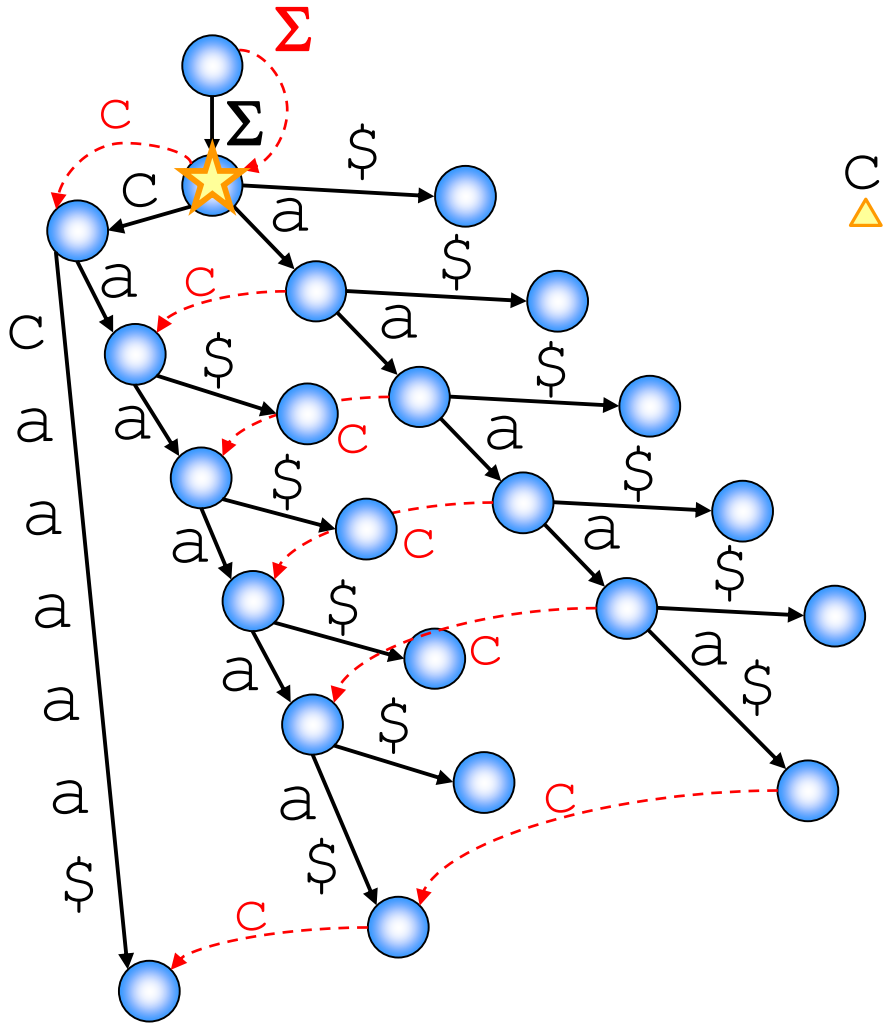
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$ T_1

caaaaa$ T_2

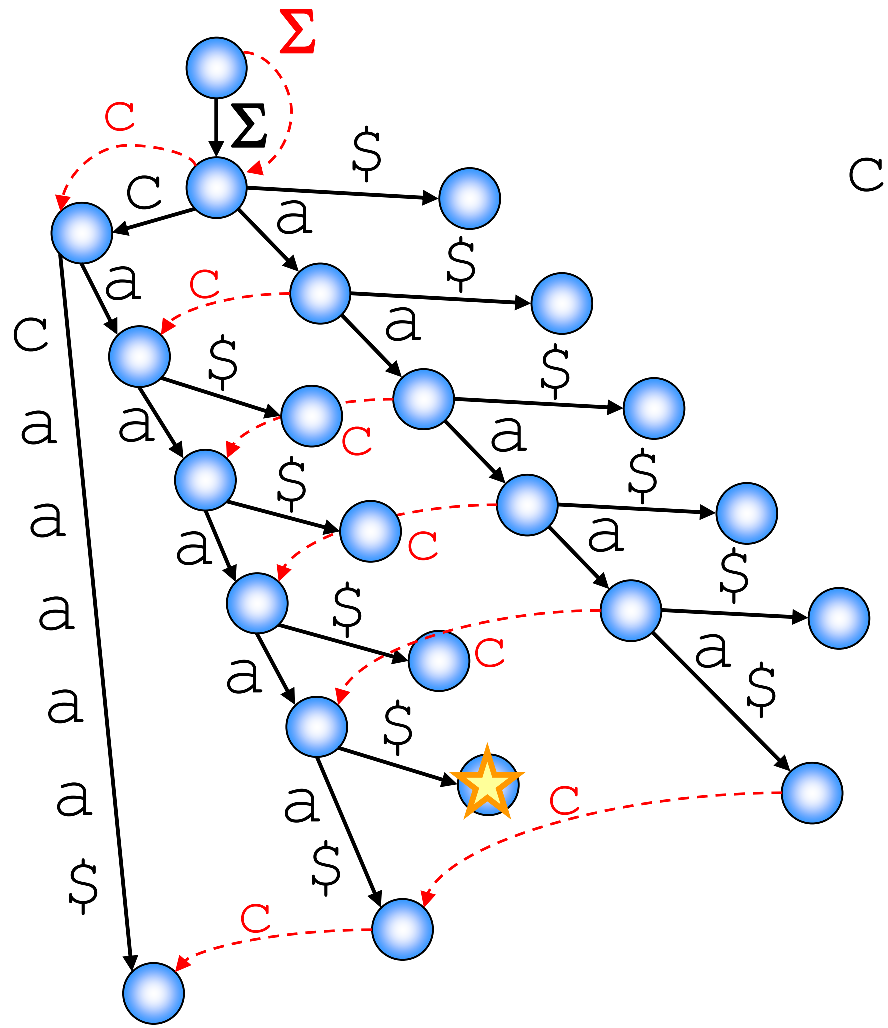
caaa$ T_3

caa$ T_4

ca$ T_5

Note: some Weiner links are omitted for simplicity

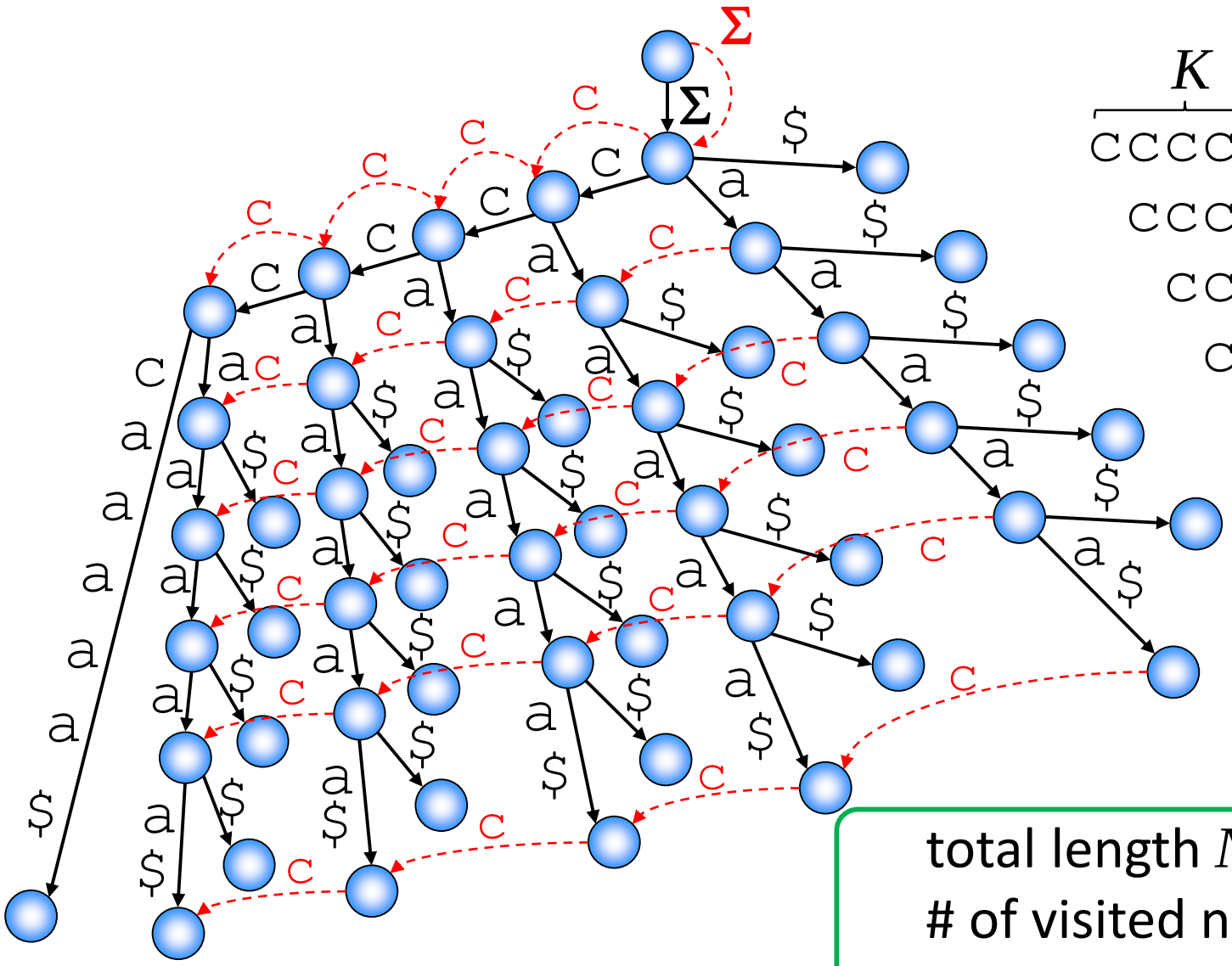
Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



ccaaaaaa$	T_1
ccaaaaa$	T_2
caaa$	T_3
caa$	T_4
ca$	T_5

Note: some Weiner links are omitted for simplicity

Fully-online Weiner visits $\Omega(N^{1.5})$ nodes



K K

ccccccaaaaa$

cccccaaaa$

ccccccaaa$ K

ccccccaa$

cccccca$

total length $N = \Theta(K^2)$ &

# of visited nodes = $\Theta(K^3)$

→ # of visited nodes = $\Omega(N^{1.5})$

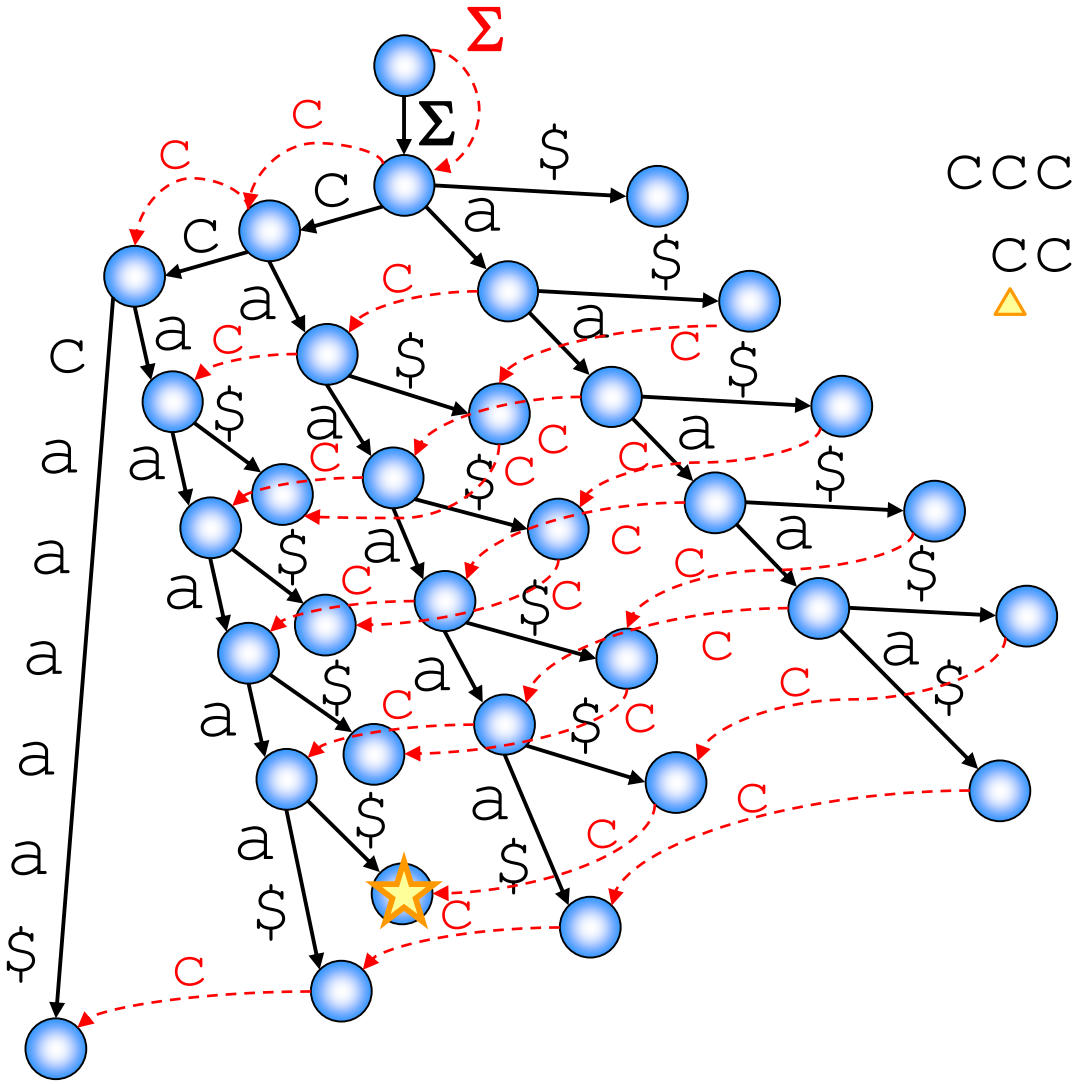
Fully-online Weiner

Theorem 1

Weiner's right-to-left suffix tree algorithm for fully-online multiple texts needs to visit $\Omega(N^{1.5})$ nodes, and this bound is tight ($O(N^{1.5})$ in the worst case).

We can do better with non-trivial use of nearest marked ancestor (NMA) data structure!

Speed-up with NMA



ccccaaaaa$ T_1

ccccaaaa$ T_2

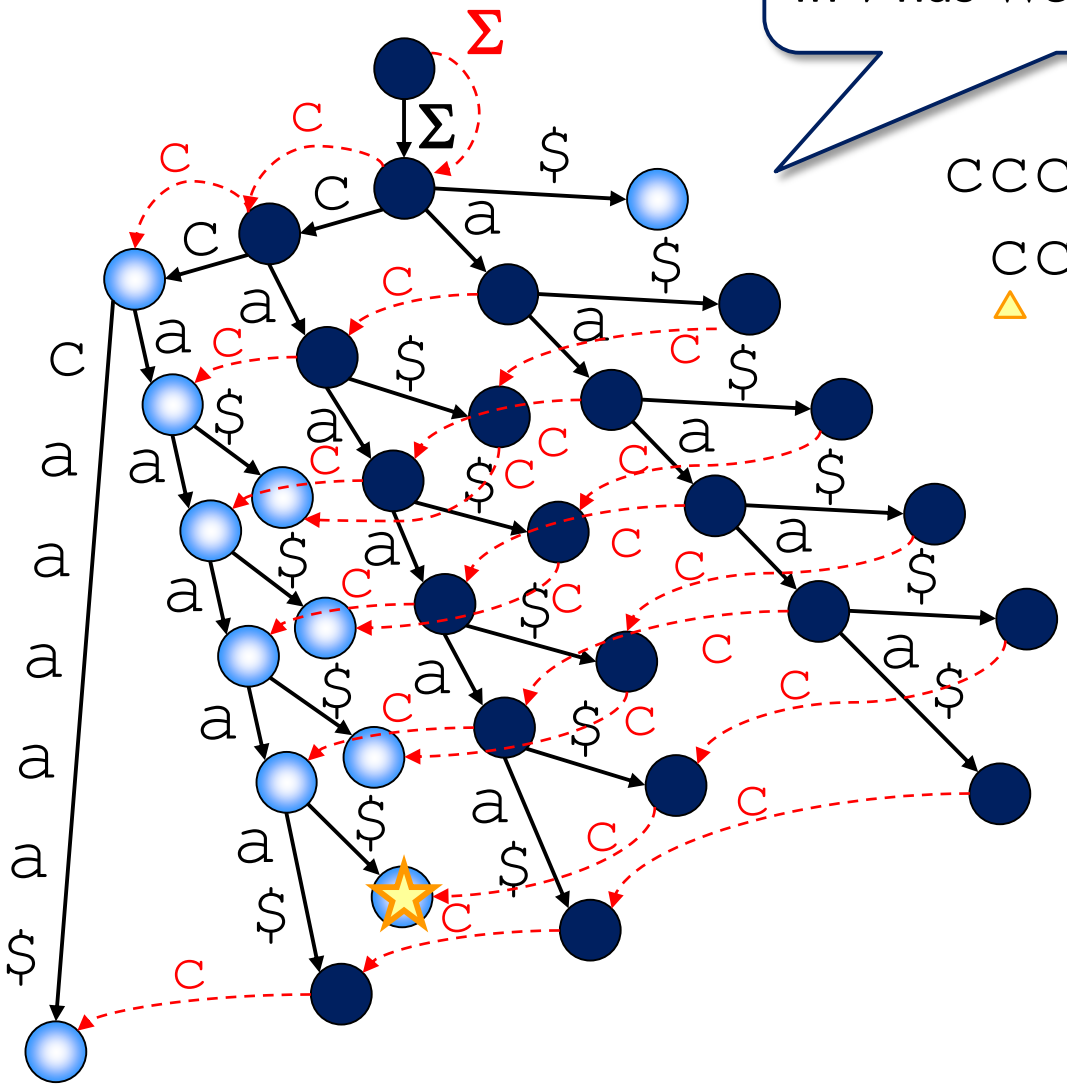
cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Speed-up with NMA

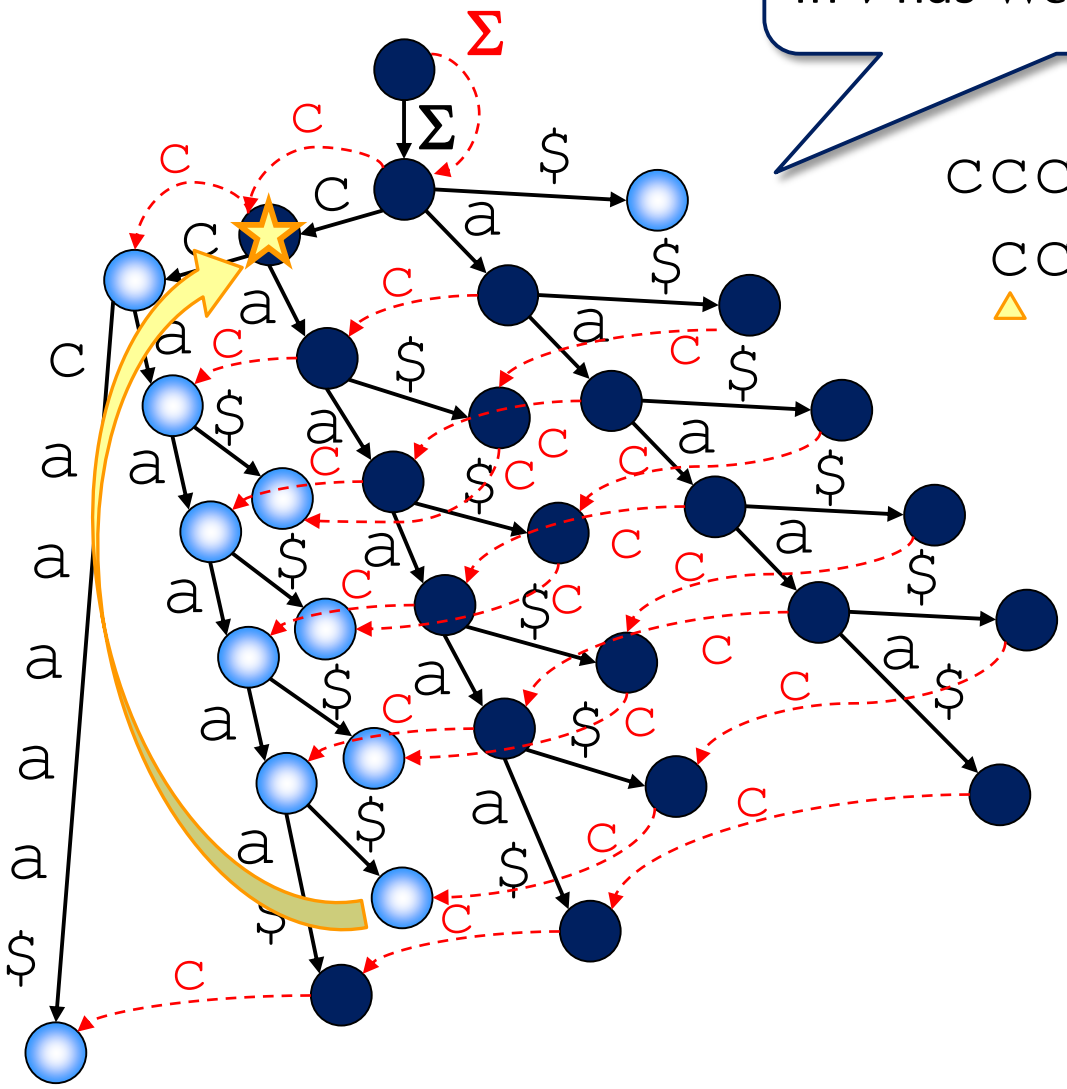
Node v is marked ●
iff v has Weiner link with c



ccccaaaaa$ T_1
ccccaaaa$ T_2
△
cccaaa$ T_3
cccaa$ T_4
ccca$ T_5

Speed-up with NMA

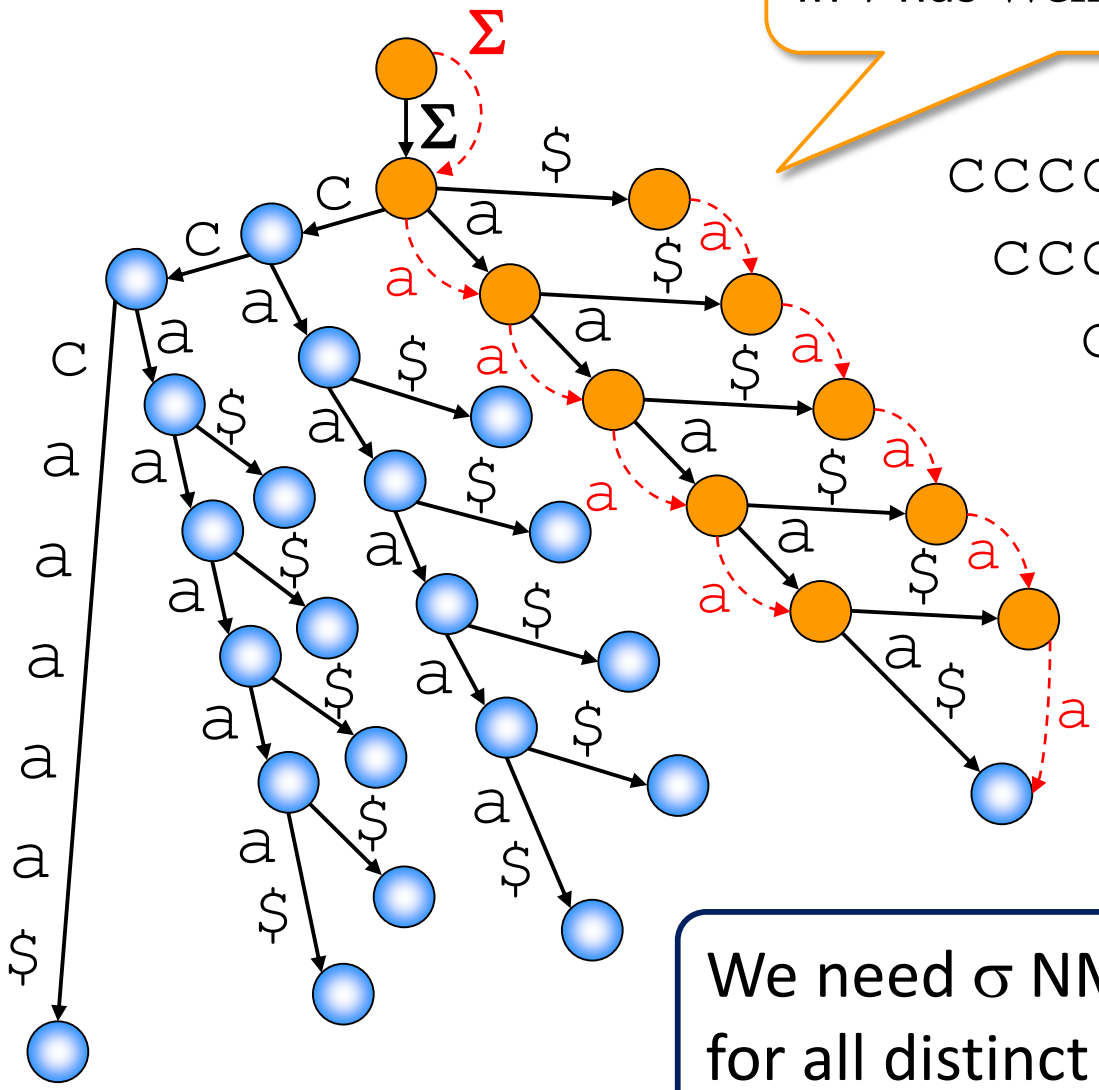
Node v is marked ●
iff v has Weiner link with c



ccccaaaaa$ T_1
ccccaaaa$ T_2
△
cccaaa$ T_3
cccaa$ T_4
ccca$ T_5

Speed-up with NMA

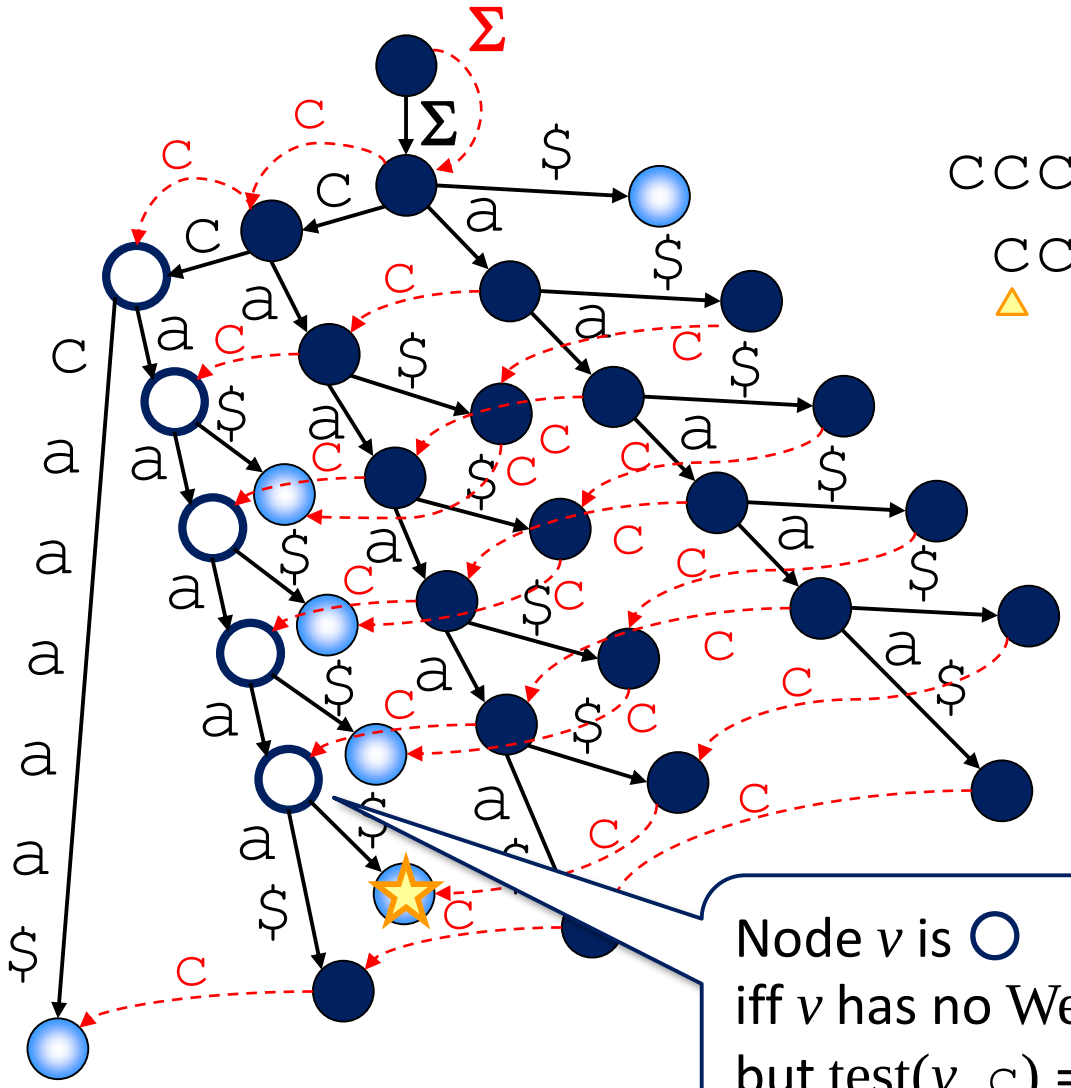
Node v is marked ●
iff v has Weiner link with a



ccccaaaaa$ T_1
ccccaaaa$ T_2
cccaaa$ T_3
cccaa$ T_4
ccca$ T_5

We need σ NMA structures for all distinct σ characters
→ $O(\sigma N)$ space…?

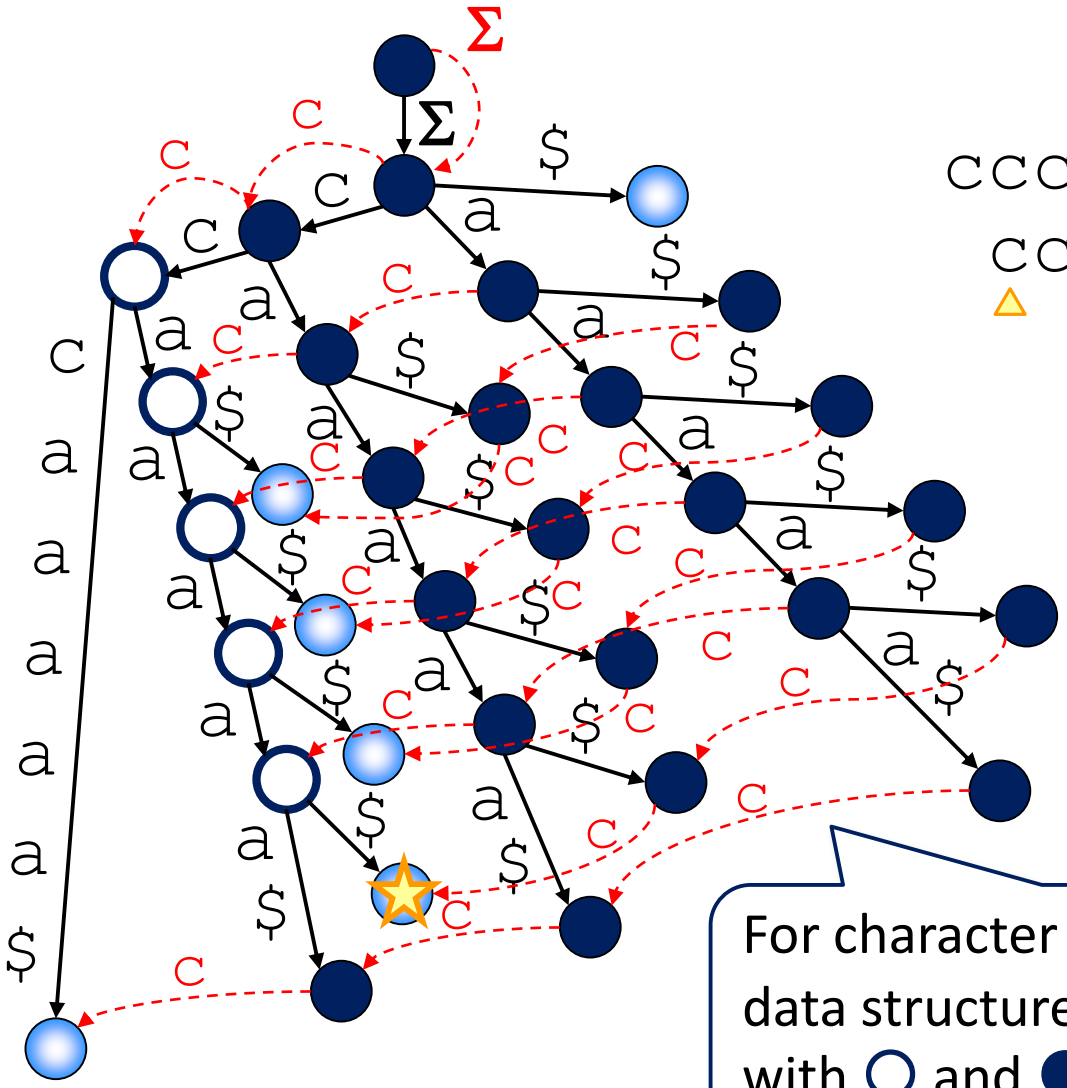
$O(N)$ space is enough for NMA



ccccaaaaa$ T_1
ccccaaaa$ T_2
cccaaa$ T_3
cccaa$ T_4
ccca$ T_5

Node v is ○
iff v has no Weiner link with c
but test(v, c) = true

$O(N)$ space is enough for NMA



ccccaaaaa$ T_1

ccccaaaa$ T_2

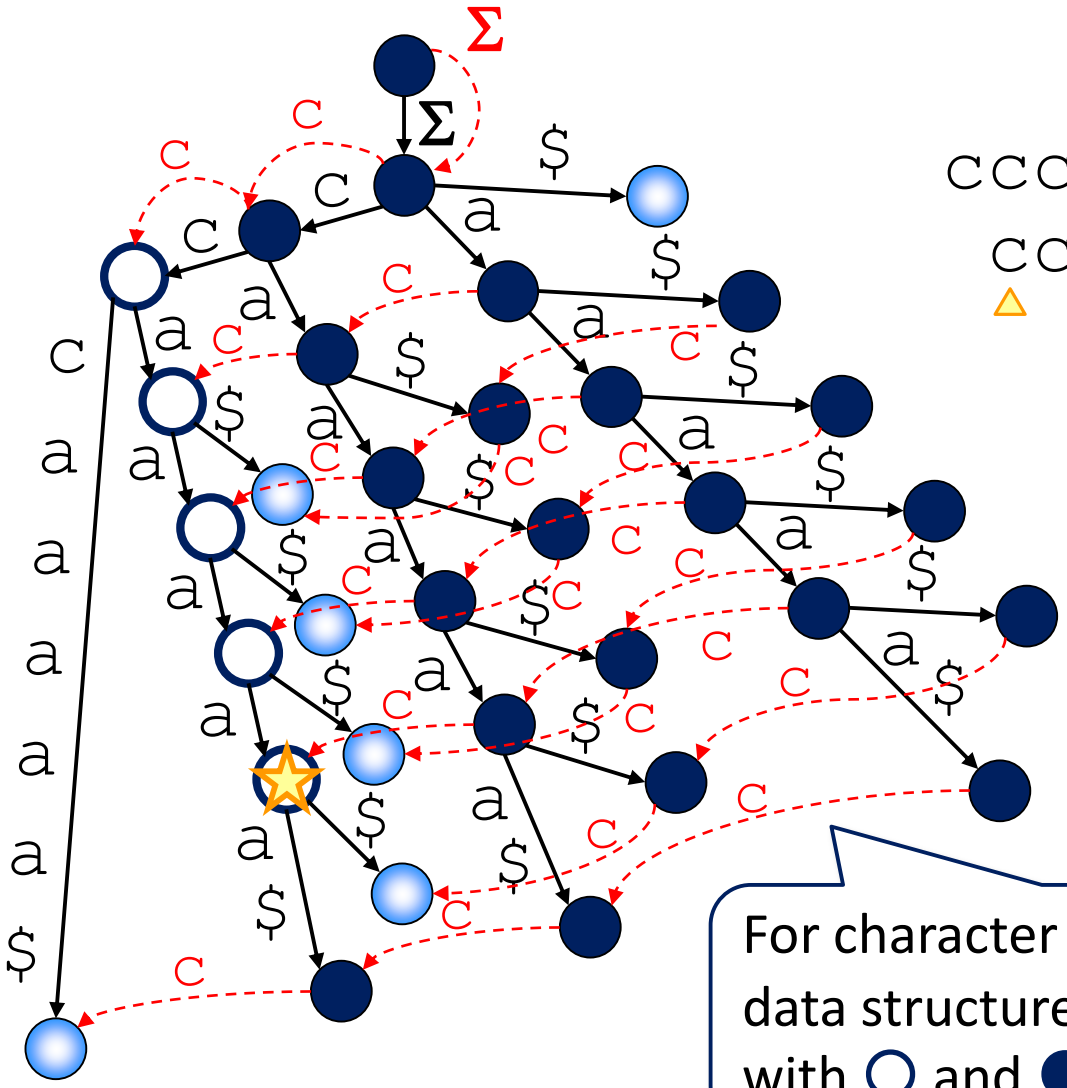
cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

For character c, we build NMA data structure for induced tree with ○ and ●

$O(N)$ space is enough for NMA



ccccaaaaa$ T_1

ccccaaaa$ T_2

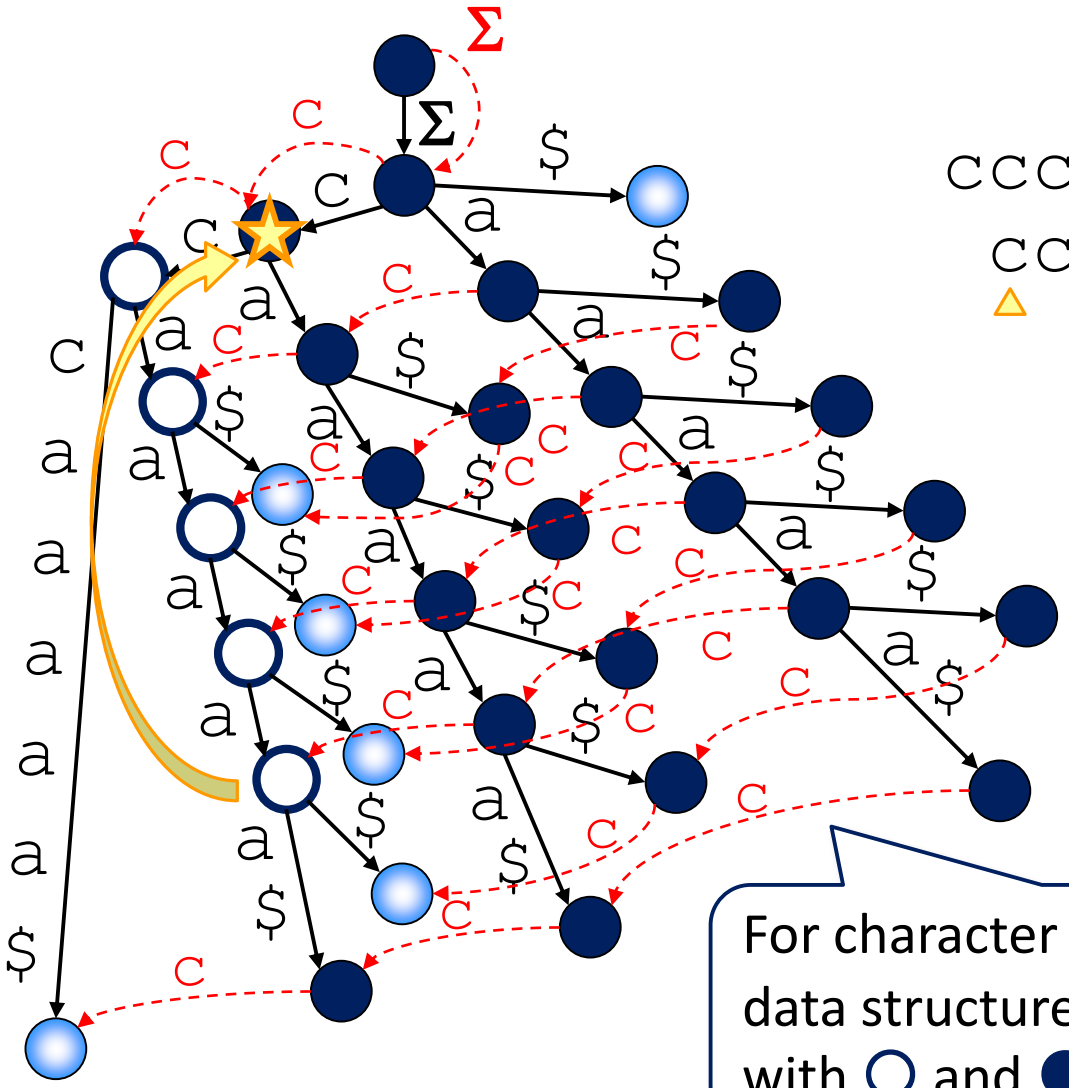
cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

For character c, we build NMA data structure for induced tree with ○ and ●

$O(N)$ space is enough for NMA



ccccaaaaa$ T_1

ccccaaaa$ T_2

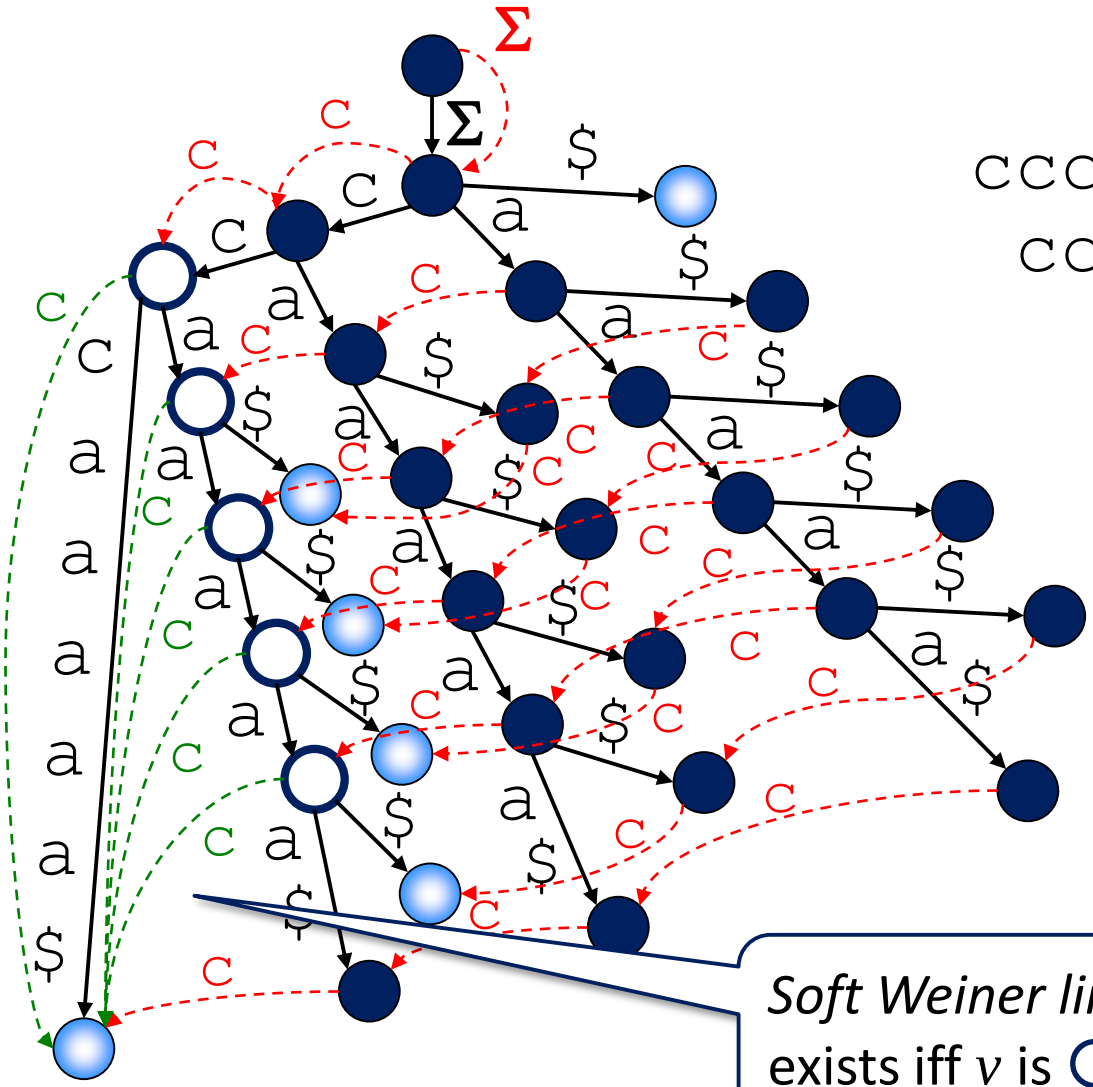
cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

For character c, we build NMA data structure for induced tree with ○ and ●

$O(N)$ space is enough for NMA



ccccaaaaa$ T_1
ccccaaaa$ T_2
cccaaa$ T_3
cccaa$ T_4
ccca$ T_5

Soft Weiner link from *v* with c exists iff *v* is ○ for c

$O(N)$ space is enough for NMA

Lemma [Blumer et al., 1985 & 1987]

# of hard Weiner links is $\leq 2N$-1, and
# of soft Weiner links is $\leq N$-1.

Corollary

Our NMA data structure on top of
Weiner's suffix tree requires $O(N)$ space.

Right-to-left fully-online suffix tree

Theorem 1

Weiner's right-to-left suffix tree algorithm for fully-online multiple texts needs to visit $\Omega(N^{1.5})$ nodes, and this bound is tight ($O(N^{1.5})$ in the worst case).

Theorem 2

With the aid of NMA, the suffix tree of multiple texts of total length N can be built in *right-to-left* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

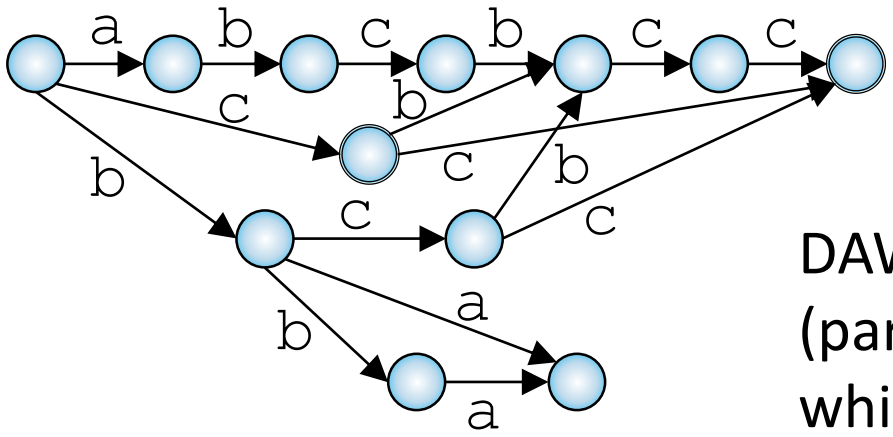
σ is the alphabet size

Left-to-right DAWG

Claim 2

DAWG (suffix automaton) of multiple texts of total length N can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

σ is the alphabet size



$DAWG(\texttt{abcbcc}, \texttt{bba})$

DAWG of multiple texts is (partial) DFA of size $O(N)$ which accepts all substrings of texts.

Fully-online DAWG construction

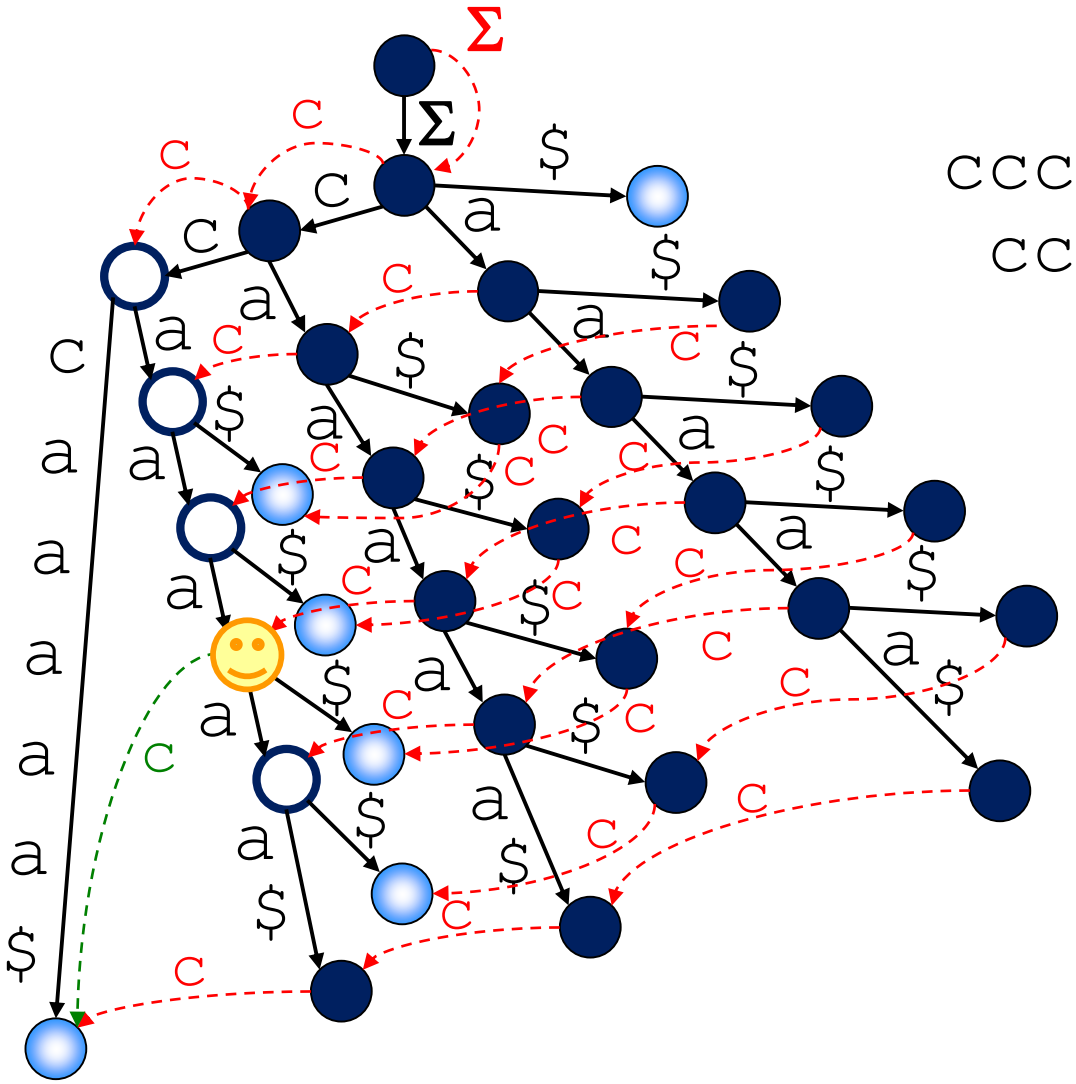
Theorem 3

Blumer at al.'s left-to-right DAWG algorithm for fully-online multiple texts needs to update $\Omega(N^{1.5})$ edges, and this bound is tight ($O(N^{1.5})$ in the worst case).

- Hard and soft Weiner links of texts form DAWG of reversed texts.
- Previous example requires $\Omega(N^{1.5})$ updates for soft Weiner links.

➔ We cannot maintain soft Weiner links explicitly…

Simulating soft W-link with NMA



ccccaaaaa$ T_1

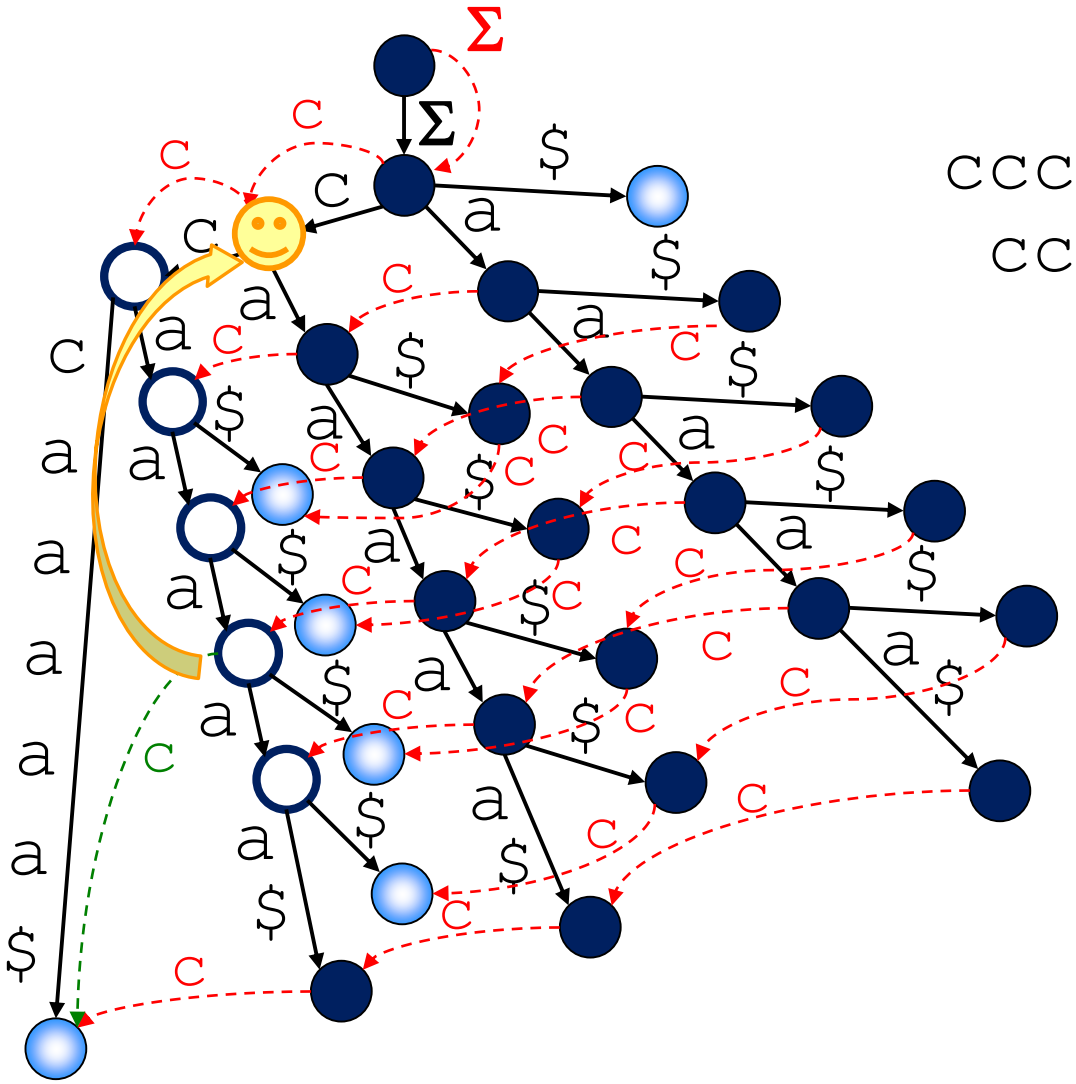
ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Simulating soft W-link with NMA



ccccaaaaa$ T_1

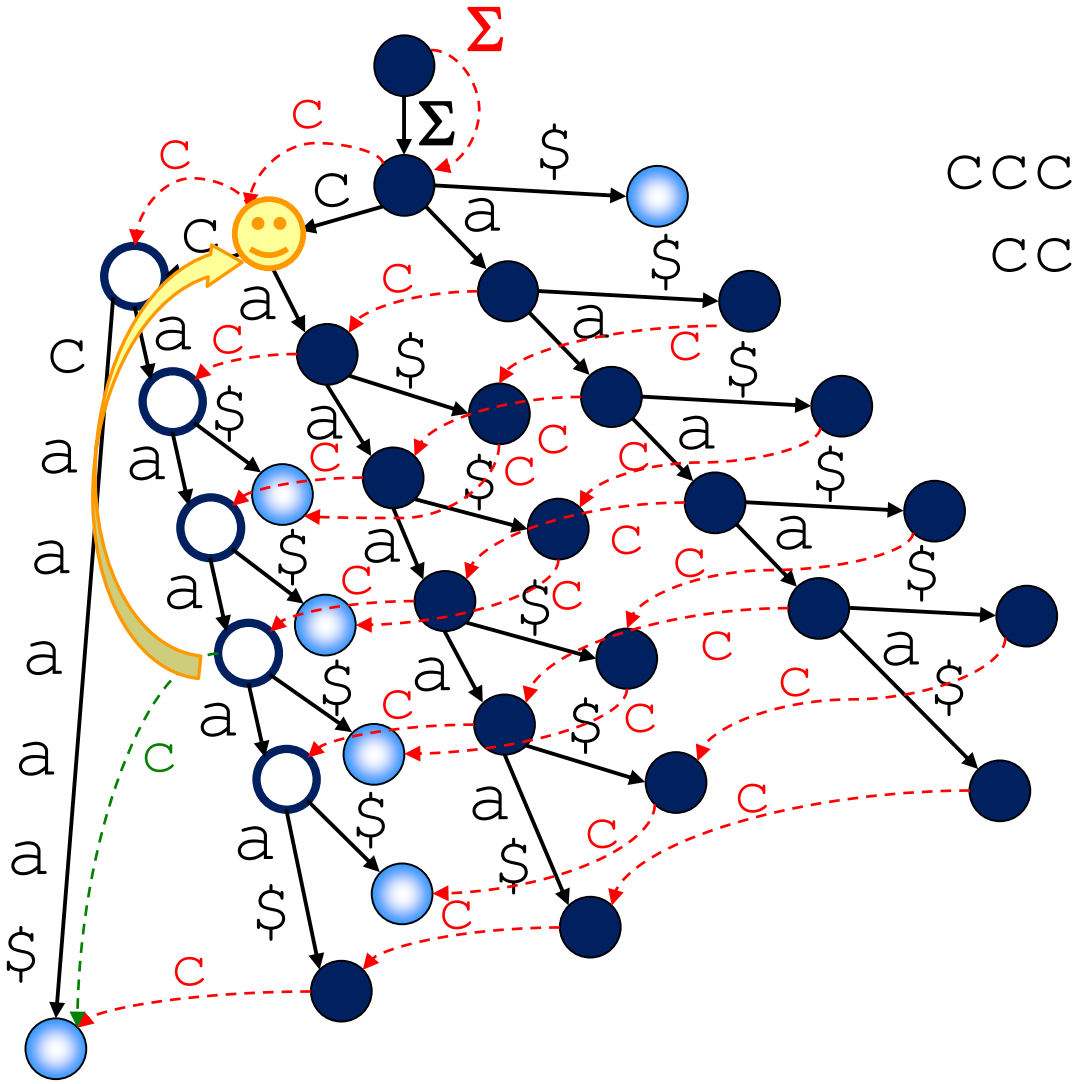
ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Simulating soft W-link with NMA



ccccaaaaa$ T_1

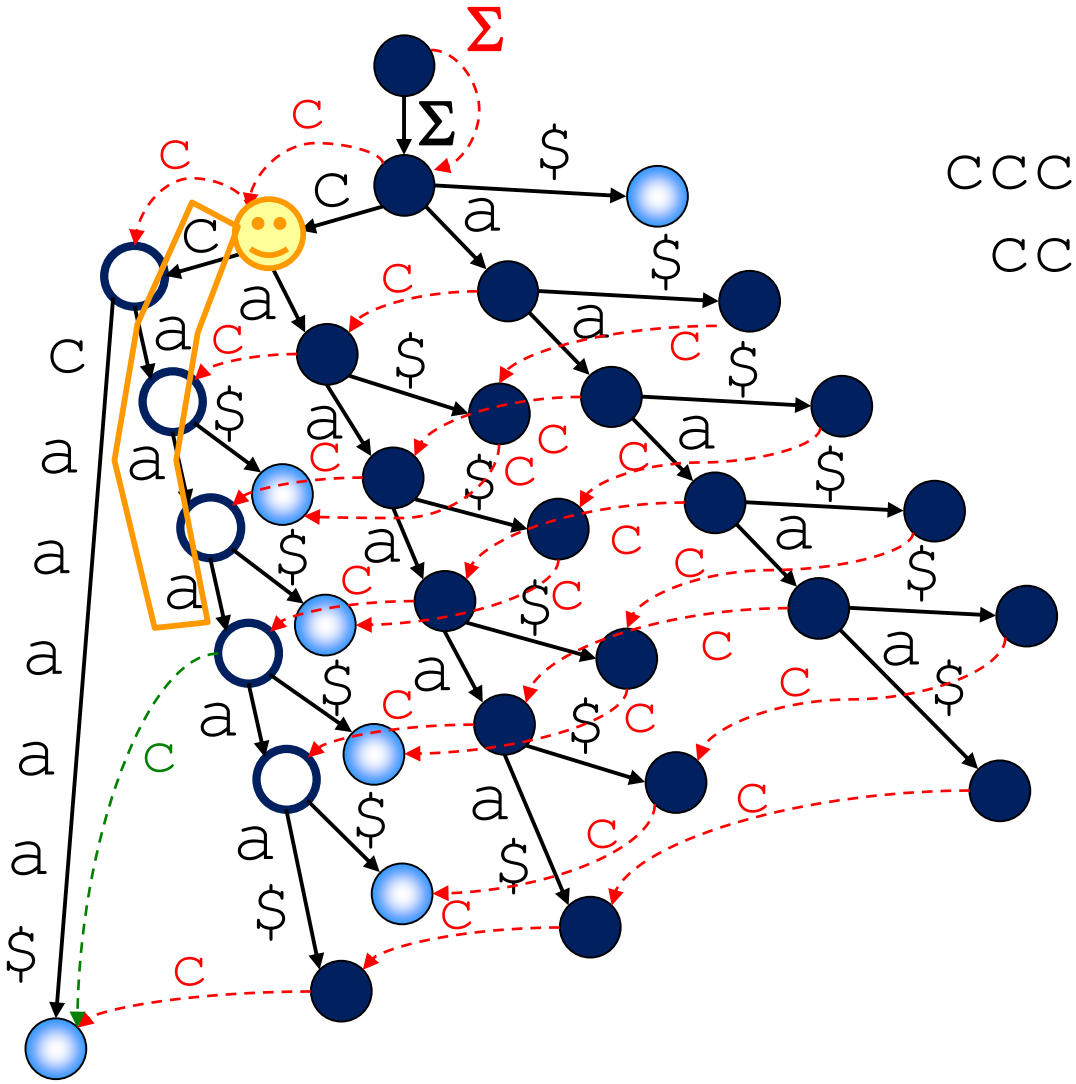
ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Simulating soft W-link with NMA



ccccaaaaa$ T_1

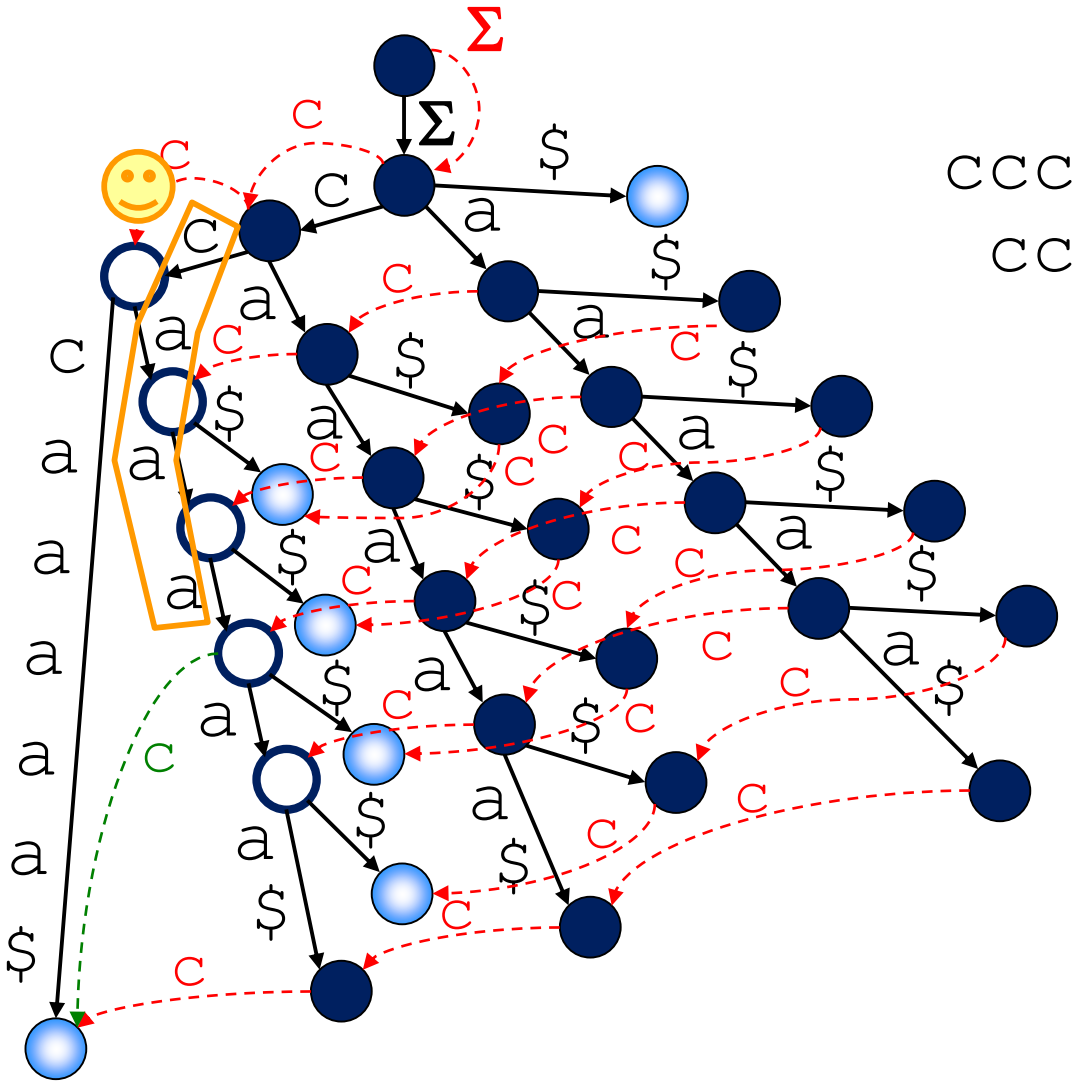
ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Simulating soft W-link with NMA



ccccaaaaa$ T_1

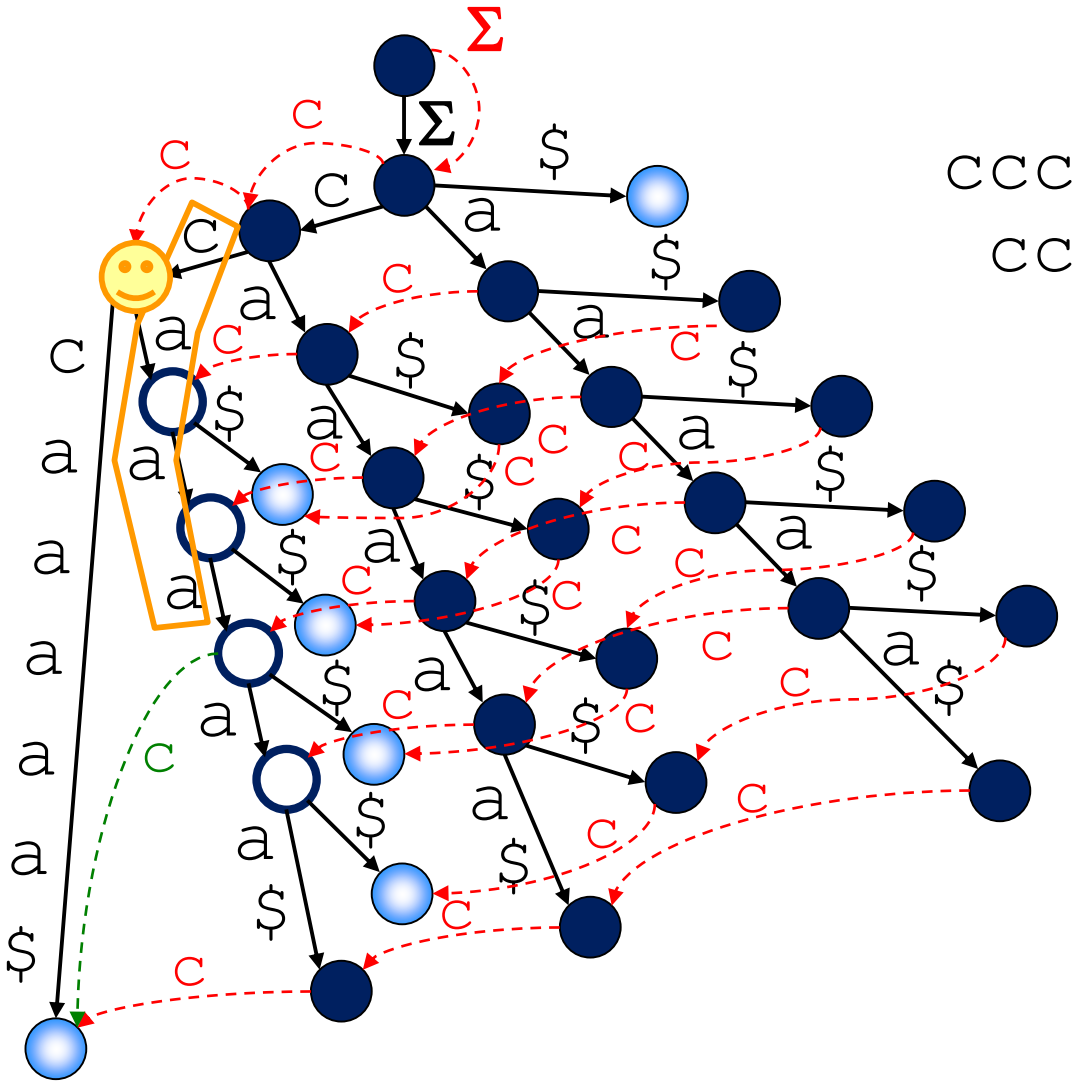
ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Simulating soft W-link with NMA



ccccaaaaa$ T_1

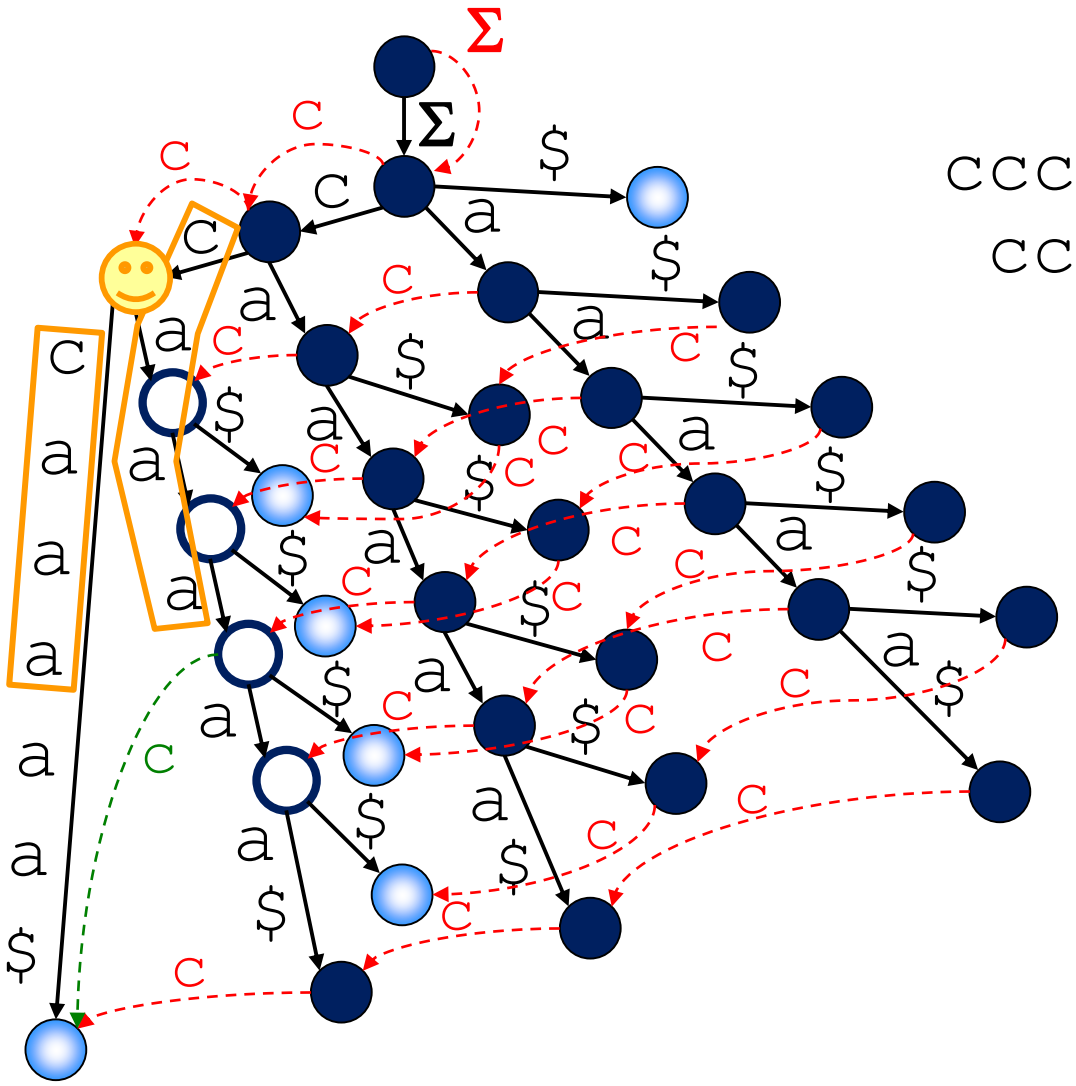
ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Simulating soft W-link with NMA



ccccaaaaa$ T_1

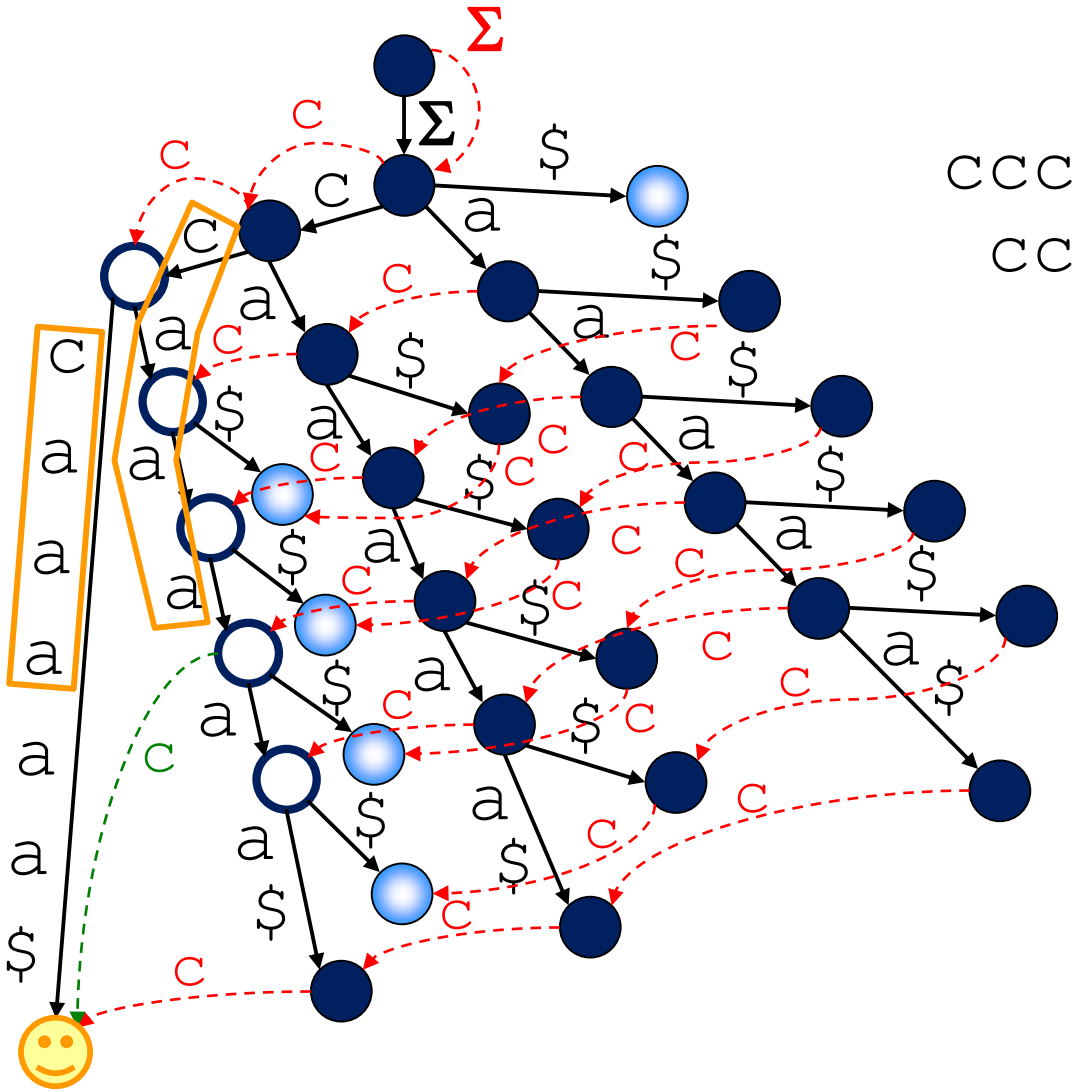
ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Simulating soft W-link with NMA



ccccaaaaa$ T_1

ccccaaaa$ T_2

cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Fully-online DAWG construction

Theorem 3

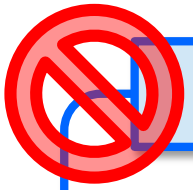
Blumer at al.'s left-to-right DAWG algorithm for fully-online multiple texts needs to update $\Omega(N^{1.5})$ edges, and this bound is tight ($O(N^{1.5})$ in the worst case).

Theorem 4

Implicit representation of DAWG of multiple texts of total length N can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space.

σ is the alphabet size

Left-to-right suffix tree



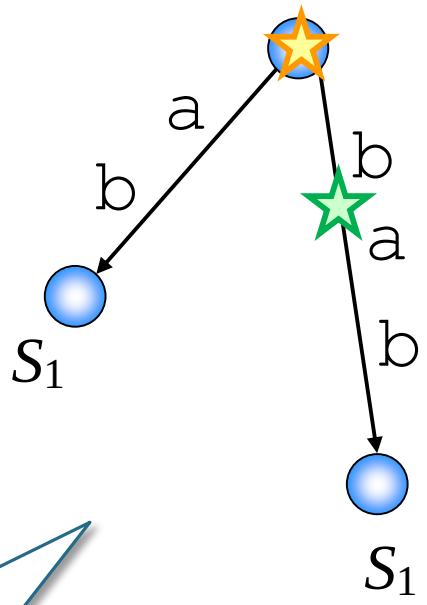
Claim 3

Suffix tree of multiple texts of total length N *without leaf edge labels* can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space, with the aid of DAWG.

σ is the alphabet size

Left-to-right fully-online suffix tree

S_1 bab
S_2 ε

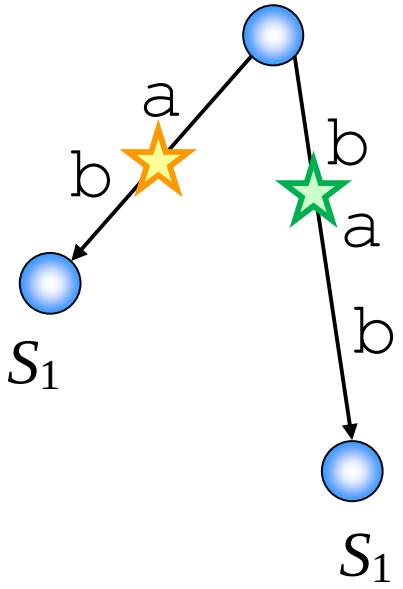


S_1 is currently the owner of both leaves

active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

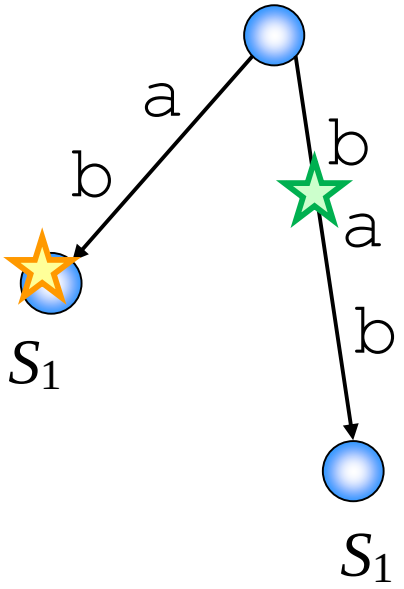
S_1 bab
S_2 a



active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

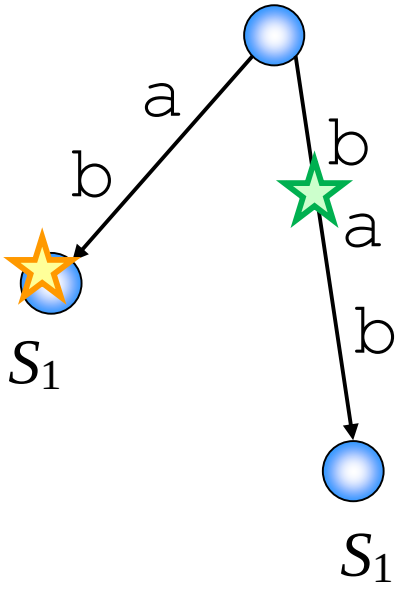
S_1 bab
S_2 ab



active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

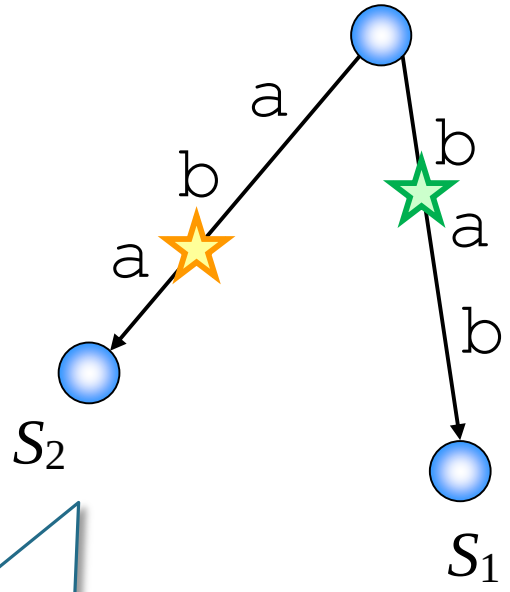
S_1 bab
S_2 aba



active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

S_1 bab
S_2 aba

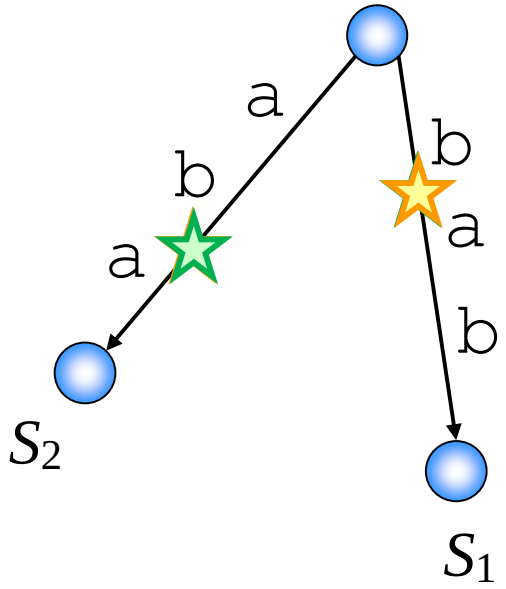


S_2 took the ownership of this leaf!

active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

S_1 bab
S_2 aba

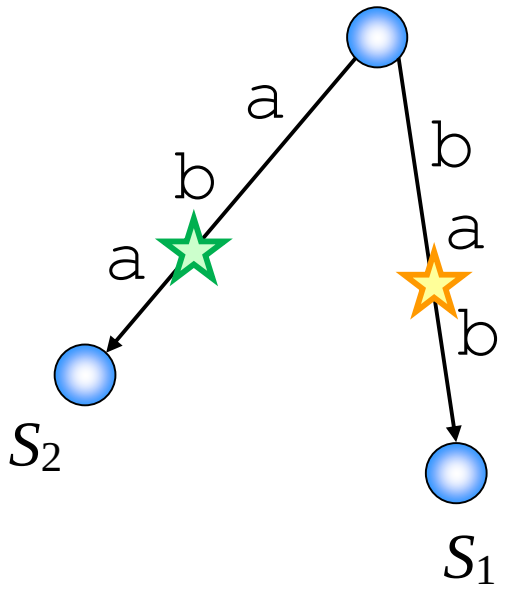


We swap the active points!

active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

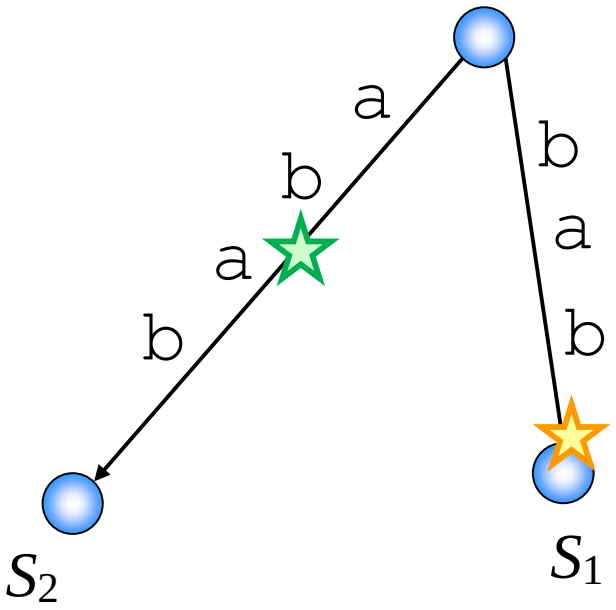
S_1 bab
S_2 aba



active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

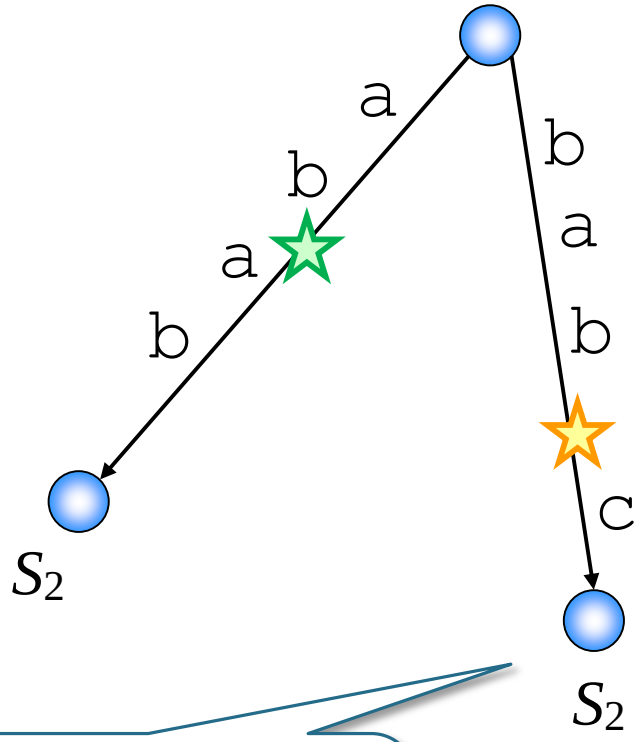
S_1 bab
S_2 abab



active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

S_1 bab
S_2 ababc

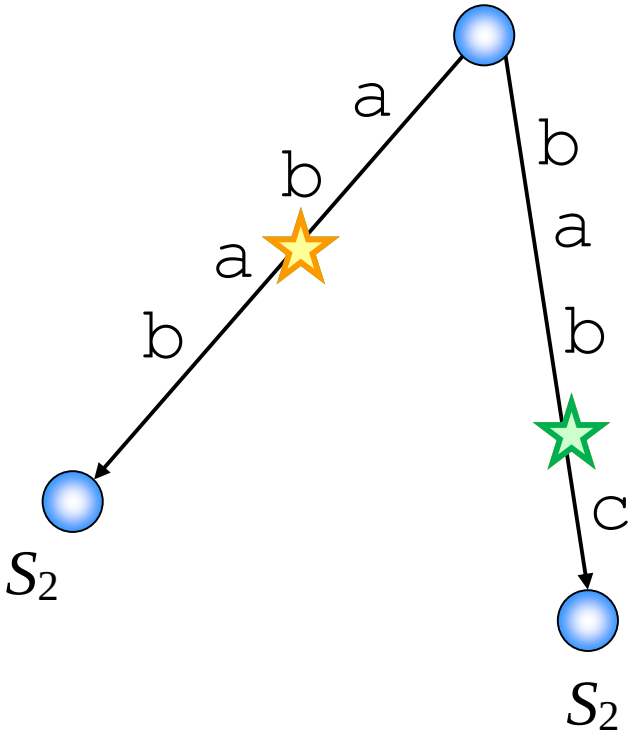


S_2 took the ownership of this leaf, too!

active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

S_1 bab
S_2 ababc

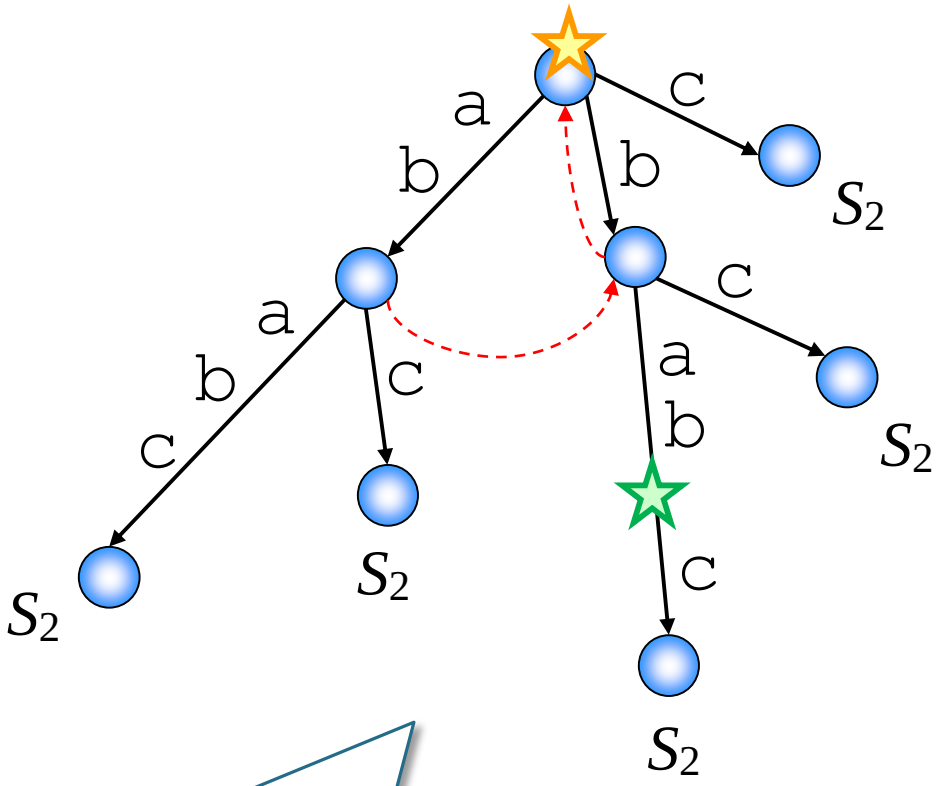


We swap the active points!

active point for S_1
active point for S_2

Left-to-right fully-online suffix tree

S_1 bab
S_2 ababc



S_2 is now the owner of all leaves

active point for S_1
active point for S_2

Leaf ownership & active point swaps

… March, 2017

When the active point of a text extends a leaf,
then we need to know the owner of that leaf.
Also, we need to locate active points to swap.

Takuya Takagi

Hiroki Arimura

Me

Danny Breslauer

Leaf ownership & active point swaps

… March, 2017

But how?

No idea…

Takuya Takagi

Hiroki Arimura

Me

Danny Breslauer

Final team!!

… March, 2017

I know how to do it!
We can do this using information from Weiner's tree.

Diptarama Hendrian

Takuya Takagi

Hiroki Arimura

Me

Danny Breslauer

The new paper

Computer Science > Data Structures and Algorithms

Fully-Online Suffix Tree and Directed Acyclic Word Graph Construction for Multiple Texts

Takuya Takagi, Shunsuke Inenaga, Hiroki Arimura, Dany Breslauer, Diptarama Hendrian

(Submitted on 28 Jul 2015 (v1), last revised 12 Jul 2018 (this version, v5))

We consider construction of the suffix tree and the directed acyclic word graph (DAWG) indexing data structures for a collection $\mathcal{T}$ of texts, where a new symbol may be appended to any text in $\mathcal{T} = \{T_1, \ldots, T_K\}$, at any time. This fully-online scenario, which arises in dynamically indexing multi-sensor data, is a natural generalization of the long solved semi-online text indexing problem, where texts $T_1, \ldots, T_k$ are permanently fixed before the next text T_{k+1} is processed for each $1 \leq k < K$. We present fully-online algorithms that construct the suffix tree and the DAWG for $\mathcal{T}$ in $O(N \log \sigma)$ time and $O(N)$ space, where N is the total lengths of the strings in $\mathcal{T}$ and σ is their alphabet size. The standard explicit representation of the suffix tree leaf edges and some DAWG edges must be relaxed in our fully-online scenario, since too many updates on these edges are required in the worst case. Instead, we provide access to the updated suffix tree leaf edge labels and the DAWG edges to be redirected via auxiliary data structures, in $O(\log \sigma)$ time per added character.

Left-to-right fully-online suffix tree

Theorem 5

Suffix tree of multiple texts of total length N *without leaf edge labels* can be built in *left-to-right* fully-online manner in $O(N \log \sigma)$ time with $O(N)$ space, with the aid of Weiner's tree.

σ is the alphabet size

Leaf ownership can be computed on-the-fly in $O(\log \sigma)$ time only when leaf is extended.

Leaf ownership in fully-online Ukkonen

Theorem 6

It takes $\Omega(N^2/K^2)$ time to explicitly maintain leaf ownership of Ukkonen's suffix tree for left-to-right fully-online multiple texts.
Thus it takes $\Omega(N^2)$ time when $K = O(1)$ and $K \geq 2$.

N = total text length
K = # of texts

Open questions

- Linear-size data structure which can efficiently answer ownership of any leaf per query?
  - $O(\log N / \log\log N)$ time might be possible via dynamic NMA.
  - Danny believed $O(1)$ or $O(\log \sigma)$ would be possible.
- Stand-alone version of Ukkonen algorithm for left-to-right fully-online multiple texts?
- BWT for fully-online multiple texts?

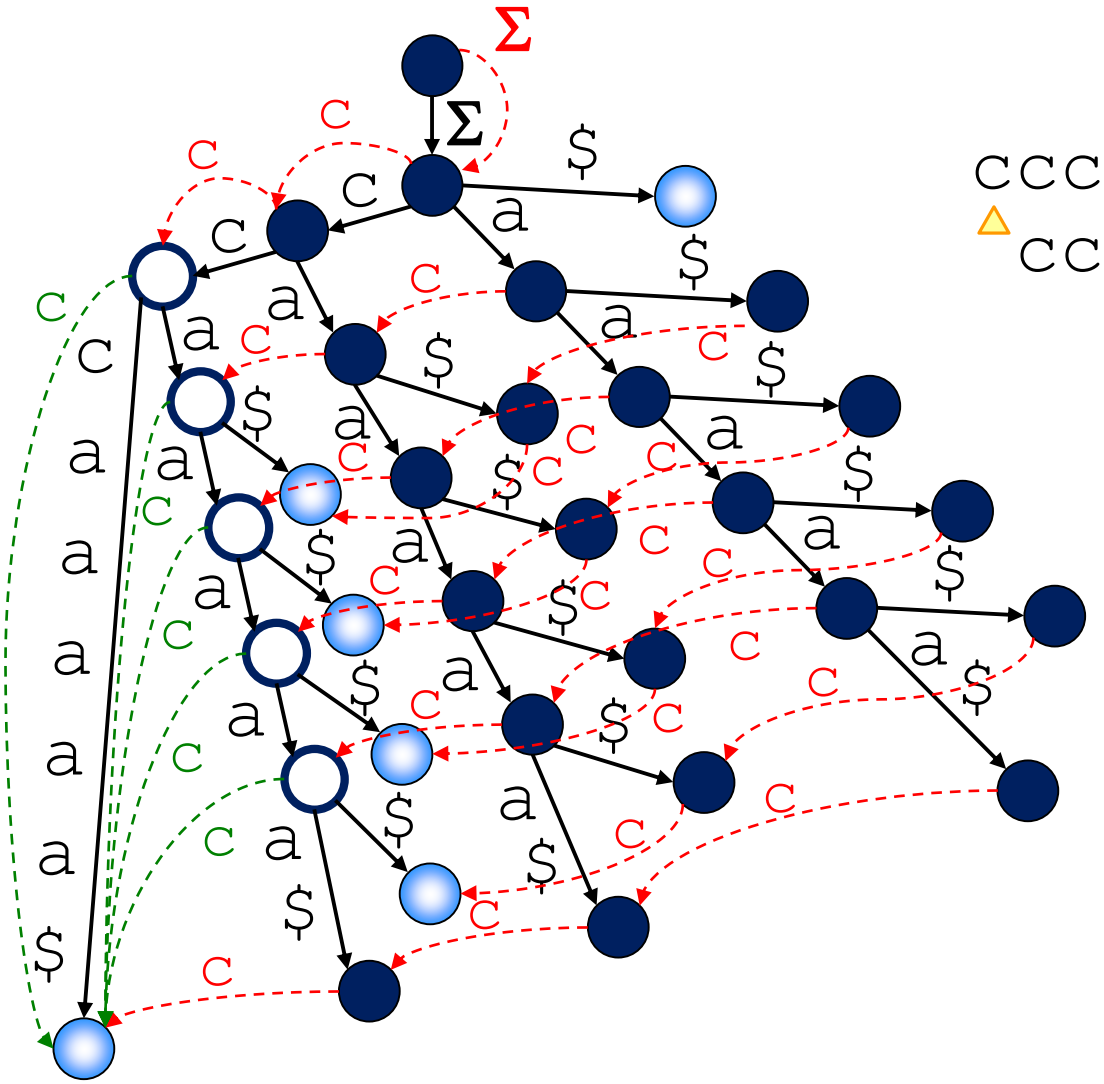
Danny @ Sapporo, June 2017

Appendix A

Theorem 3

Blumer at al.'s left-to-right DAWG algorithm for fully-online multiple texts needs to update $\Omega(N^{1.5})$ edges, and this bound is tight ($O(N^{1.5})$ in the worst case).

$\Omega(N^{1.5})$ soft W-link redirections



ccccaaaaa$ T_1

ccccaaaa$ T_2

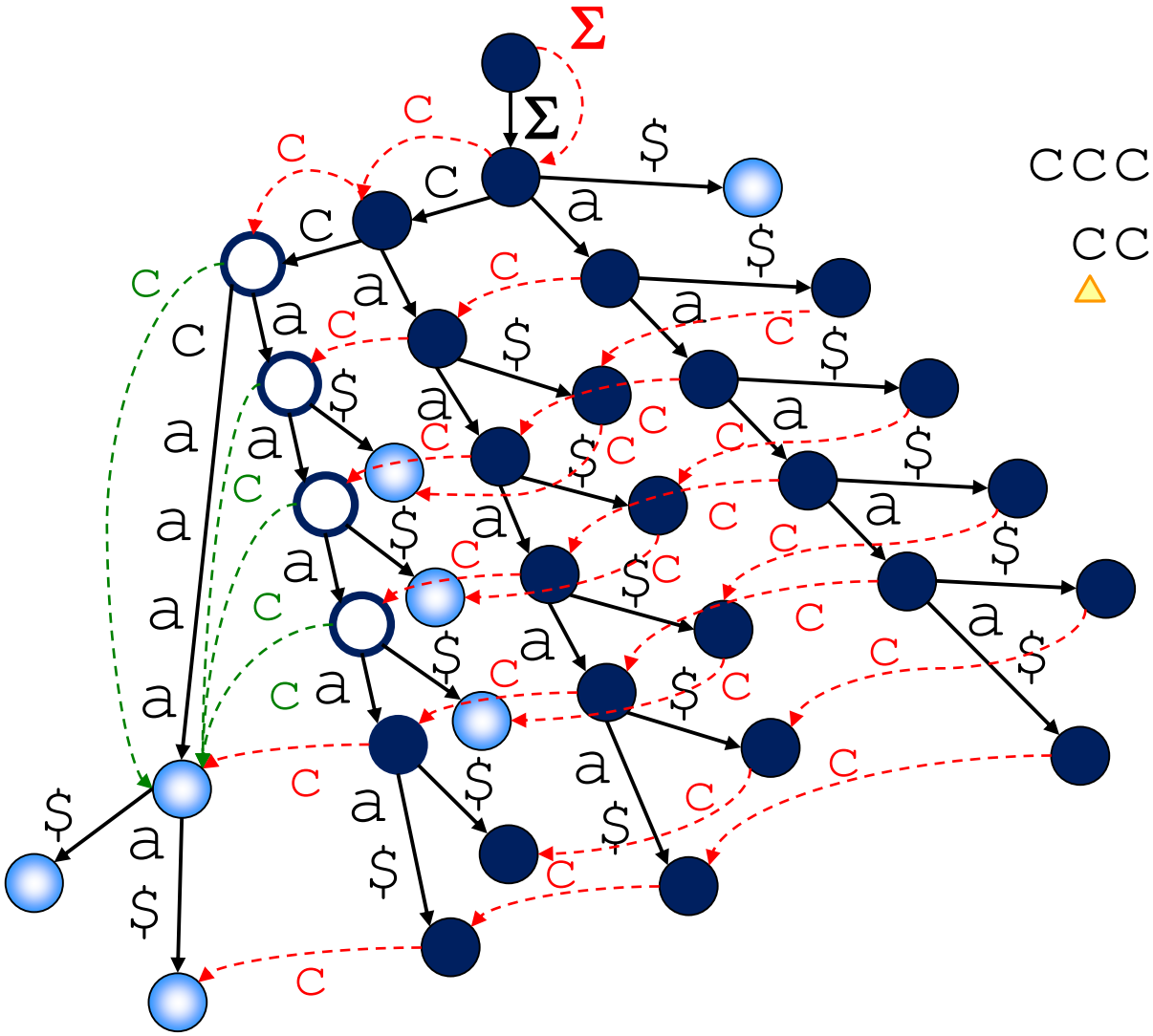
cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Note: some Weiner links are omitted for simplicity

$\Omega(N^{1.5})$ soft W-link redirections



ccccaaaaa$ T_1

ccccaaaa$ T_2

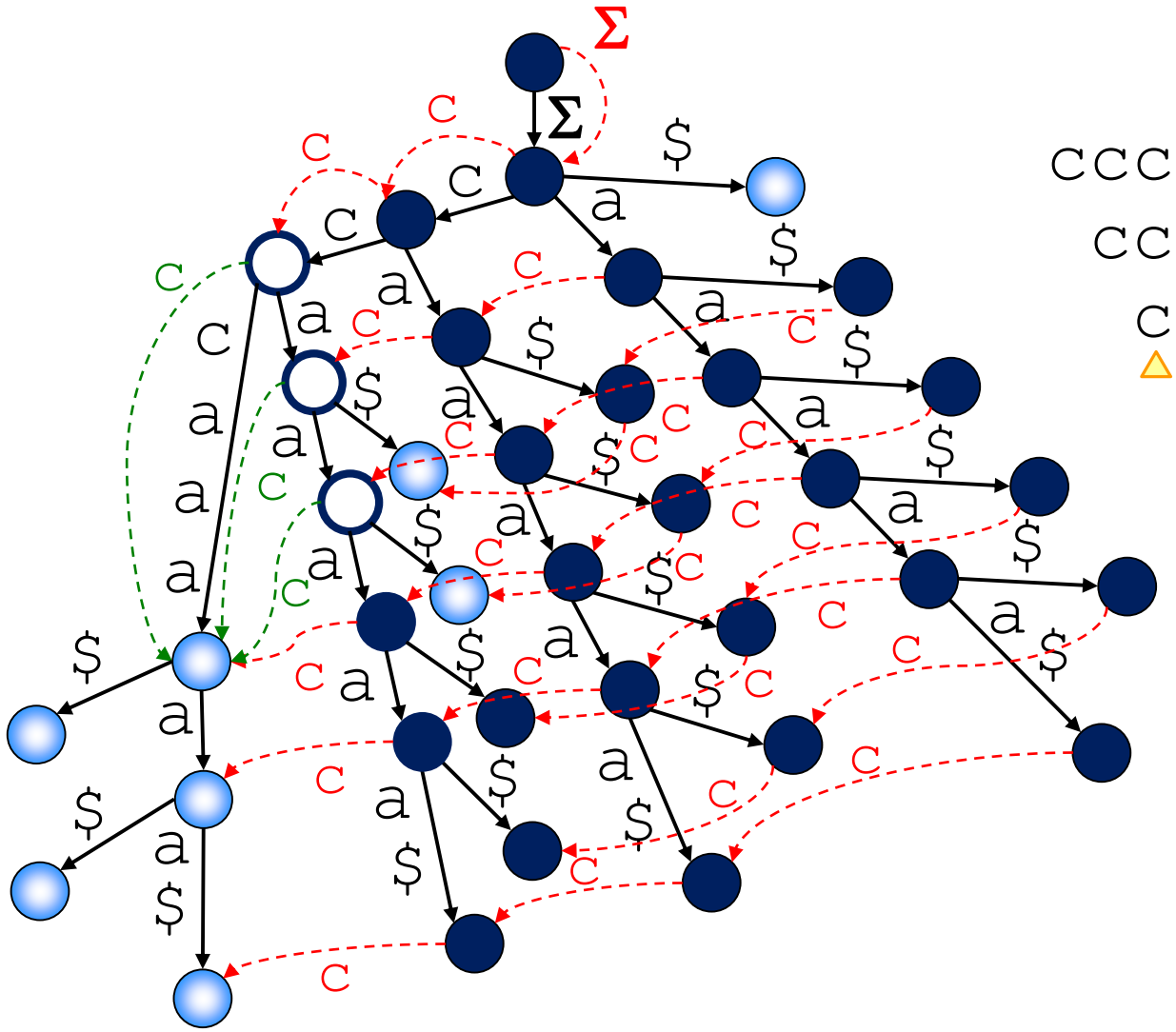
cccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Note: some Weiner links are omitted for simplicity

$\Omega(N^{1.5})$ soft W-link redirections



ccccaaaaa$ T_1

cccaaaa$ T_2

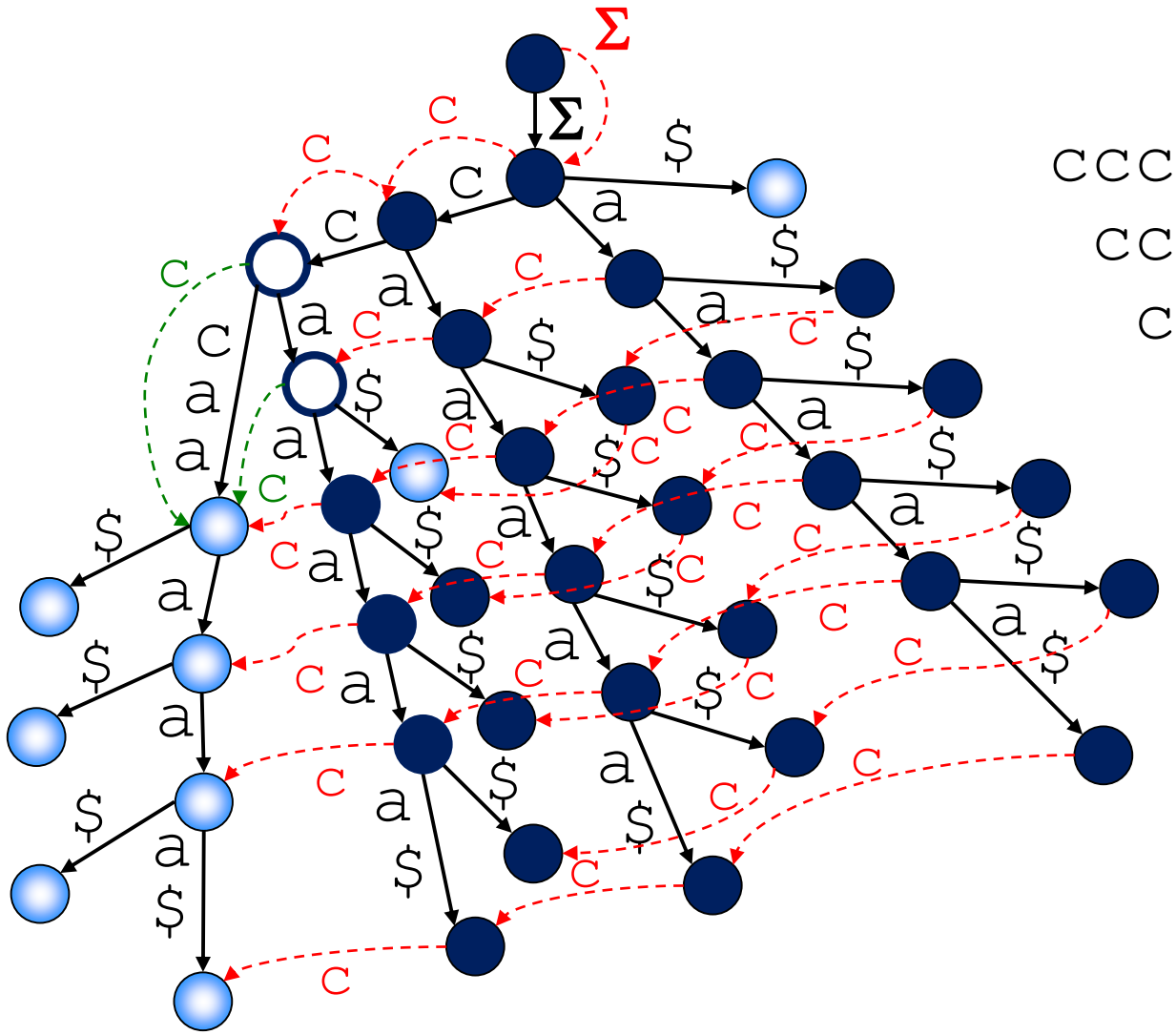
ccccaaa$ T_3

cccaa$ T_4

ccca$ T_5

Note: some Weiner links are omitted for simplicity

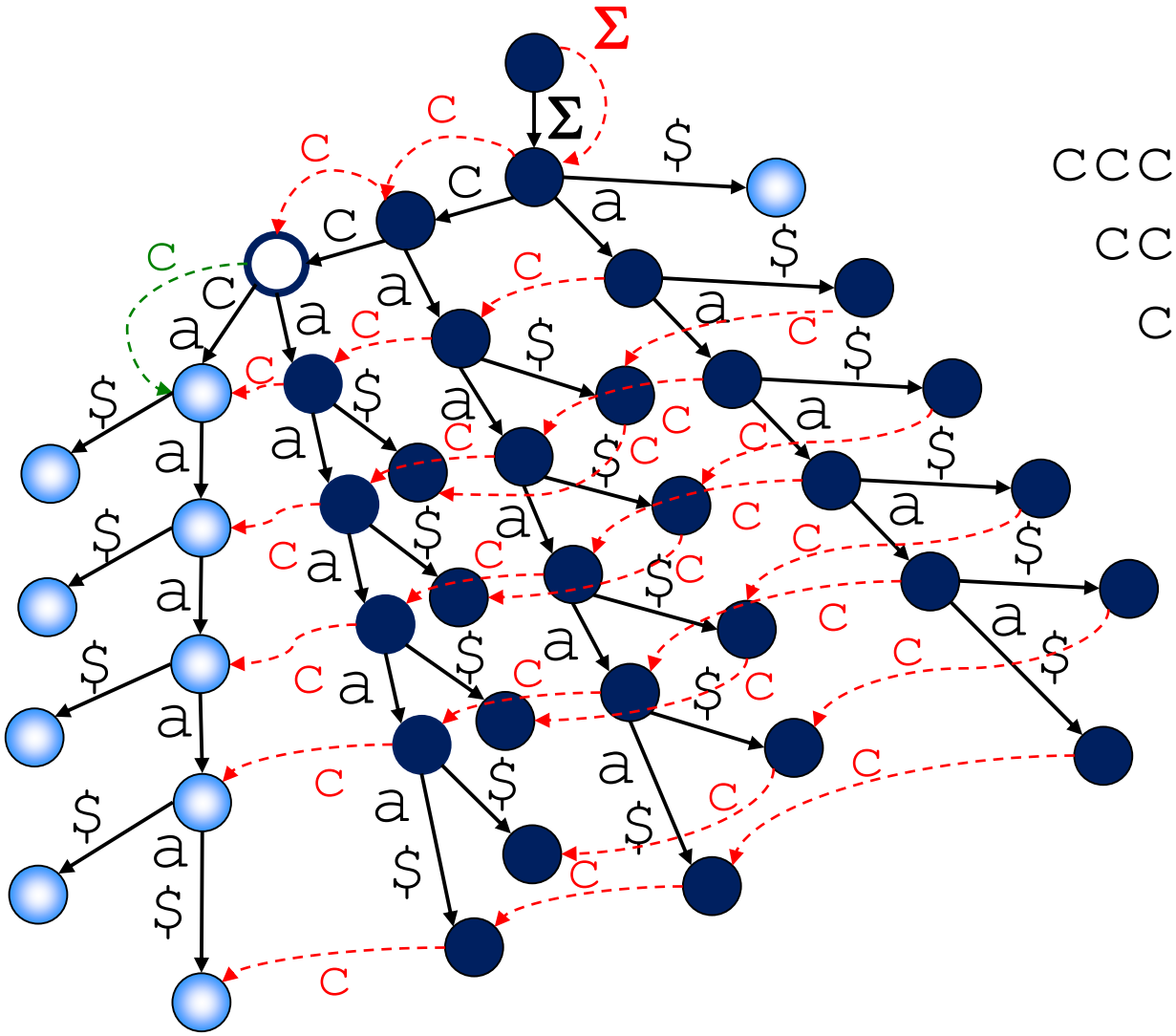
$\Omega(N^{1.5})$ soft W-link redirections



ccccaaaaa$ T_1
cccaaaa$ T_2
ccccaaa$ T_3
ccccaa$ T_4
ccca$ T_5

Note: some Weiner links are omitted for simplicity

$\Omega(N^{1.5})$ soft W-link redirections



ccccaaaaa$ T_1
cccaaaa$ T_2
ccccaaa$ T_3
ccccaa$ T_4
cccca$ T_5

Note: some Weiner links are omitted for simplicity

Appendix B

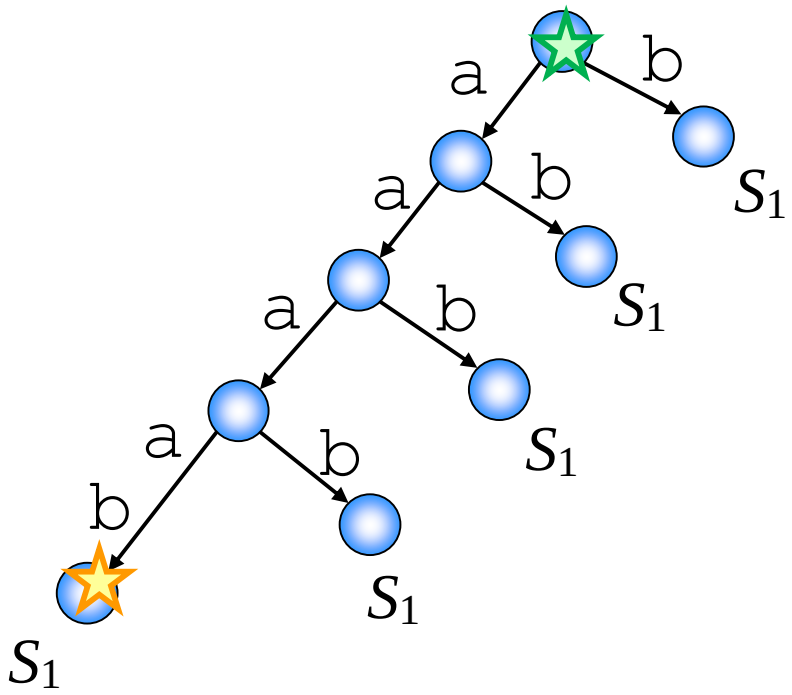
Theorem 6

It takes $\Omega(N^2/K^2)$ time to explicitly maintain leaf ownership of Ukkonen's suffix tree for left-to-right fully-online multiple texts.
Thus it takes $\Omega(N^2)$ time when $K = O(1)$ and $K \geq 2$.

N = total text length
K = # of texts

$\Omega(N^{1.5})$ time for explicit leaf ownership

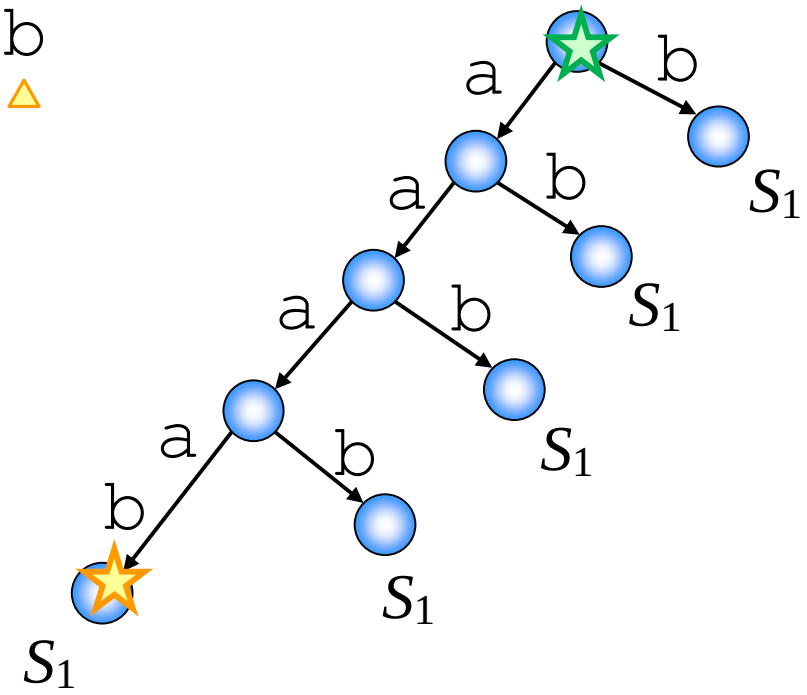
S_1 aaaab
S_2 aaaab



active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

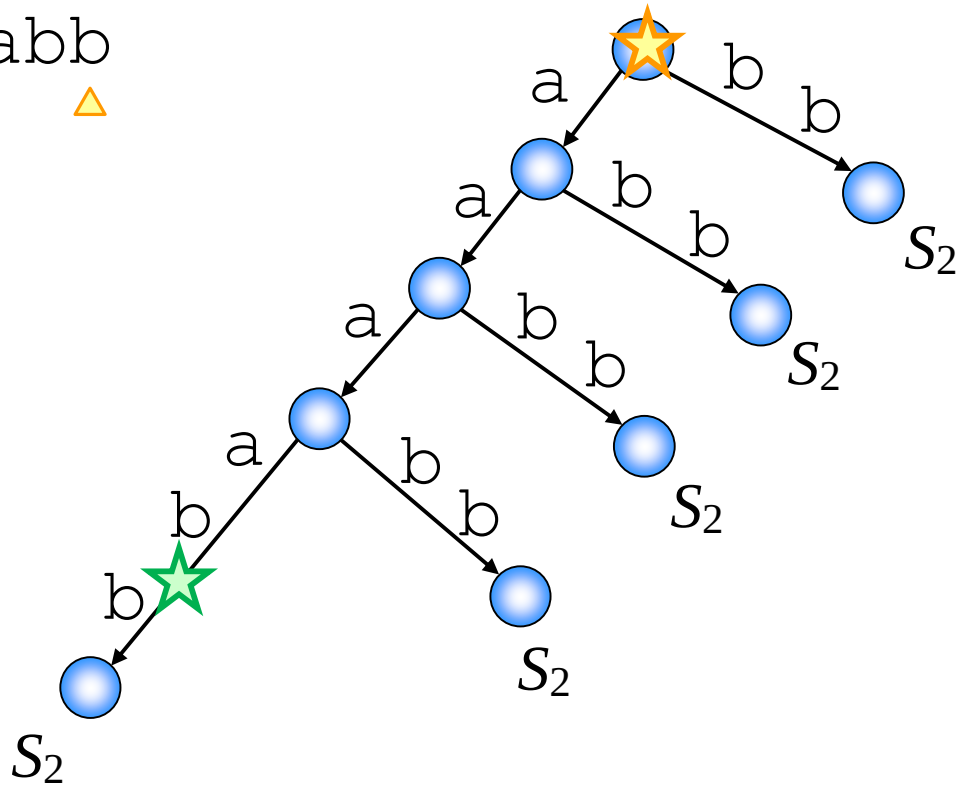
S_1 aaaab
S_2 aaaabb



active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

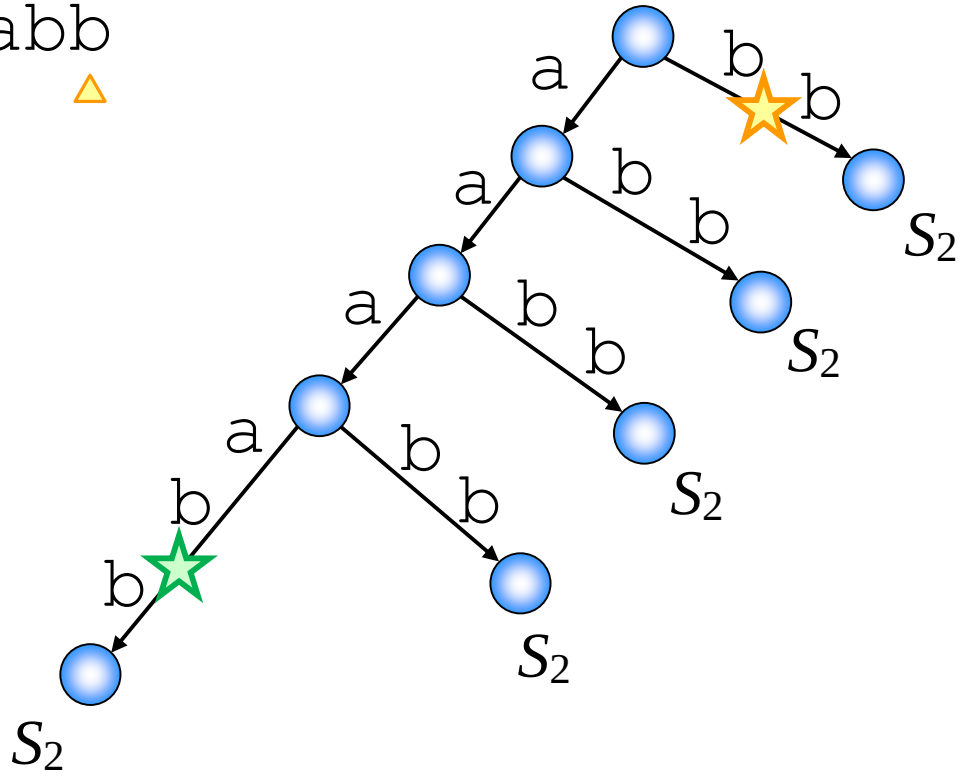
S_1 aaaab
S_2 aaaabb



active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

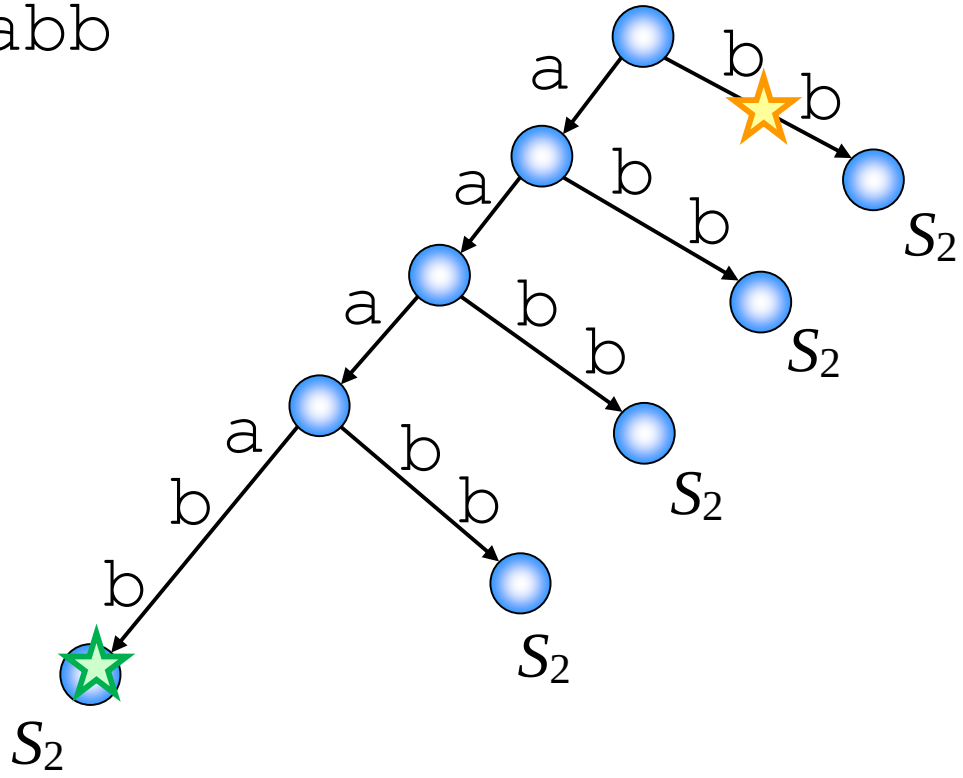
S_1 aaaab
S_2 aaaabb



active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

S_1 aaaabb▽
S_2 aaaabb

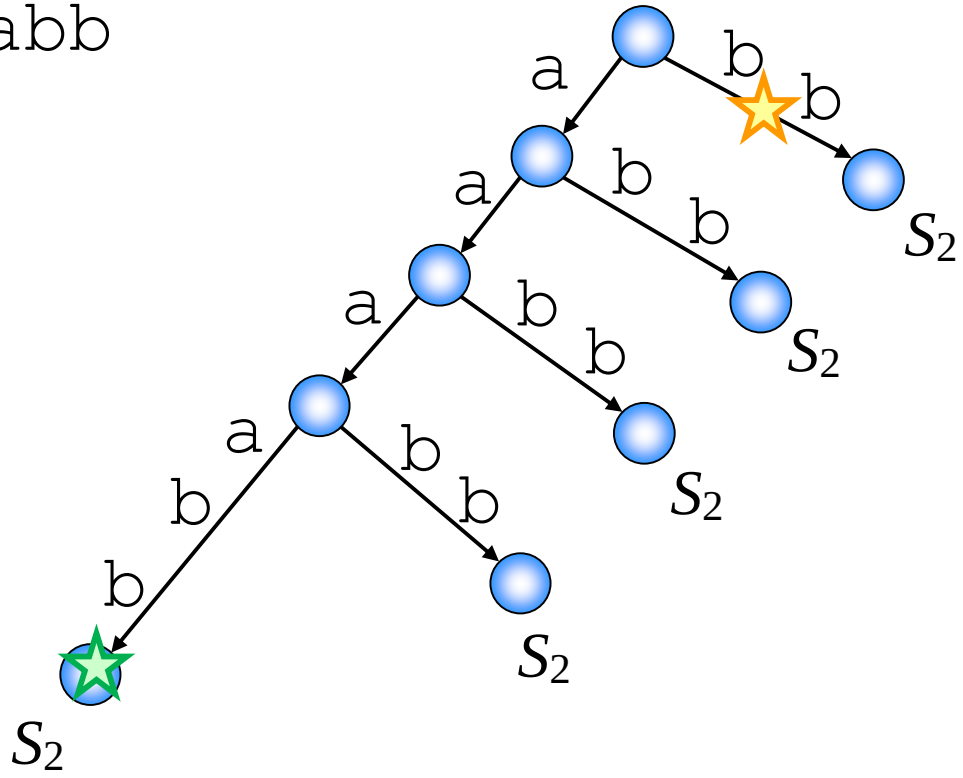


active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

S_1 aaaabbb▽

S_2 aaaabb



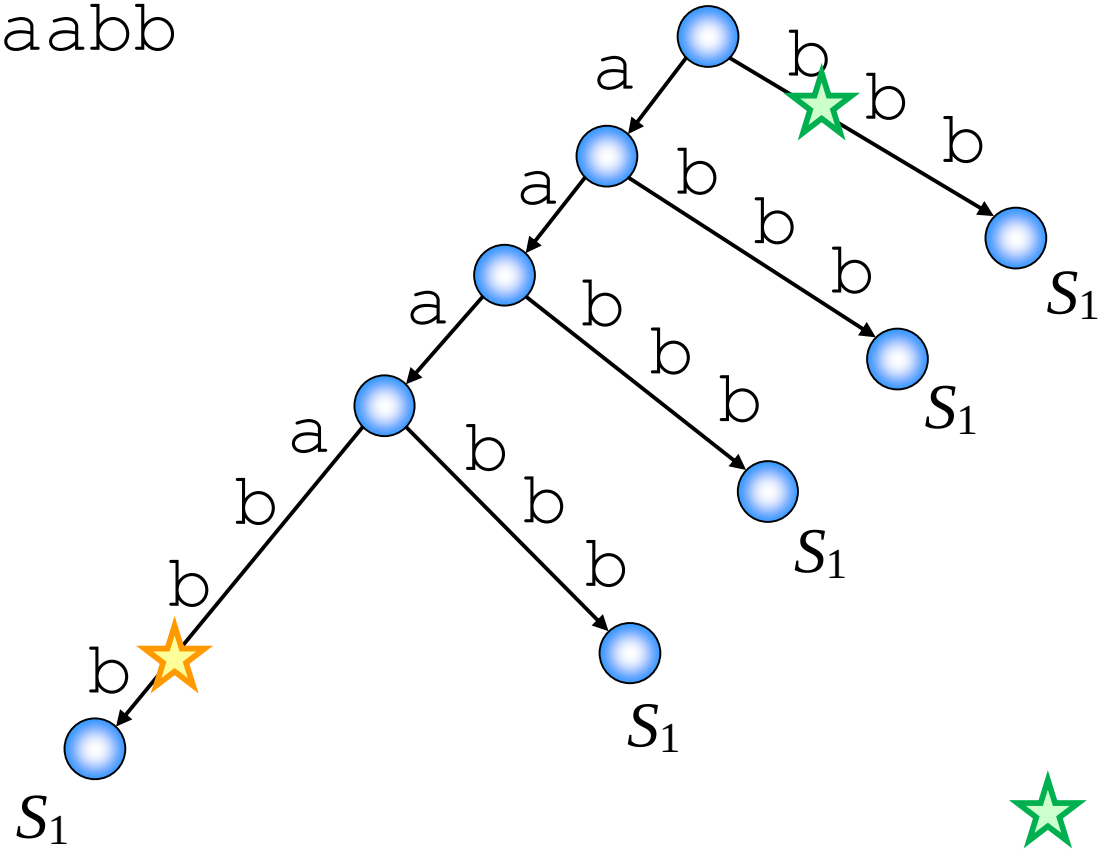
active point for S_1

active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

S_1 aaaabbb▽

S_2 aaaabb



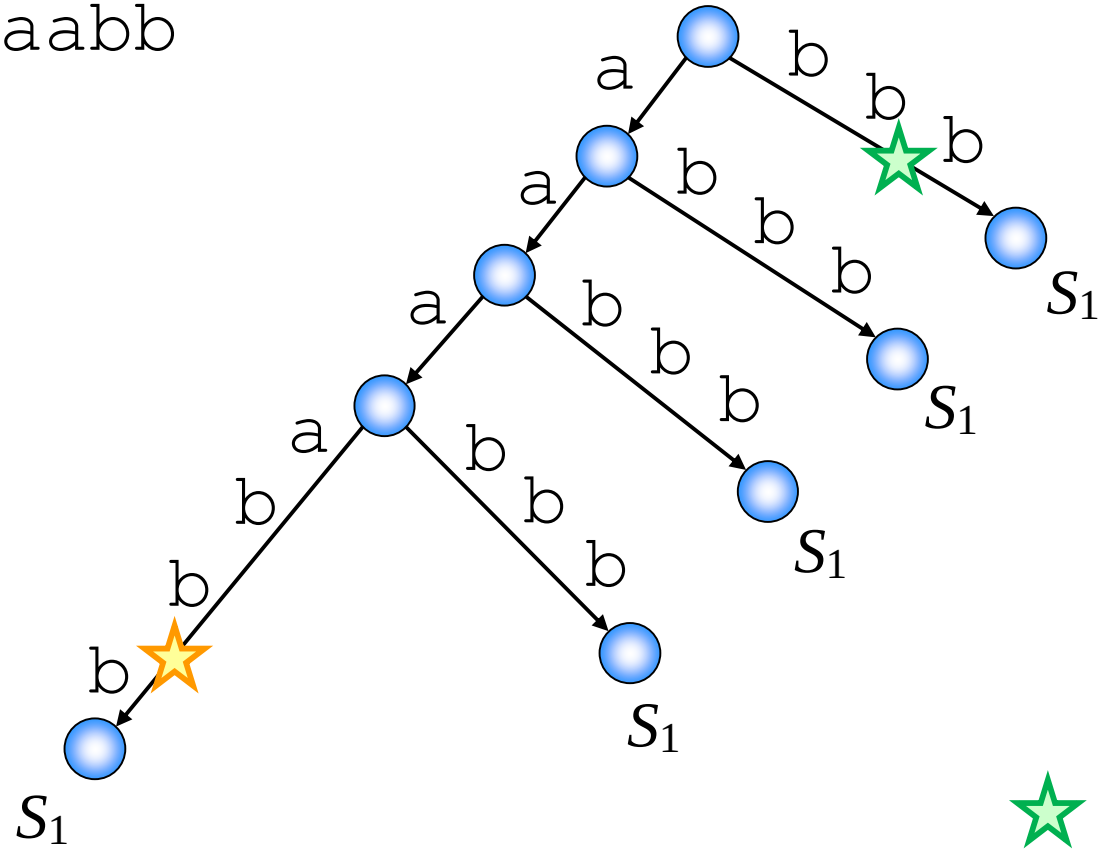
active point for S_1

active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

S_1 aaaabbb▽

S_2 aaaabb

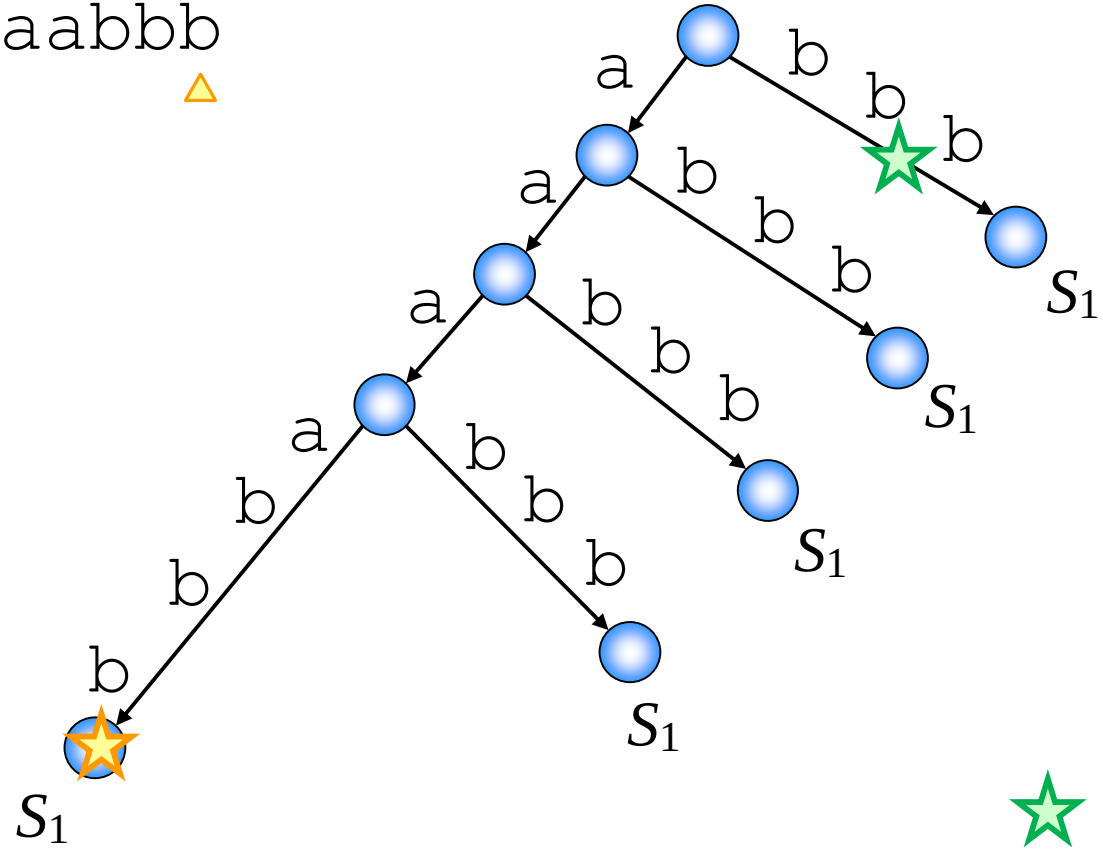


active point for S_1

active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

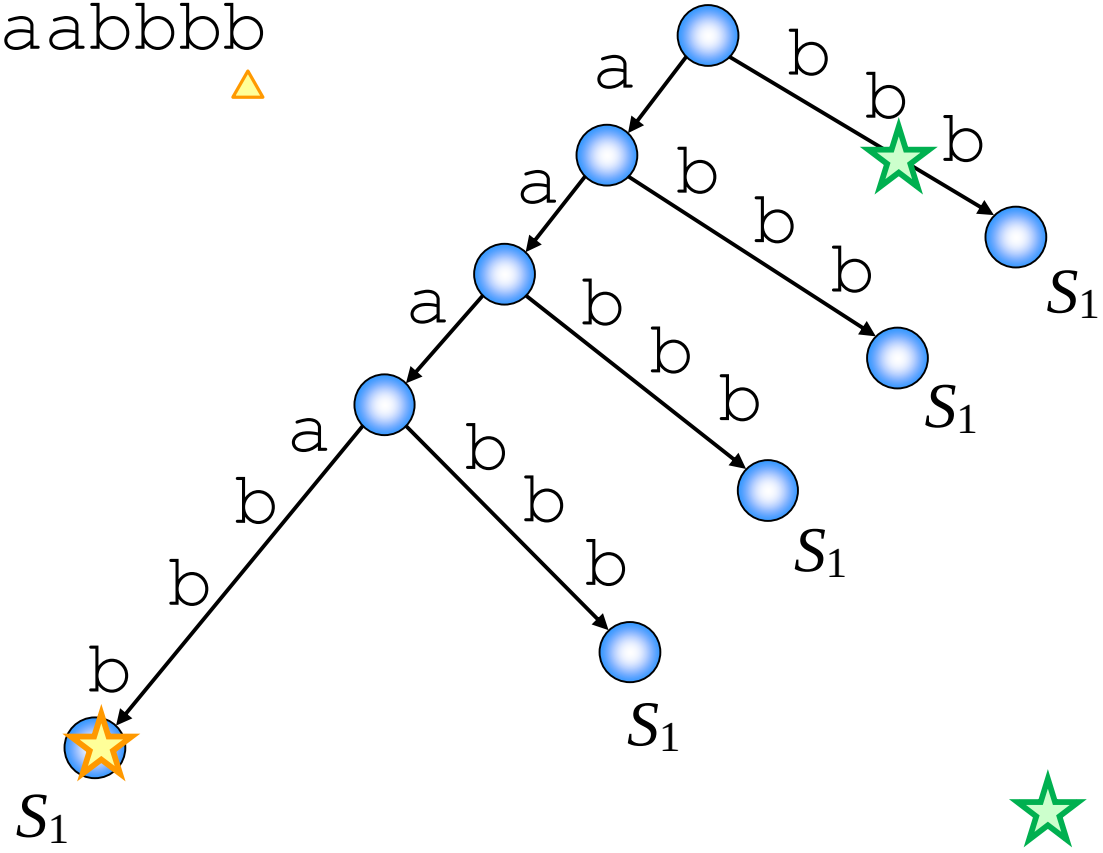
S_1 aaaabbb
S_2 aaaabbb



active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

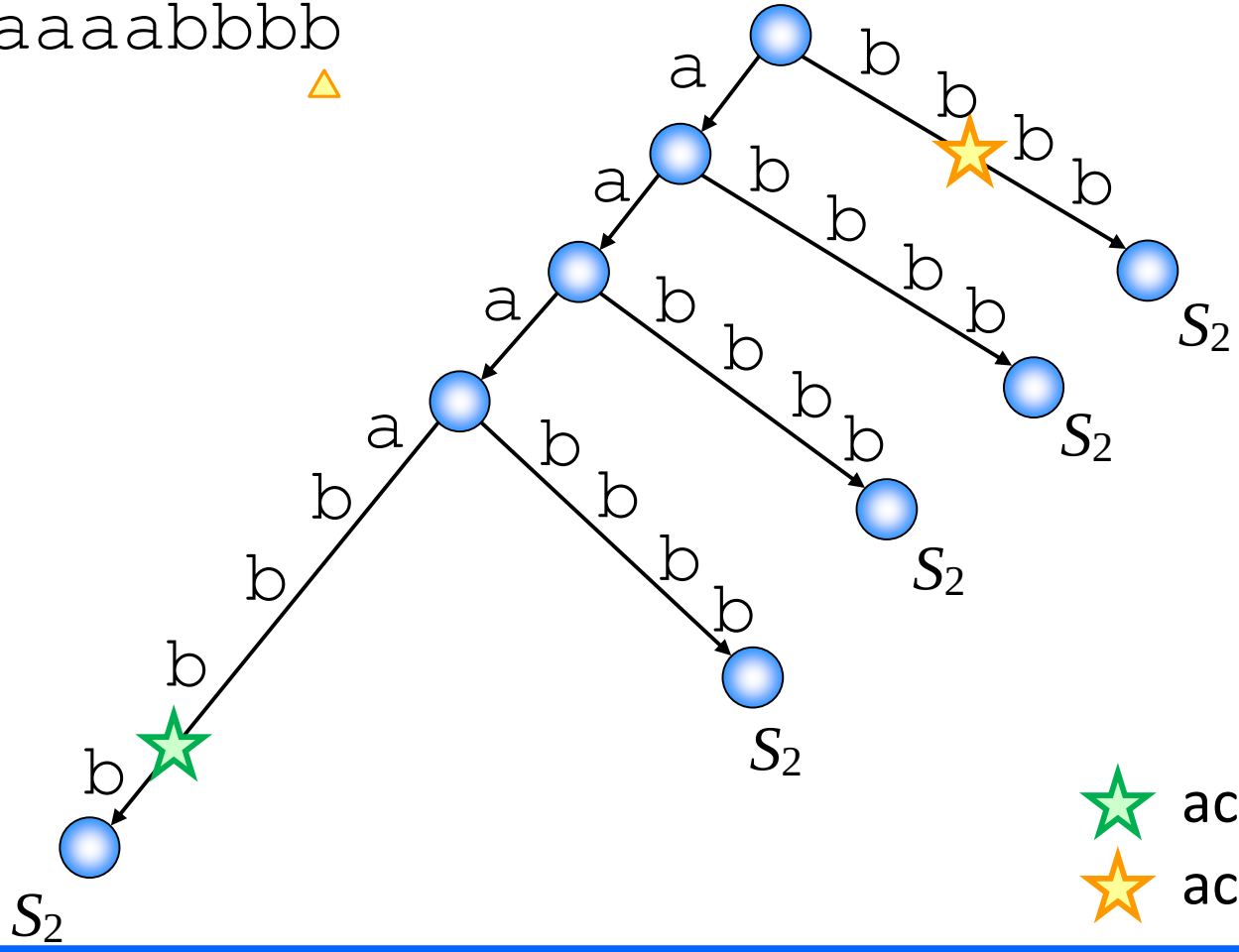
S_1 aaaabbb
S_2 aaaabbbb



active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

S_1 aaaabbb
S_2 aaaabbbb

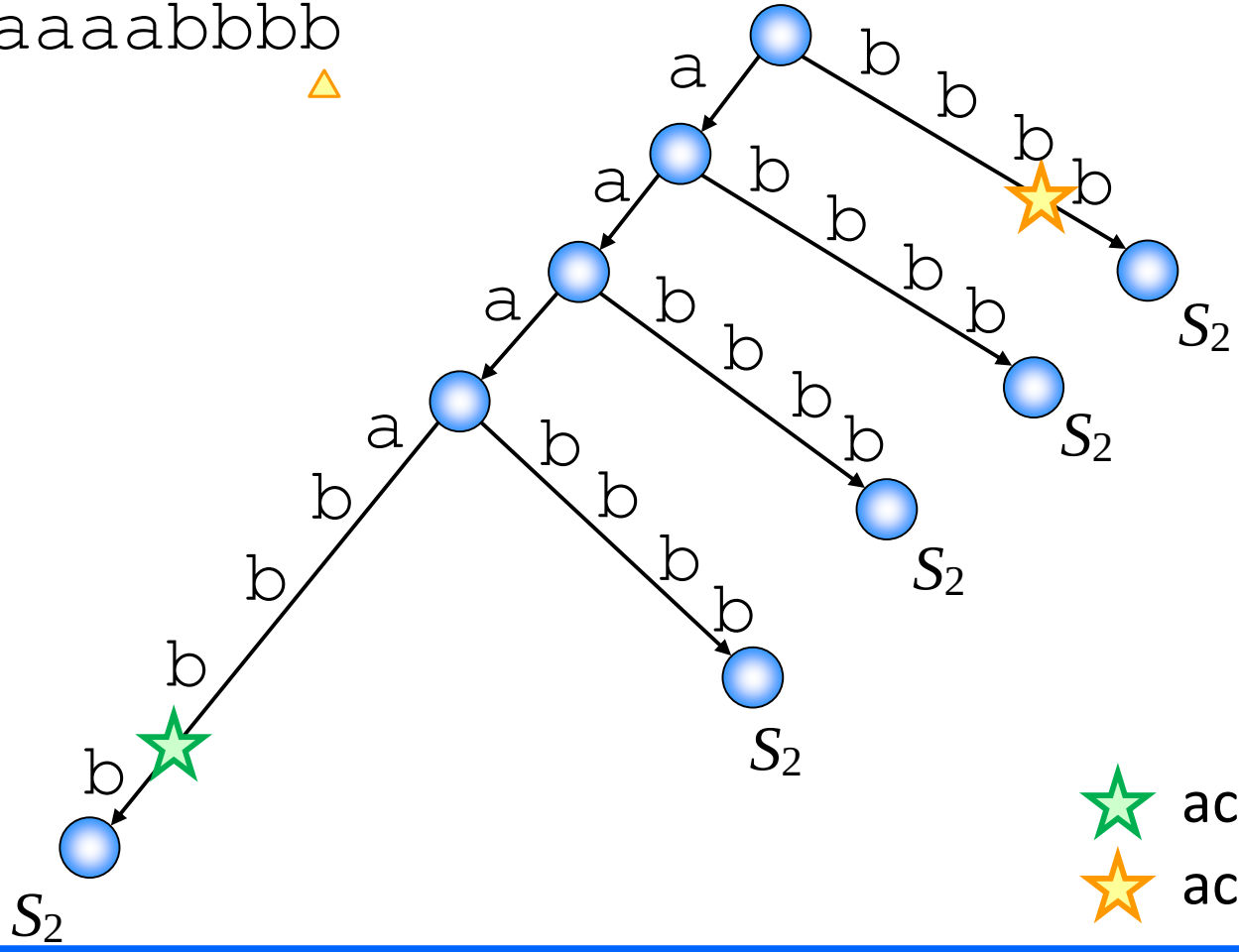


active point for S_1
active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership

S_1 aaaabbb

S_2 aaaabbbb



active point for S_1

active point for S_2

$\Omega(N^{1.5})$ time for explicit leaf ownership