

The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:

s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---

The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:

s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---

Idea:

Try to search the patterns in the text *in parallel*

The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---

Idea:

Try to search the patterns in the text *in parallel*

The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---



Idea:

Try to search the patterns in the text *in parallel*

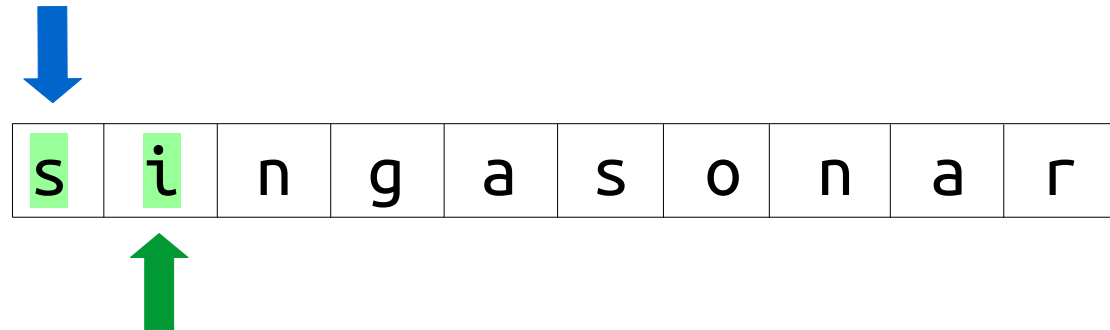
The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



Idea:

Try to search the patterns in the text *in parallel*

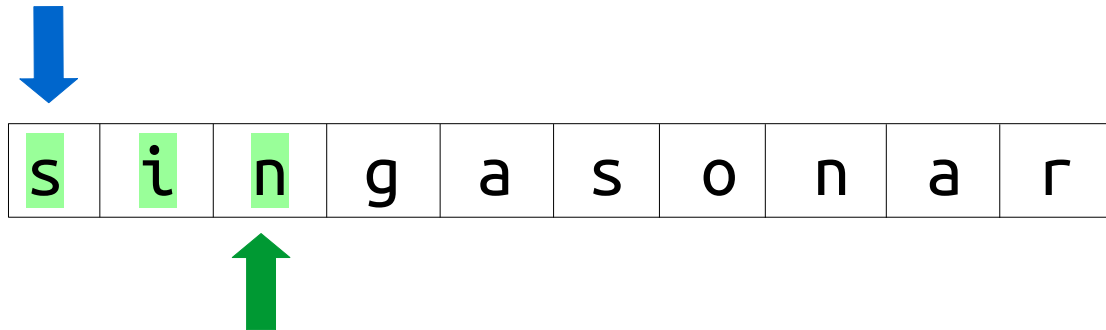
The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



“sin” found in position 0

Idea:

Try to search the patterns in the text *in parallel*

The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---



“sing” found in position 0

Idea:

Try to search the patterns in the text *in parallel*

The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---

Idea:

Try to search the patterns in the text *in parallel*

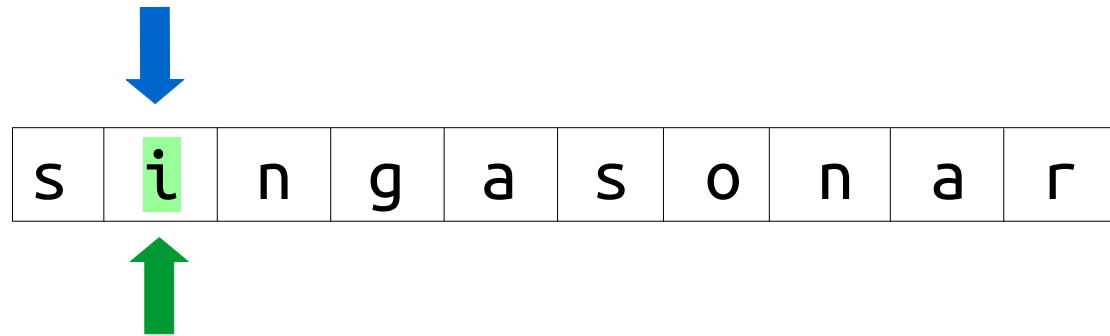
The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



Idea:

Try to search the patterns in the text *in parallel*

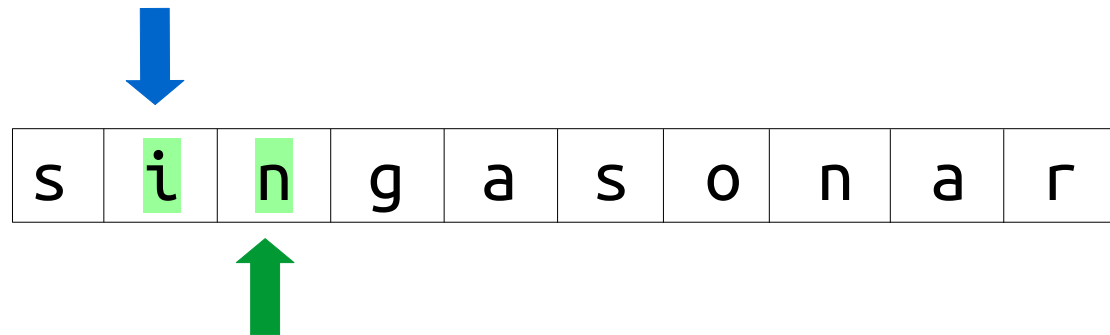
The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



“in” found in position 1

Idea:

Try to search the patterns in the text *in parallel*

The Problem

Given a text and multiple patterns,
find all the patterns in the text.

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

text:



s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---

Idea:

Try to search the patterns in the text *in parallel*

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

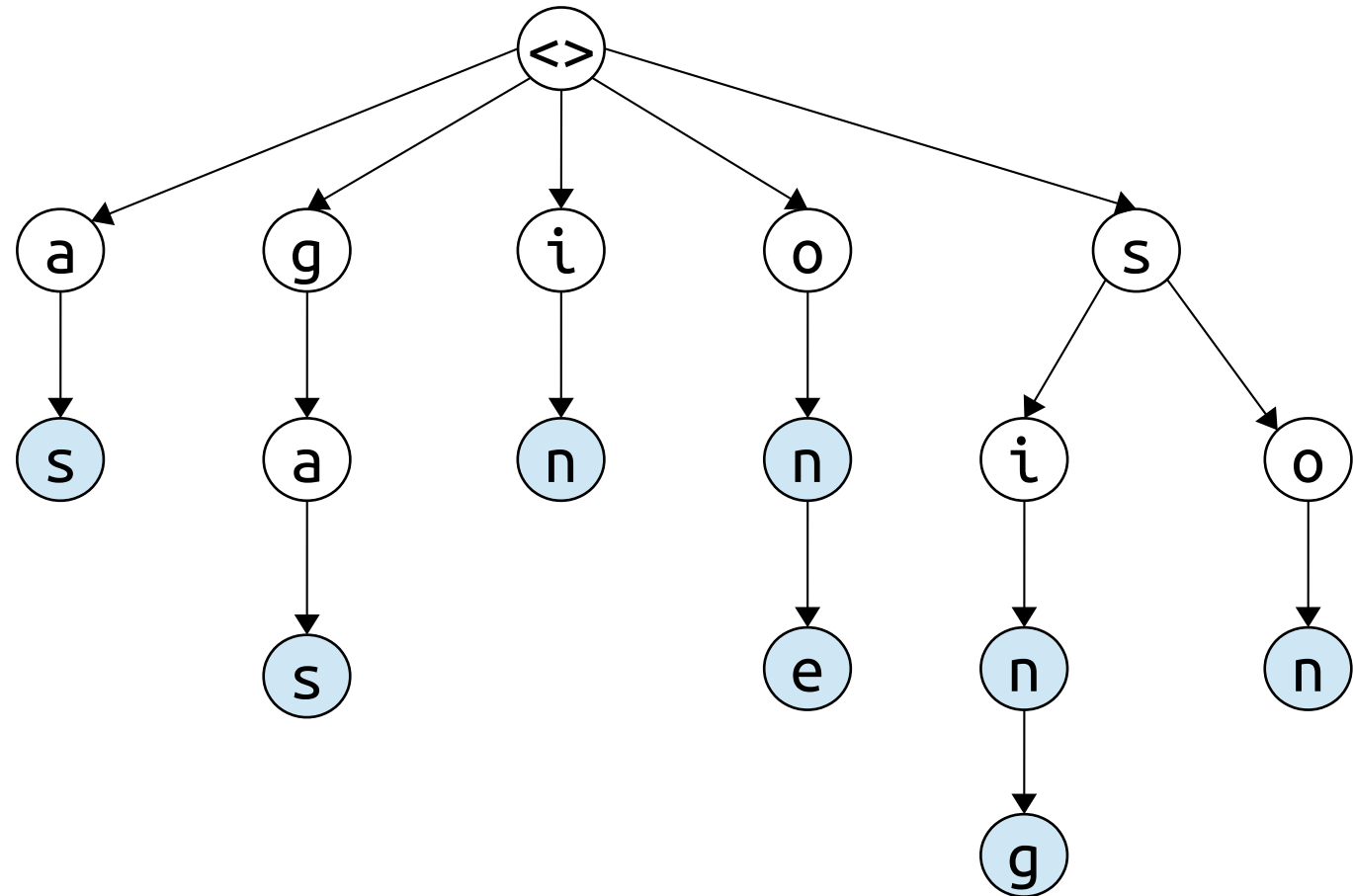
a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	

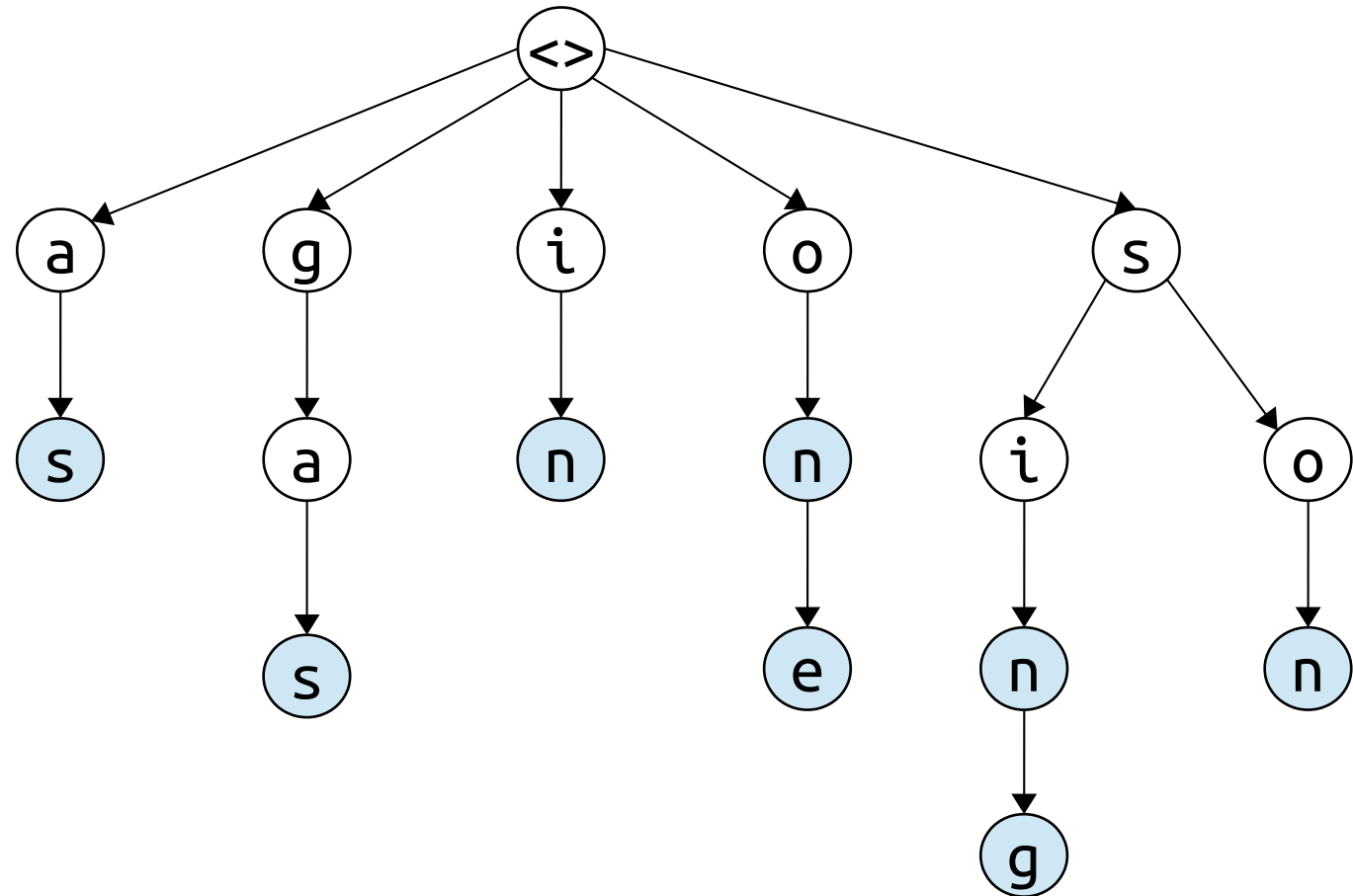


Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



text:

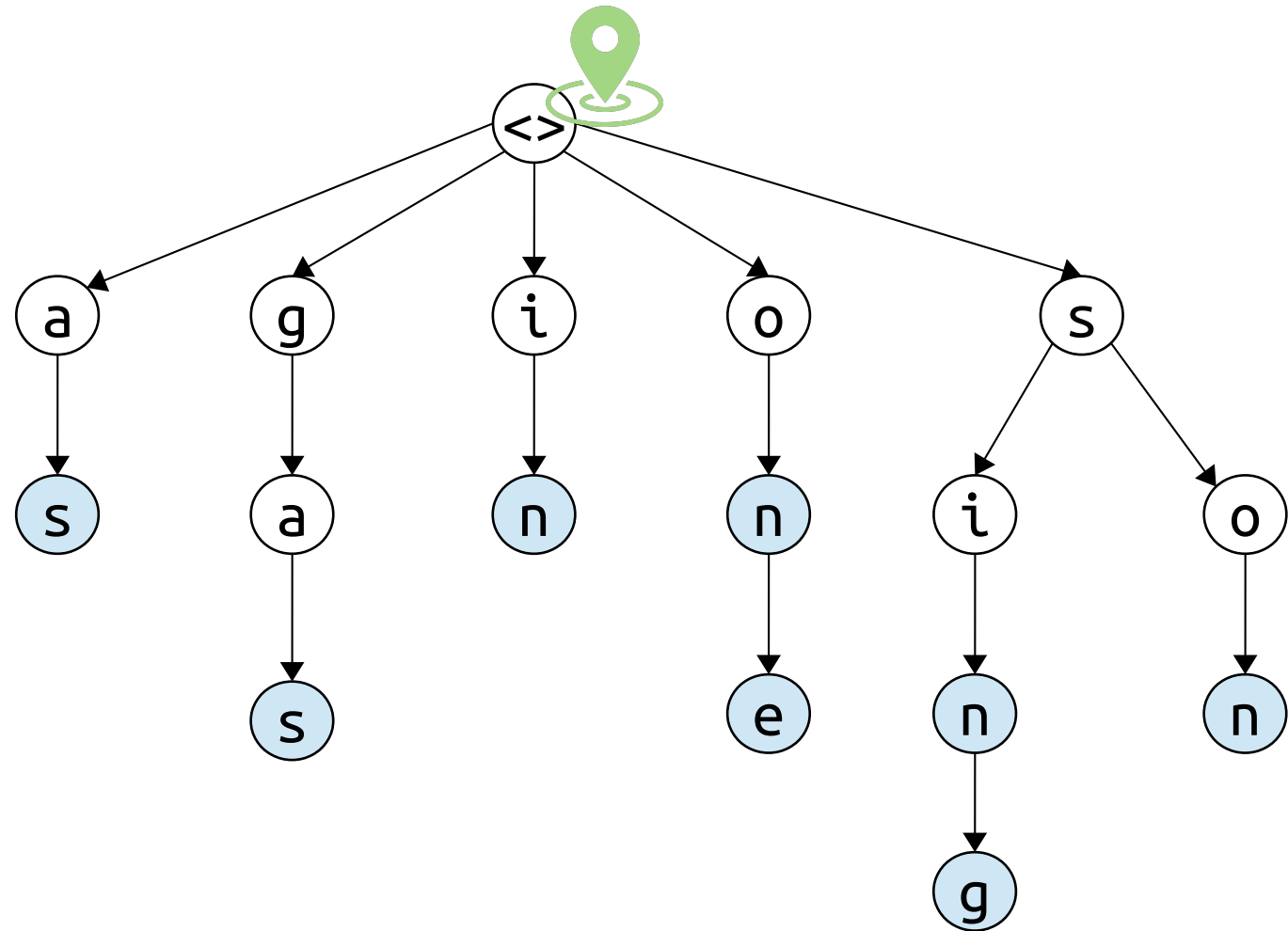
s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



↓

text:

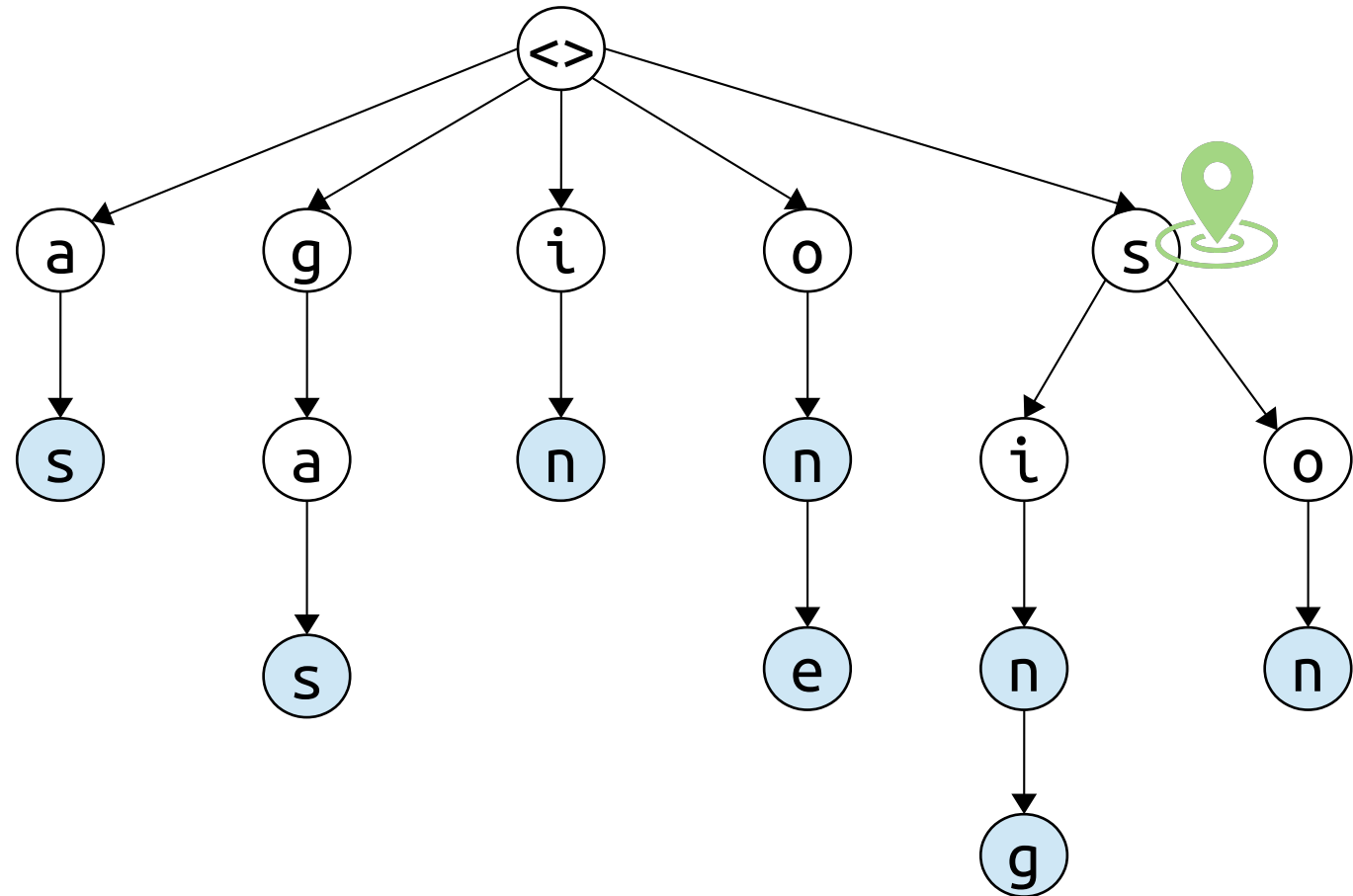
s	i	n	g	a	s	o	n	a	r
---	---	---	---	---	---	---	---	---	---

Aho-Corasick

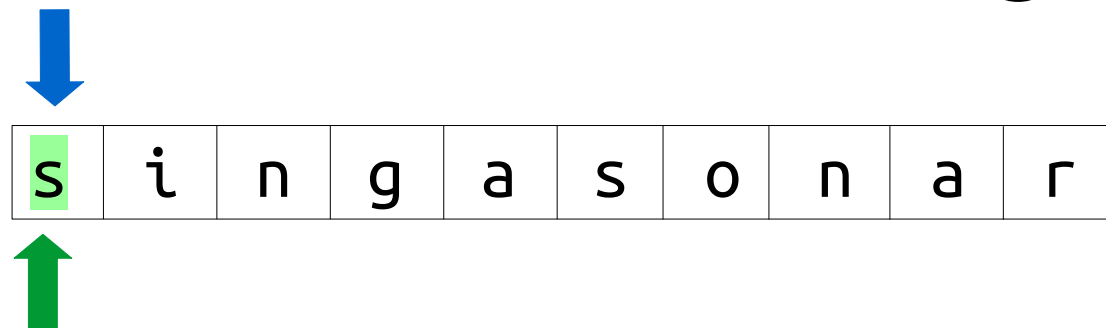
Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



text:

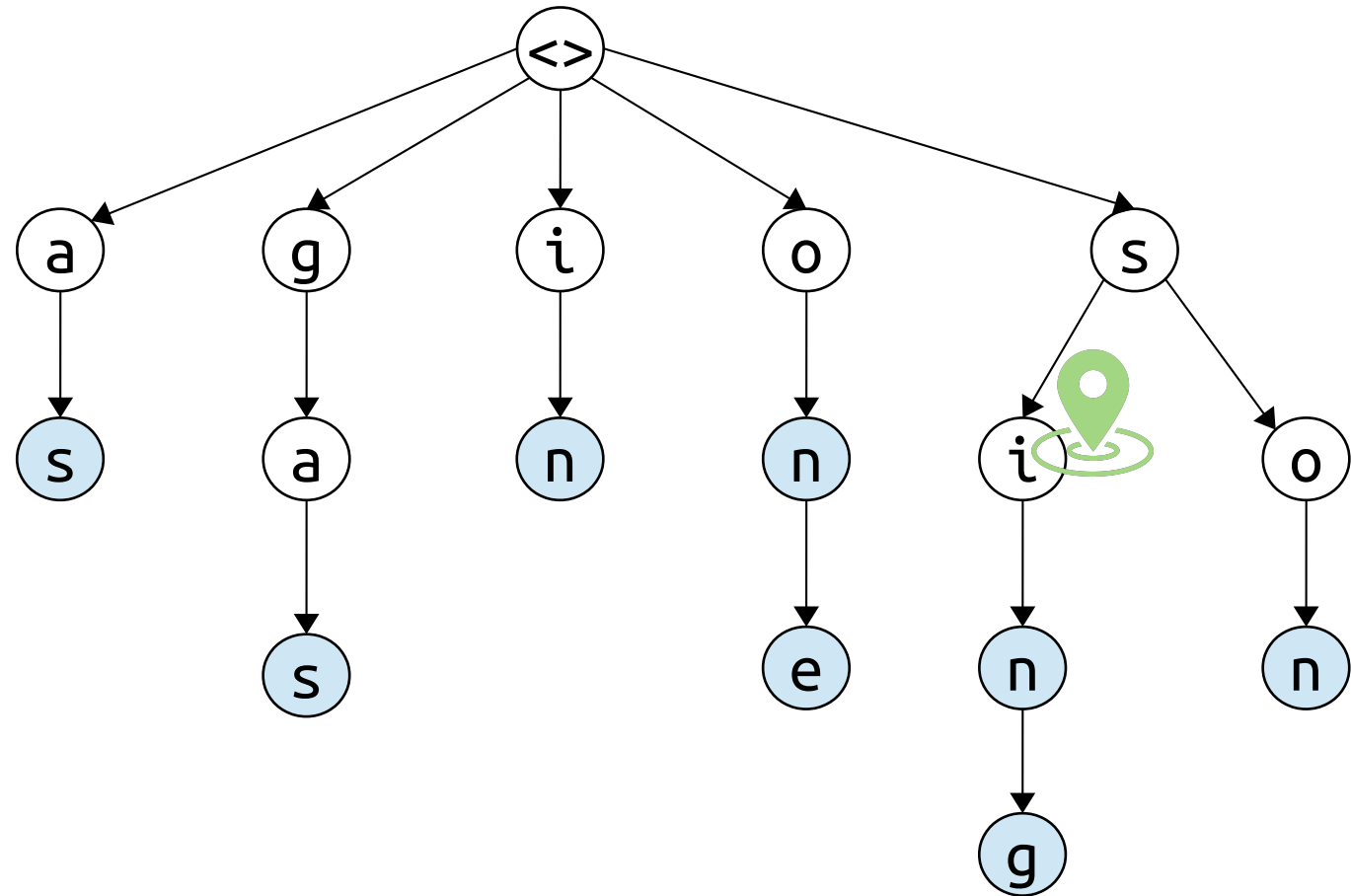


Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



text:

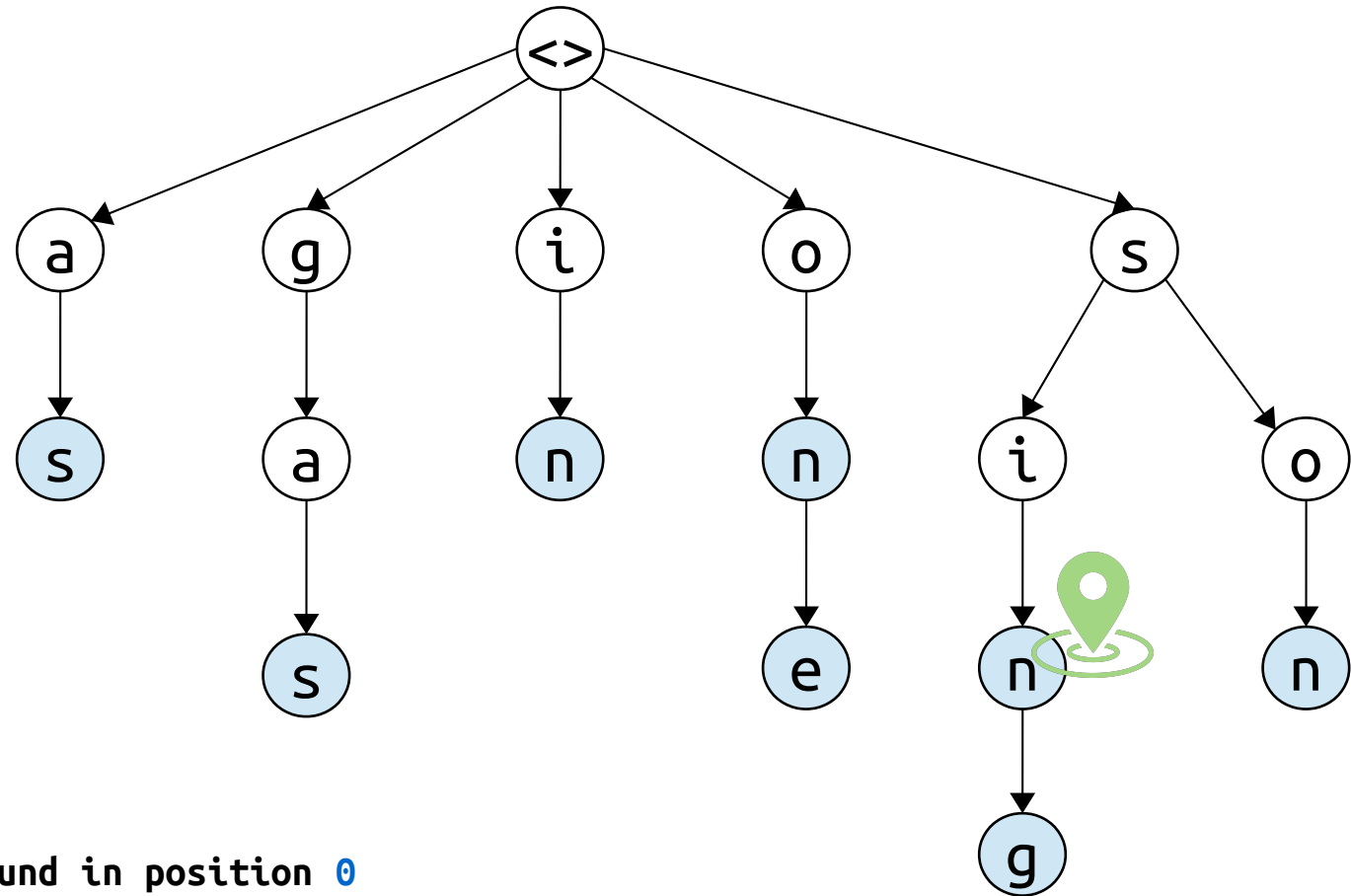
s i n g a s o n a r

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



“sin” found in position 0

text:

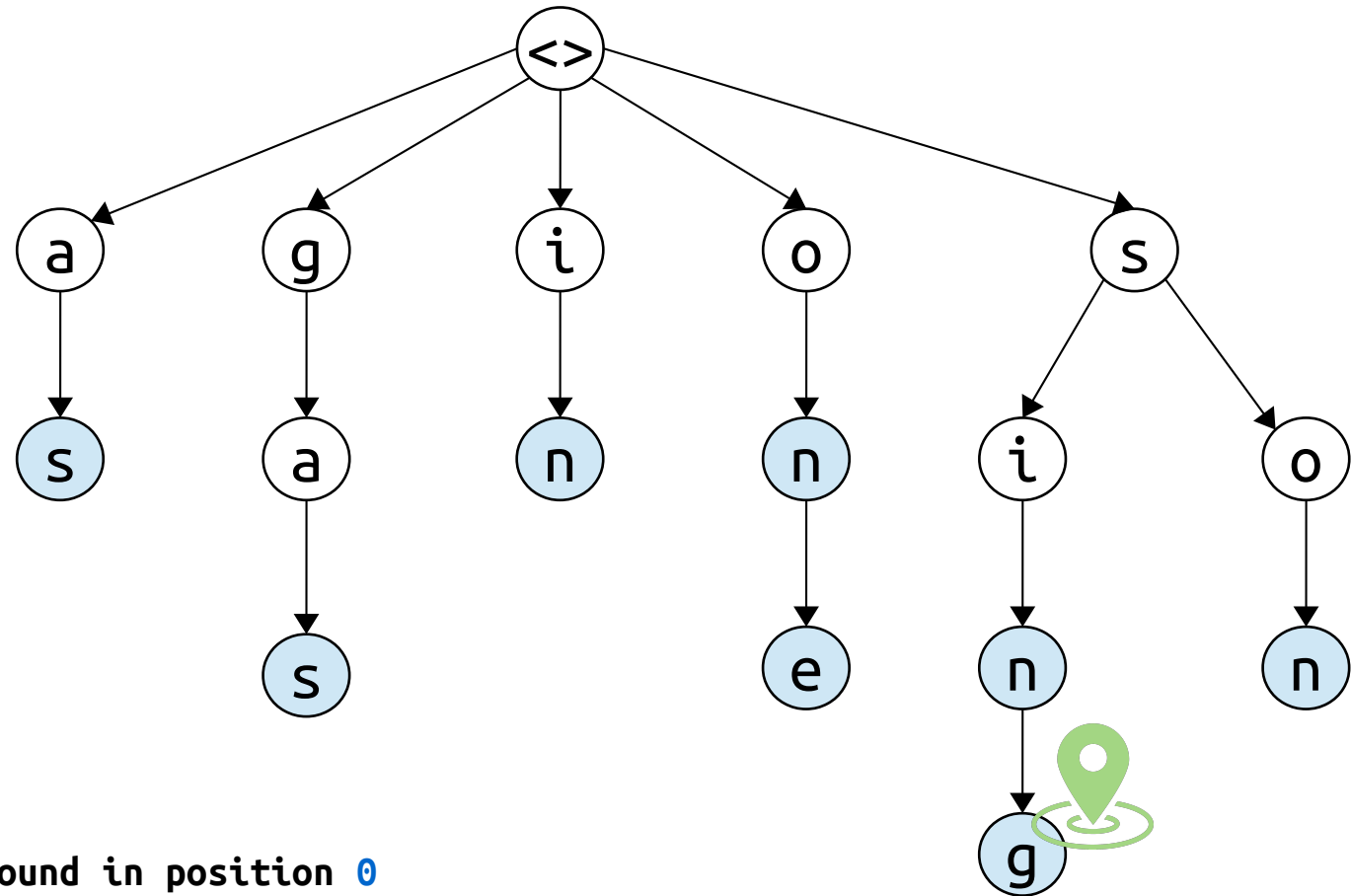
s i n g a s o n a r

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



“sing” found in position 0

text:

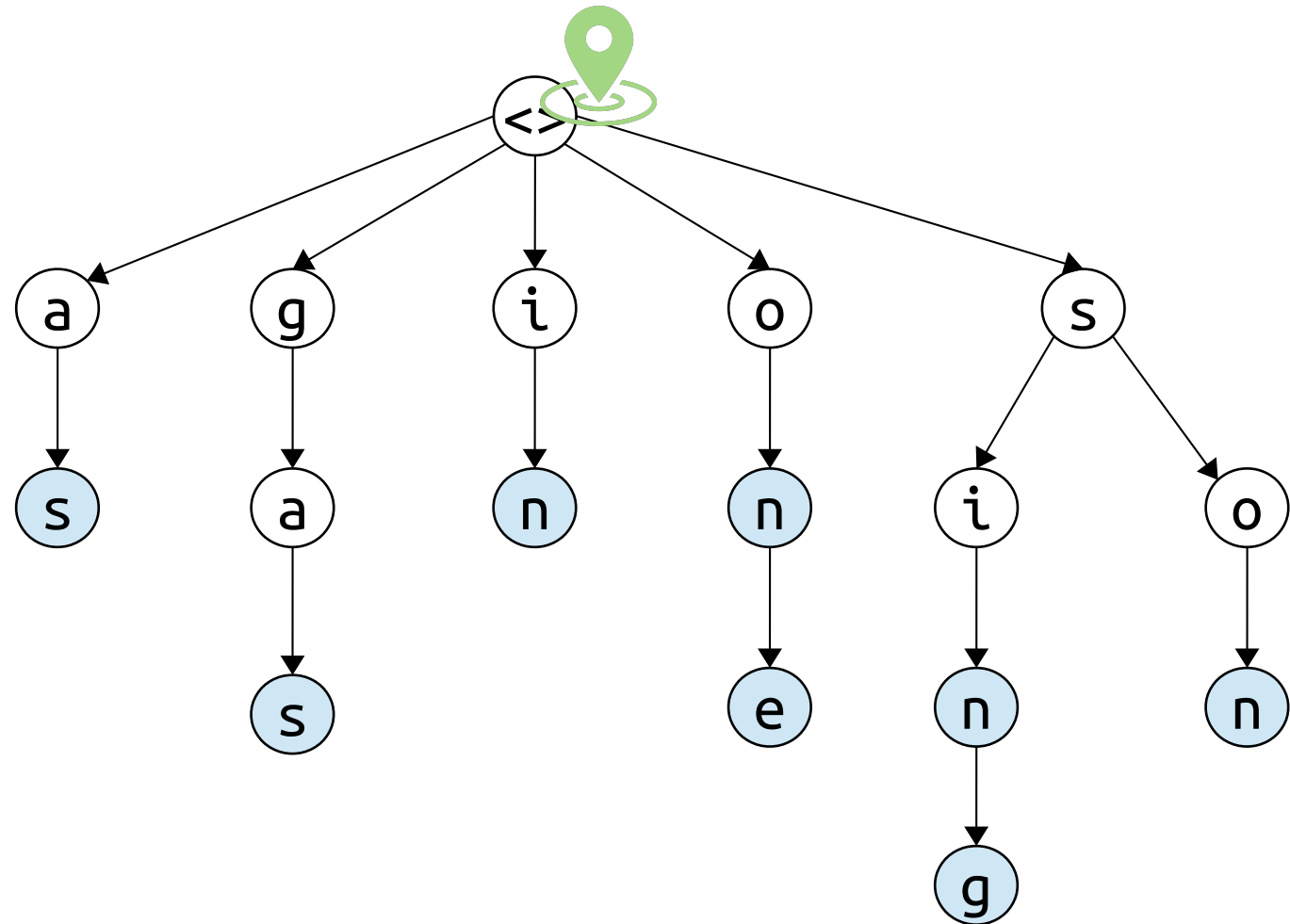
s i n g a s o n a r

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



text:

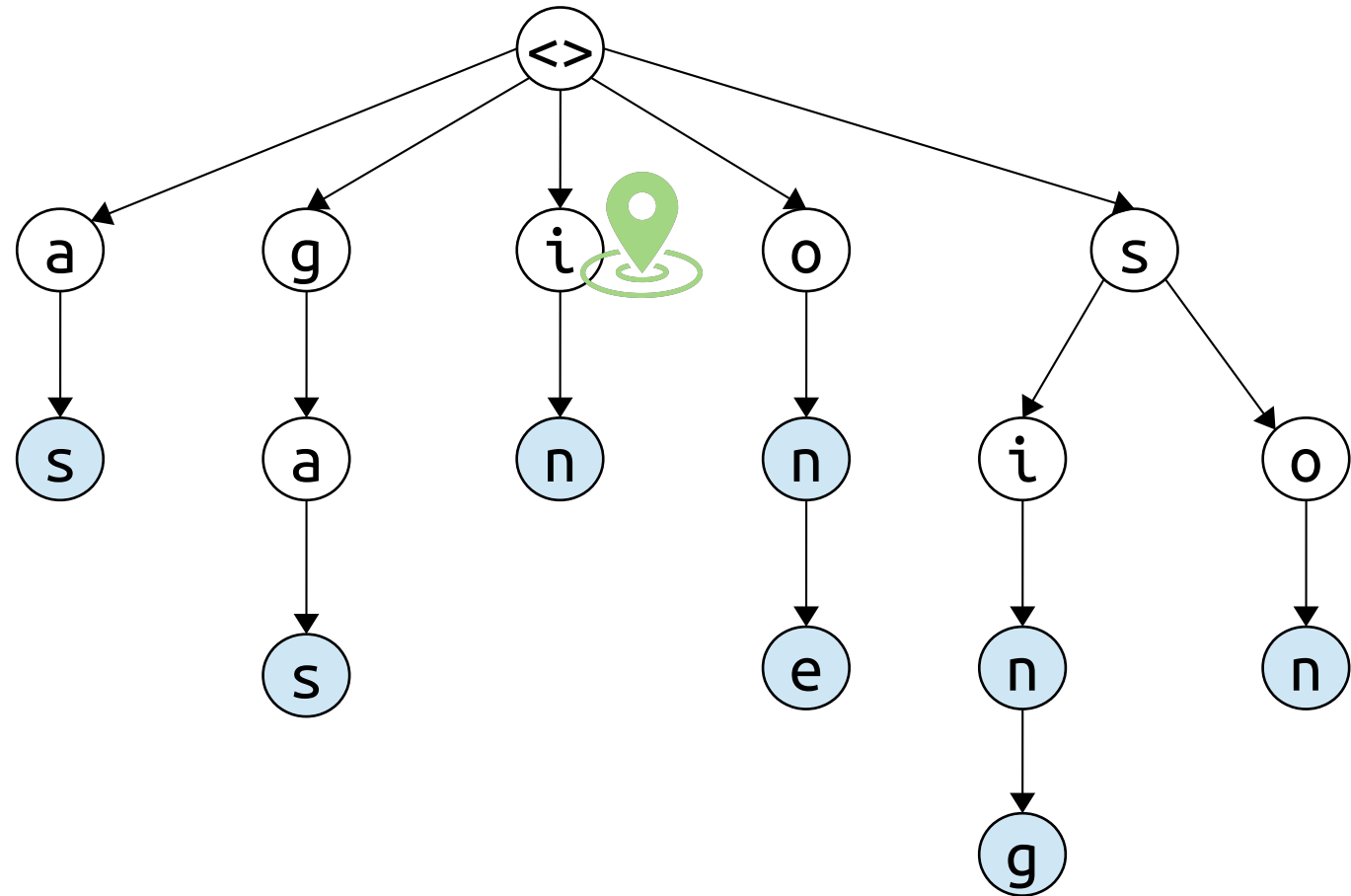
s i n g a s o n a r

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



text:

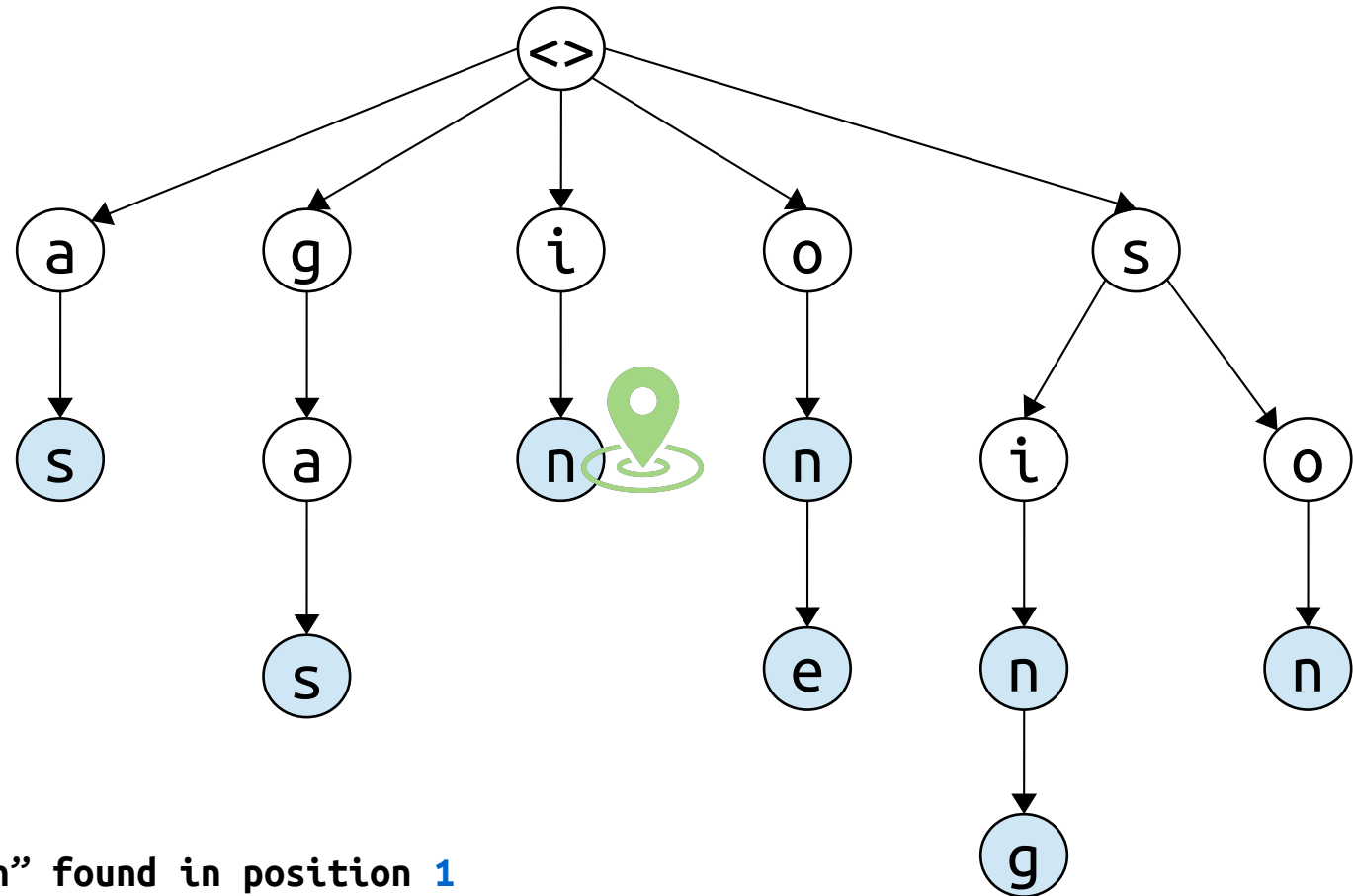
s i n g a s o n a r

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



“in” found in position 1

text:

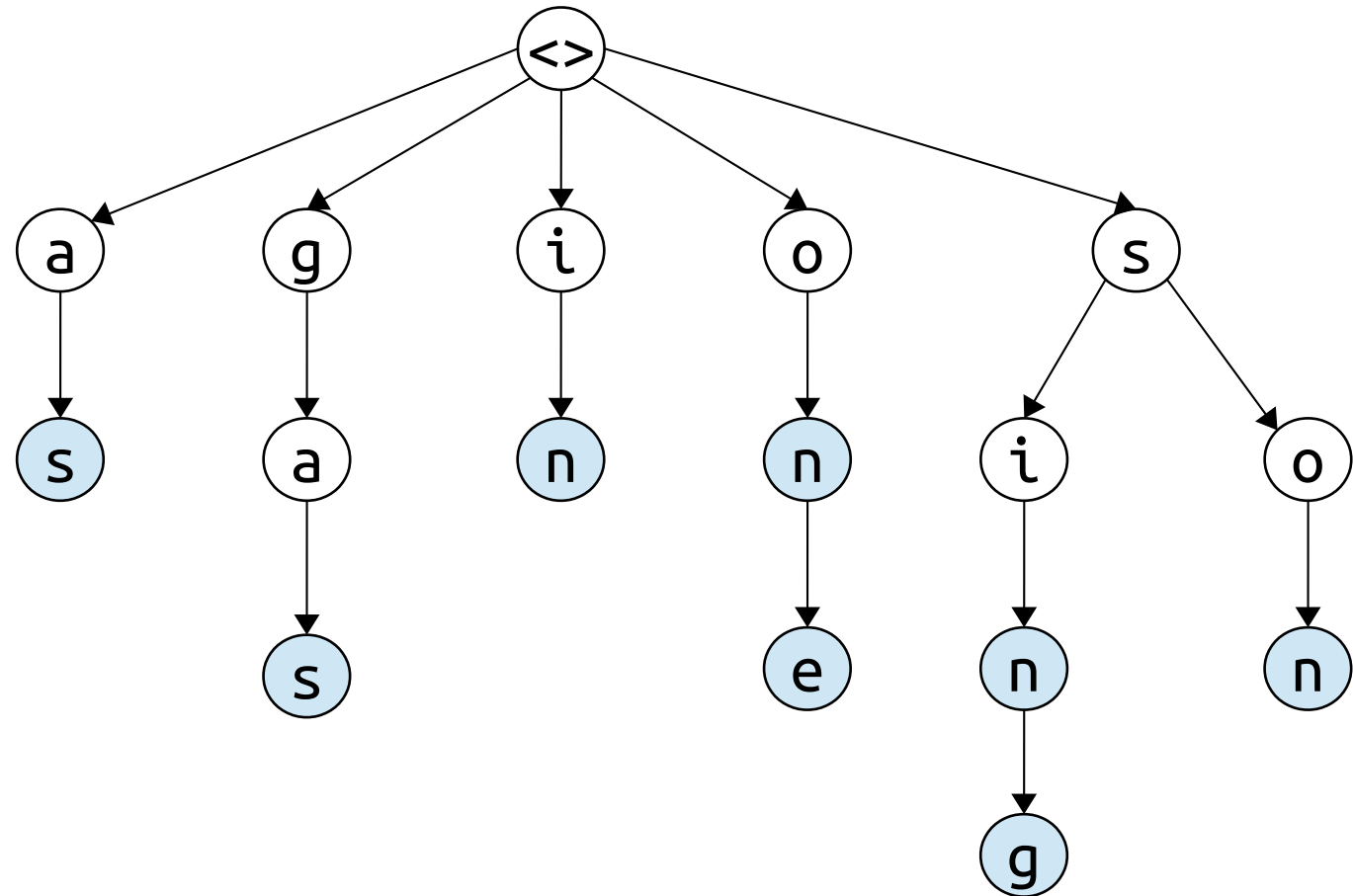
s i n g a s o n a r

Aho-Corasick

Given a text and multiple patterns stored in a trie
find all the patterns in the text

patterns:

a	s		
g	a	s	
i	n		
o	n		
o	n	e	
s	i	n	g
s	i	n	
s	o	n	



AHO-CORASICK BASIC: TIME COMPLEXITY

$O(M * L_{\max})$ where

M : size of the text

L_{\max} : size of the longest pattern

Aho-Corasick – Suffix Links

In Aho-Corasick Tries (also called Aho-Corasick Automata)
Suffix links are used to *speed up* the search.

*Using **suffix links**,*

*We can make Aho-Corasick faster
and take down*

$O(M * L_{\max})$

To

$O(M)$

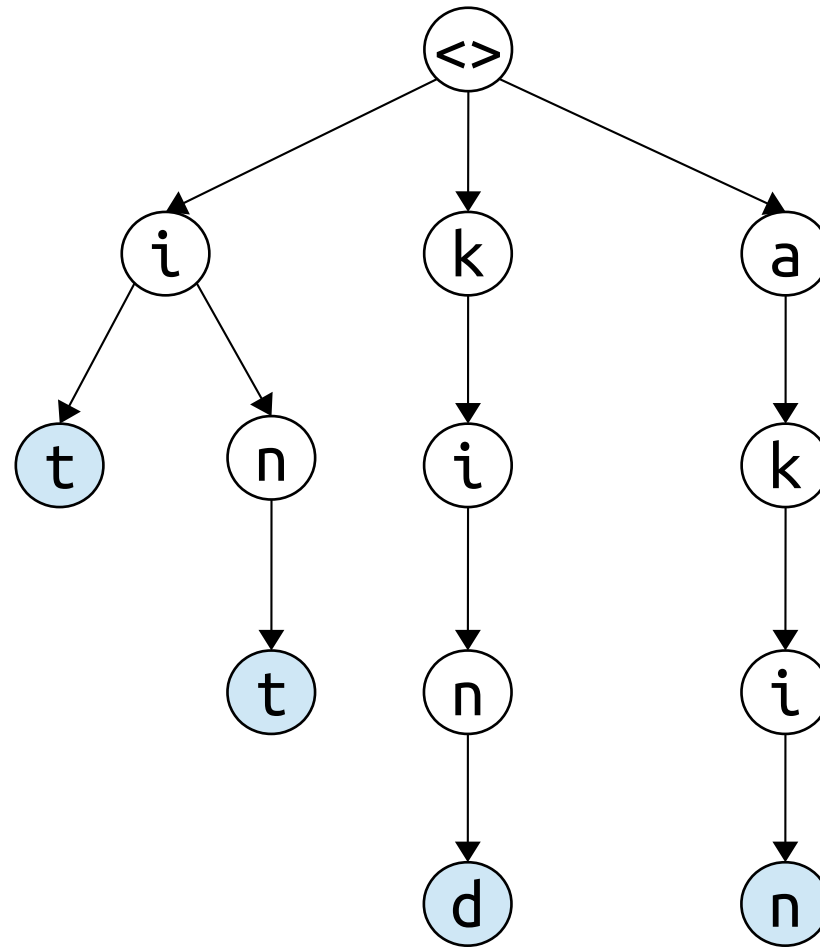
M : size of the text

L_{\max} : size of the longest pattern

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n

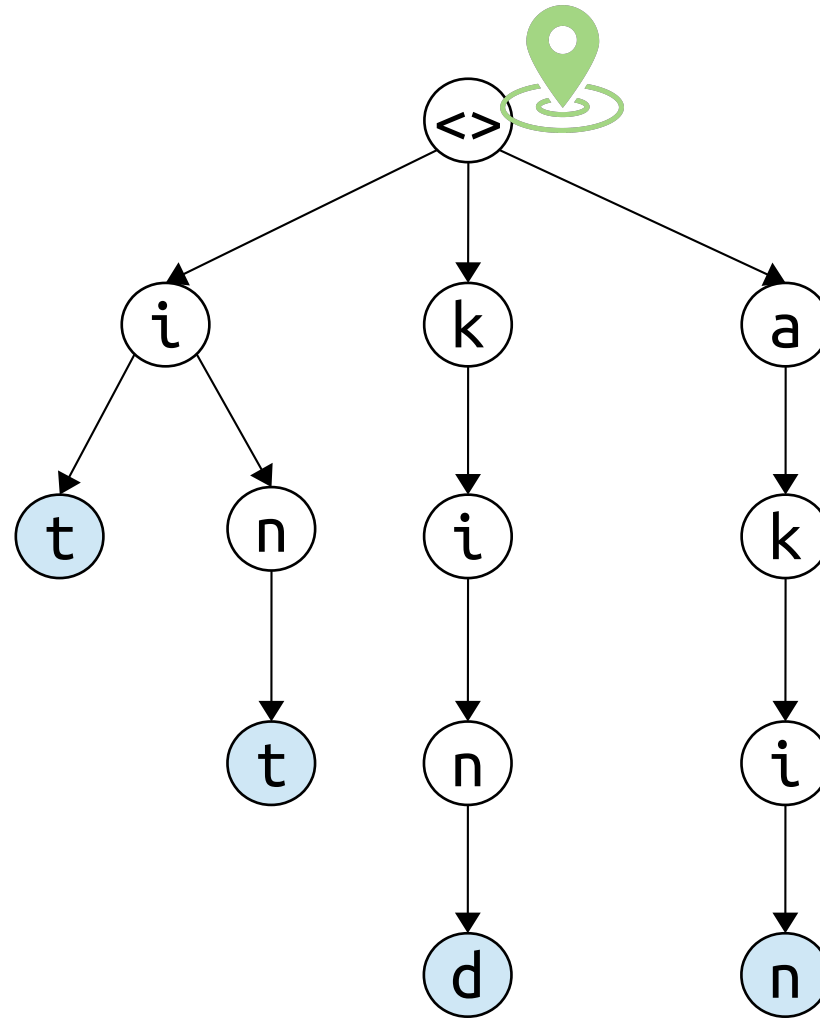



text: a k i n d

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n


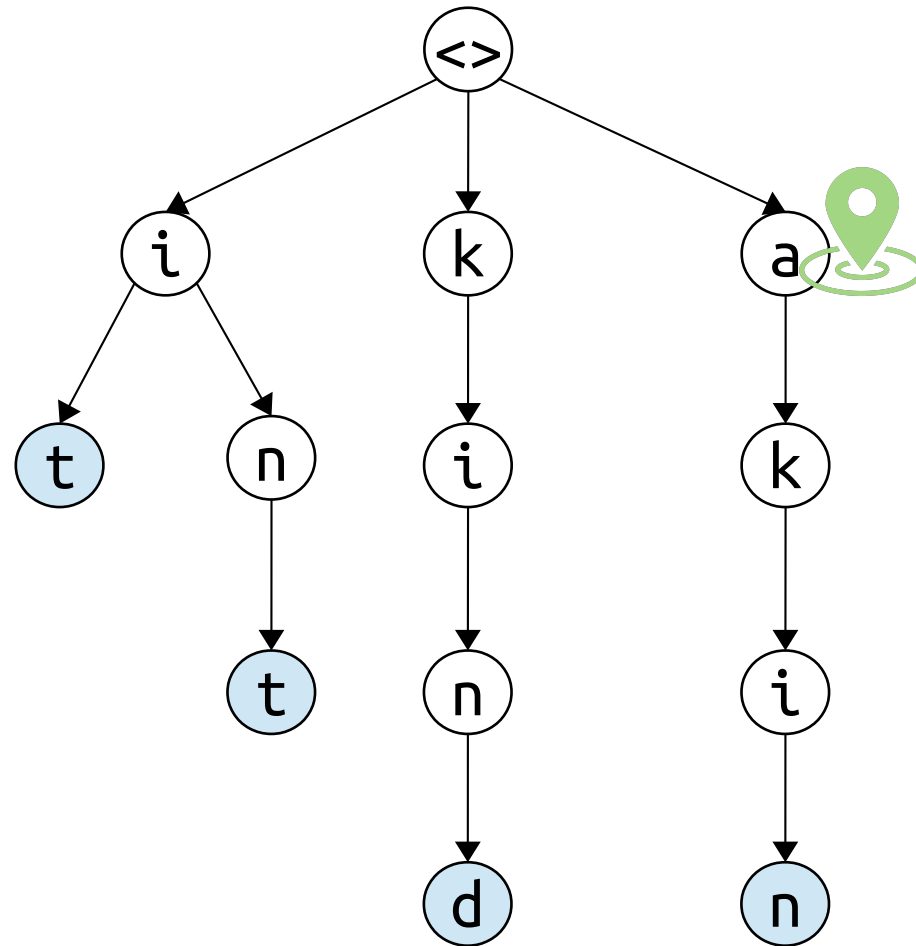


text:  a k i n d

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n

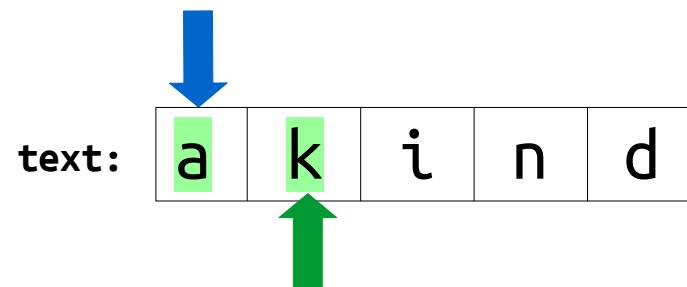
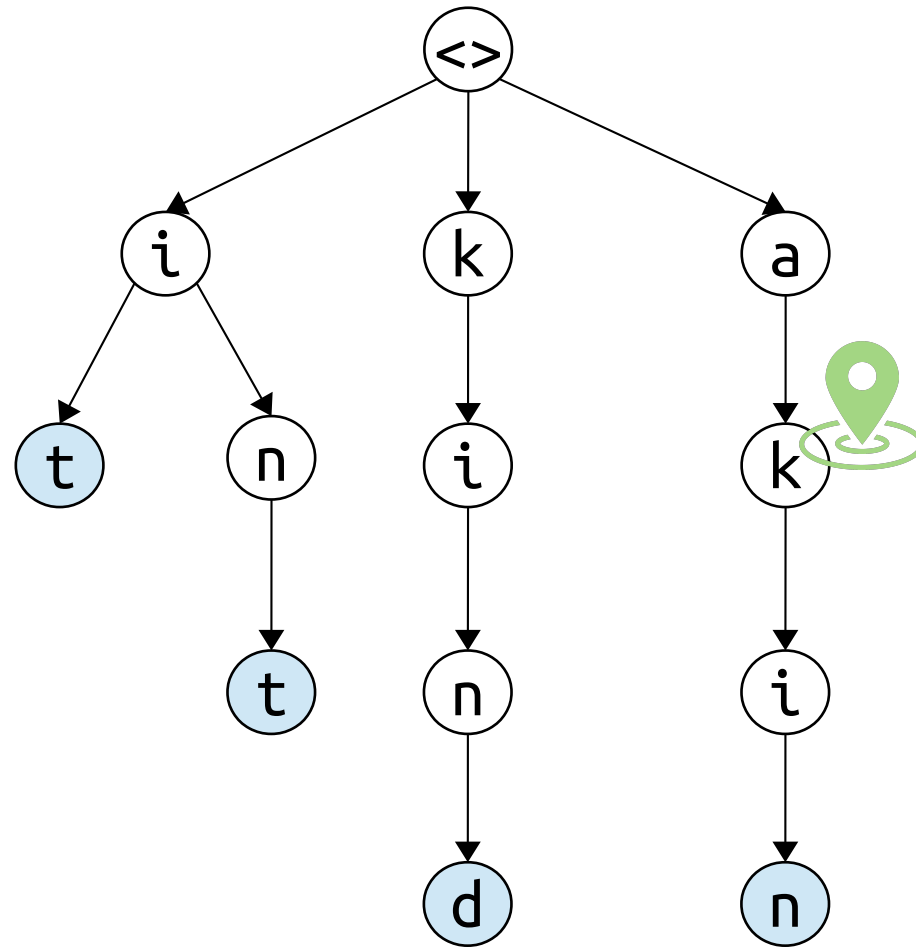


text: a k i n d

Aho-Corasick – Suffix Links

patterns:

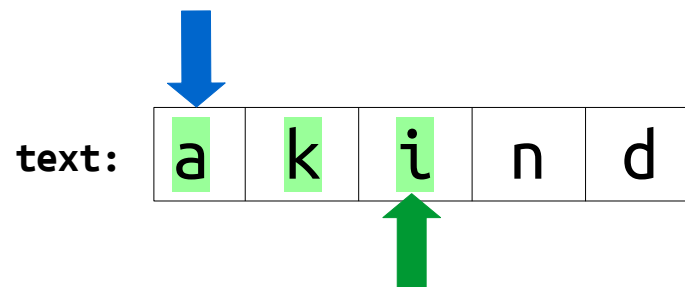
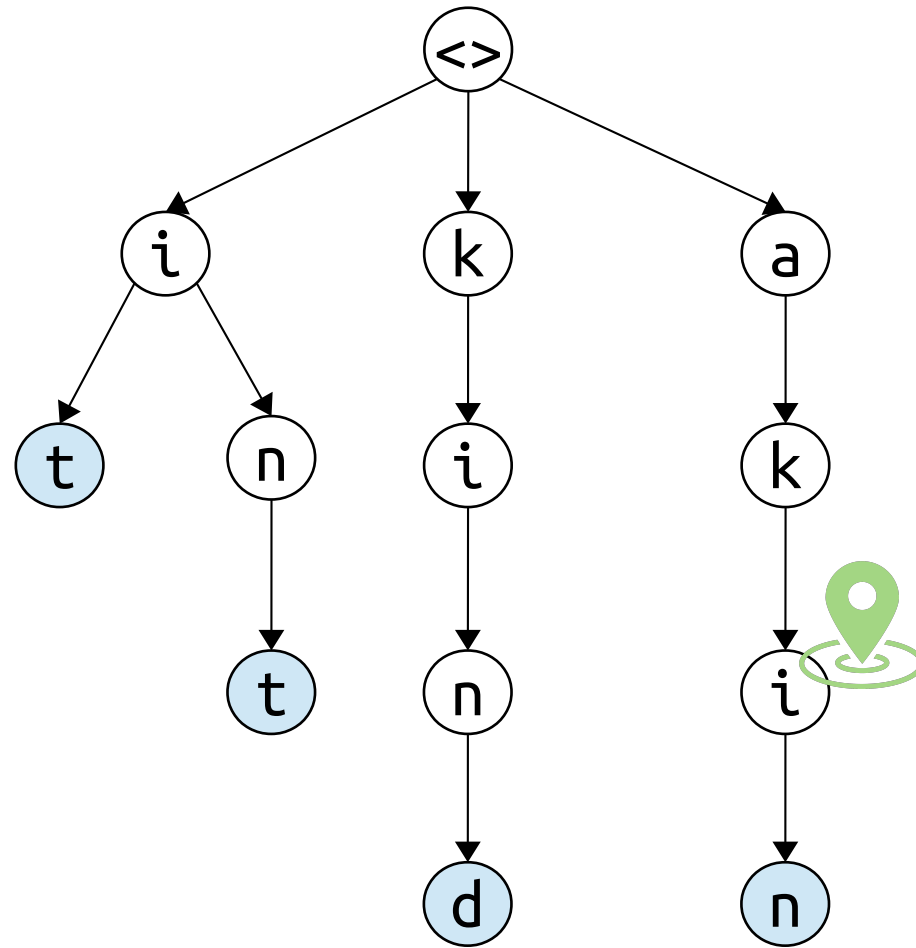
i	t		
i	n	t	
k	i	n	d
a	k	i	n



Aho-Corasick – Suffix Links

patterns:

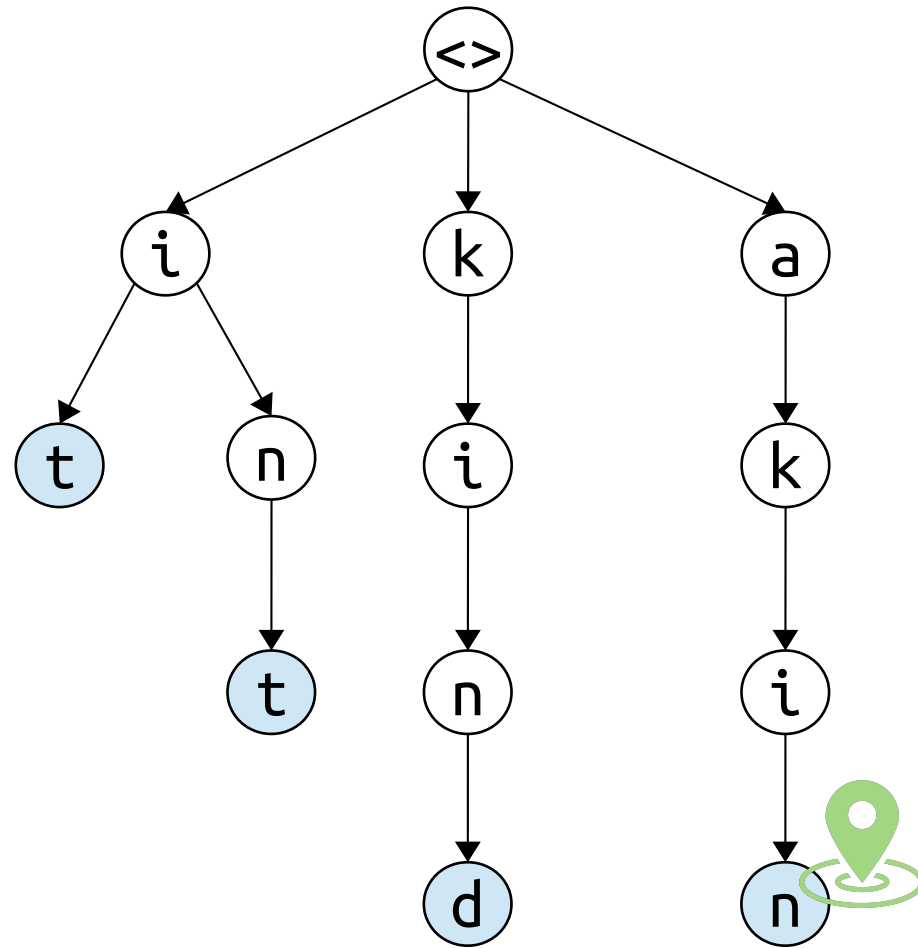
i	t		
i	n	t	
k	i	n	d
a	k	i	n



Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



“akin” found in position 0

text:

a	k	i	n	d
---	---	---	---	---

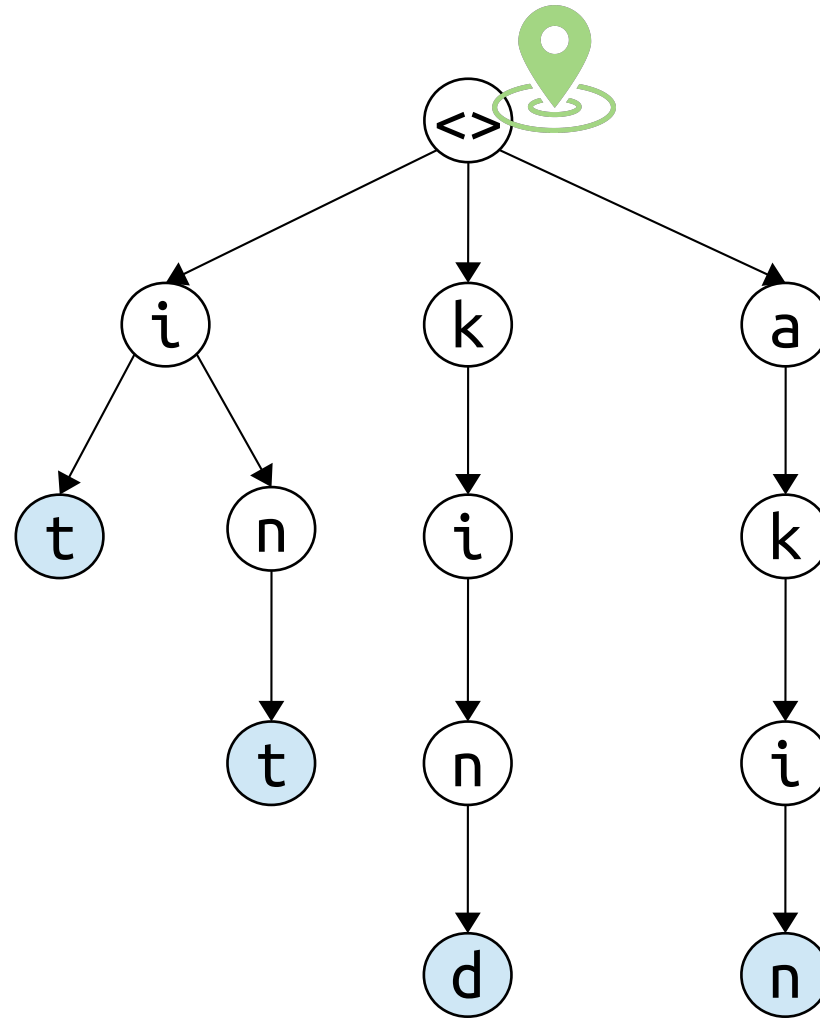
A blue arrow points from the text '“akin” found in position 0' to the first character 'a' in the text array. A green arrow points from below to the fourth character 'n' in the text array.

This node has no children, so we should return, and re-start the search from position 1 of the text

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



This node has no children, so we should return, and re-start the search from position **1** of the text

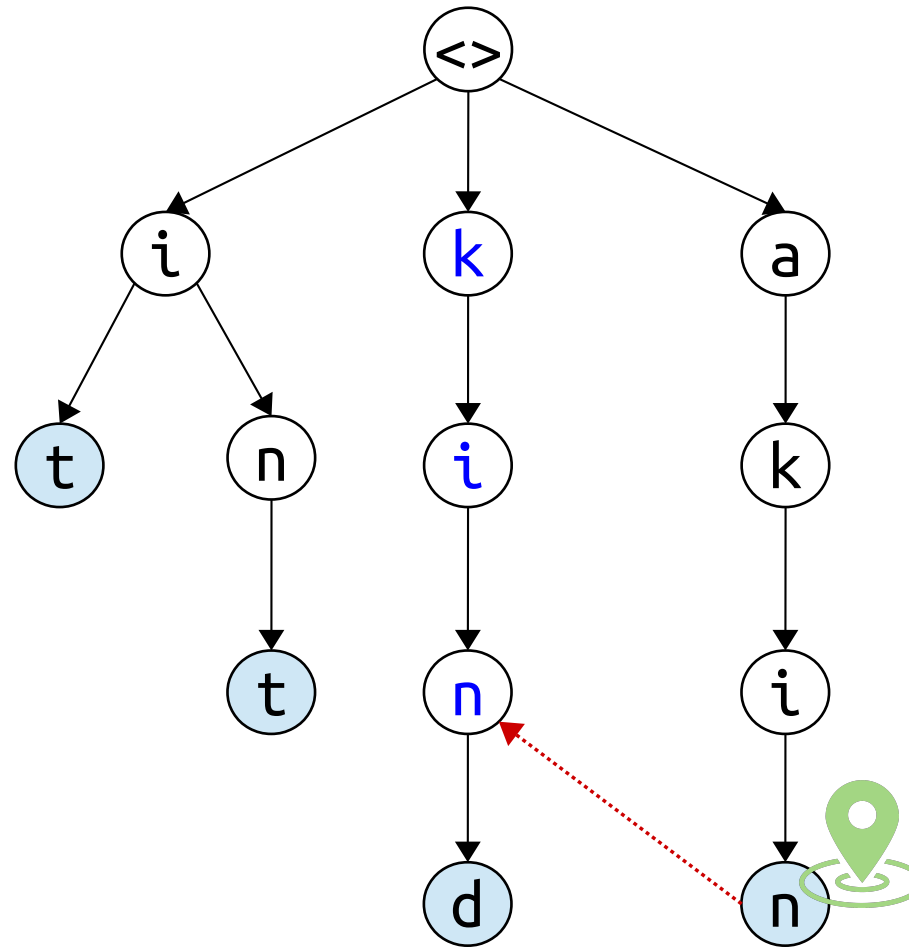
text:

a	k	i	n	d
---	---	---	---	---

Aho-Corasick – Suffix Links

patterns:

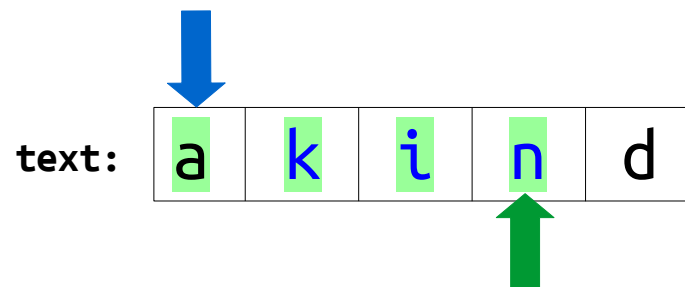
i	t		
i	n	t	
k	i	n	d
a	k	i	n



BUT,
A suffix of the sequence
we visited appears in
another branch of the
trie.

We use **suffix links** to
skip steps.

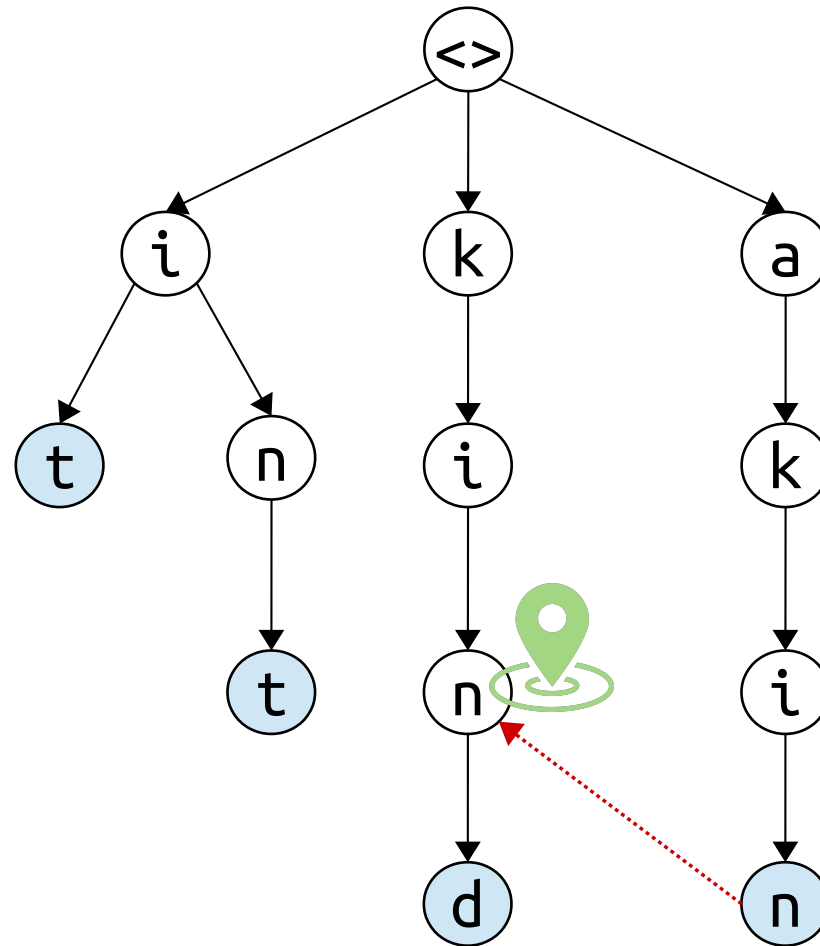
(we will see later how to
construct them)



Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



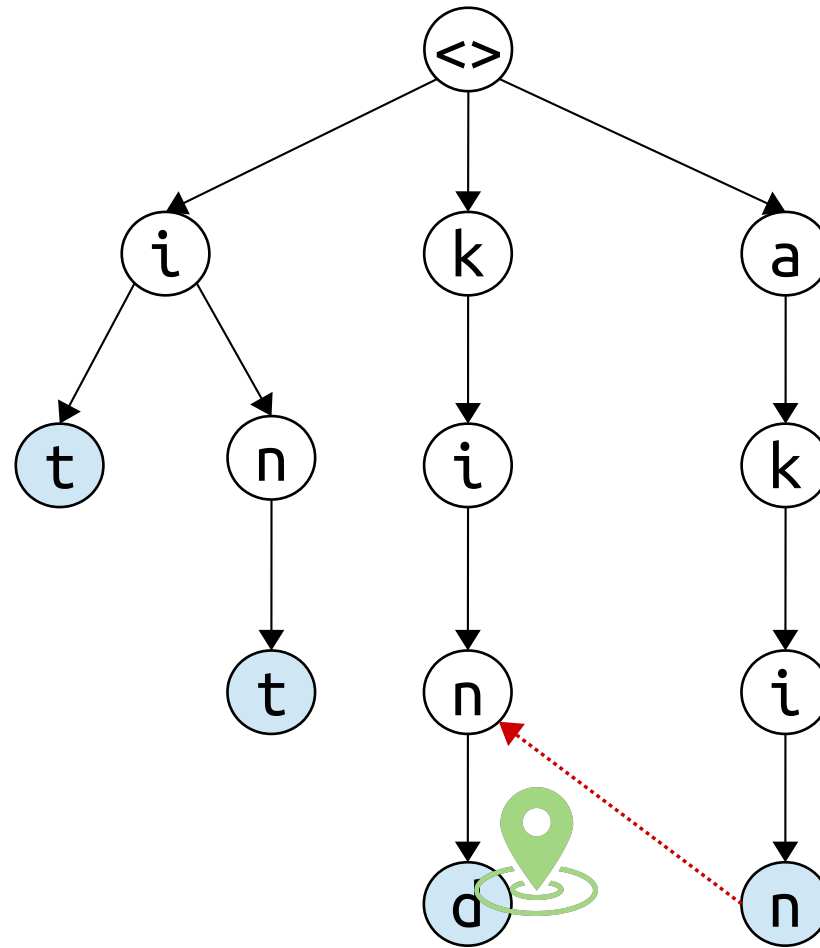
text: a k i n d

A blue arrow points down from the trie diagram to the text 'a k i n d'. A green arrow points up from the 'n' in the text to the 'n' node in the trie that is the child of 'i' (under 'k').

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n

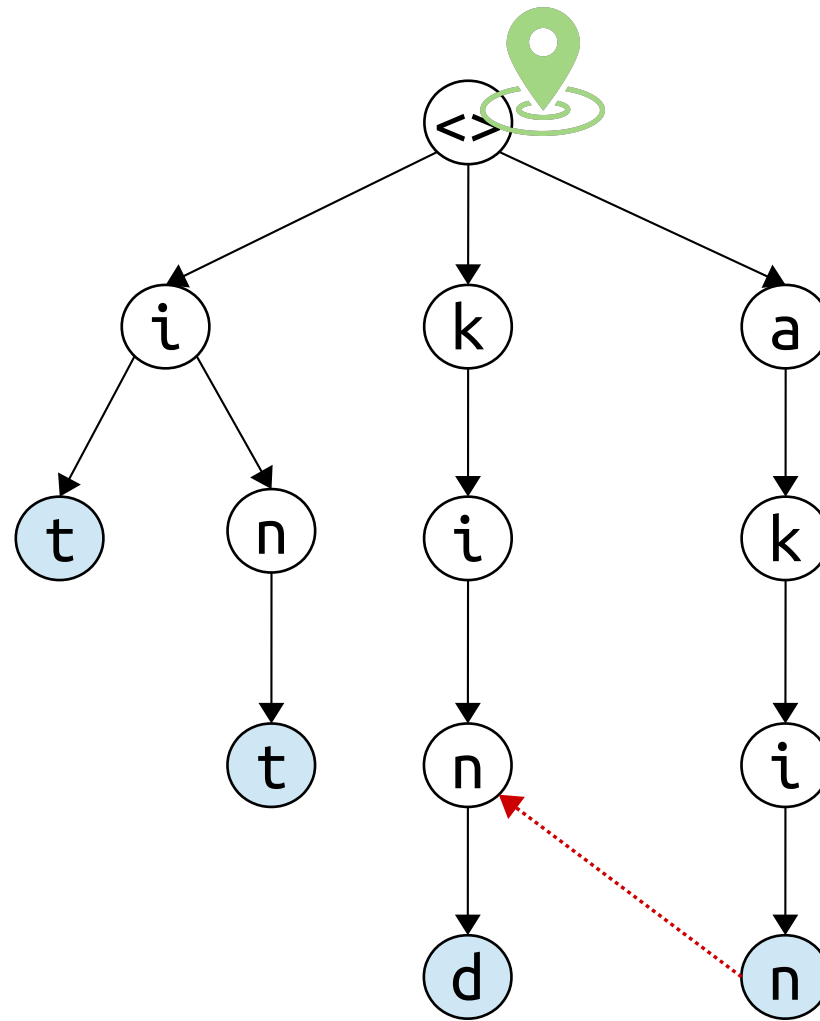


text: a k i n d

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



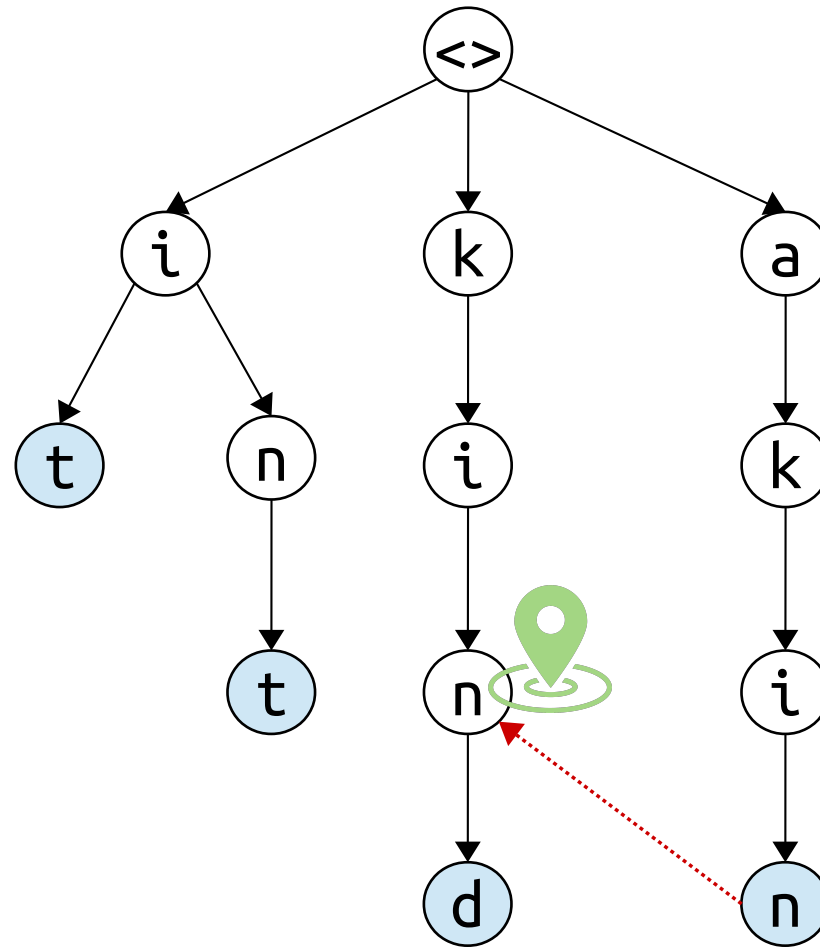
↓

text: k i n t

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



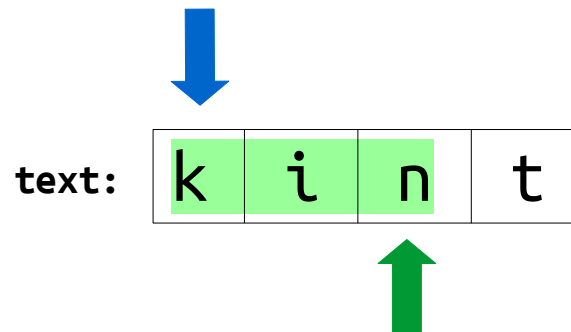
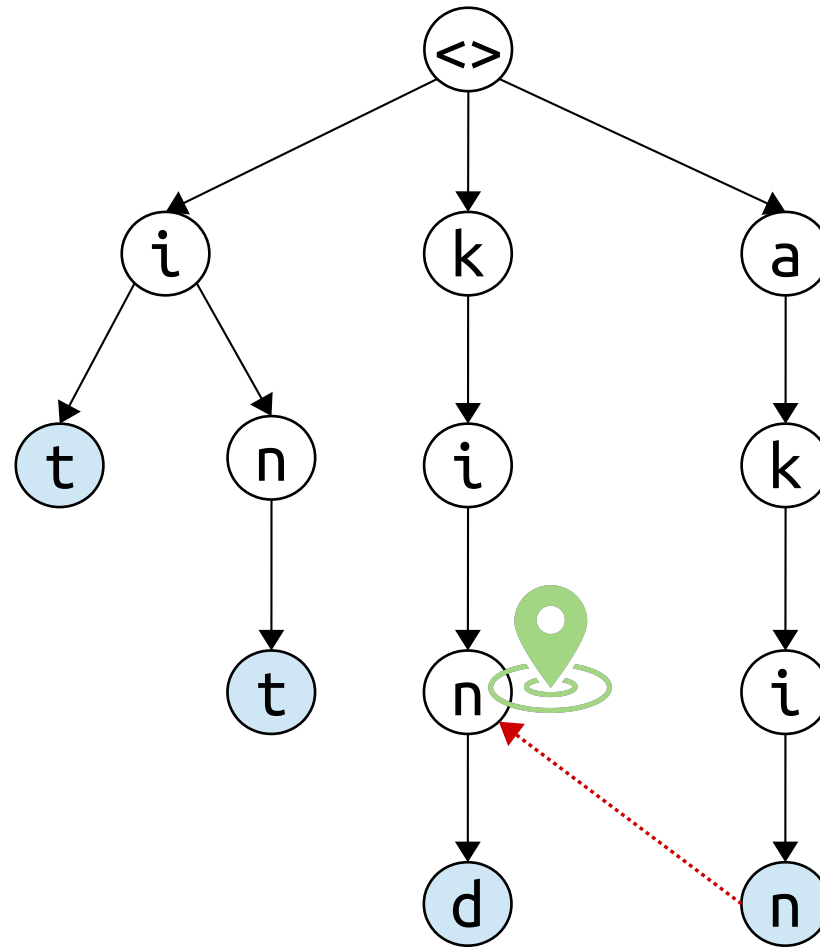
text:

k i n t

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n

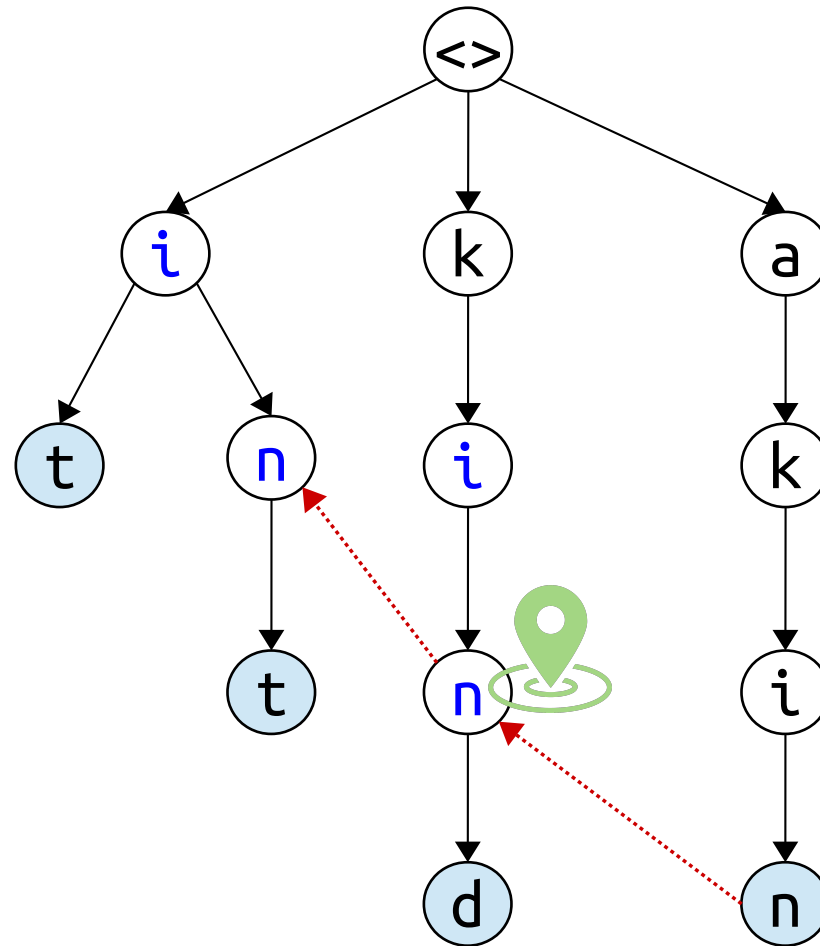


No pattern has found in the current branch.
Node “n” has not child “t”

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



text:

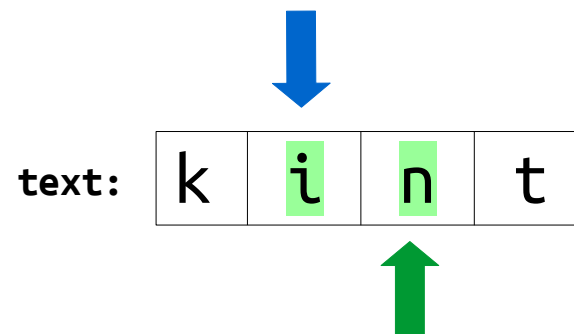
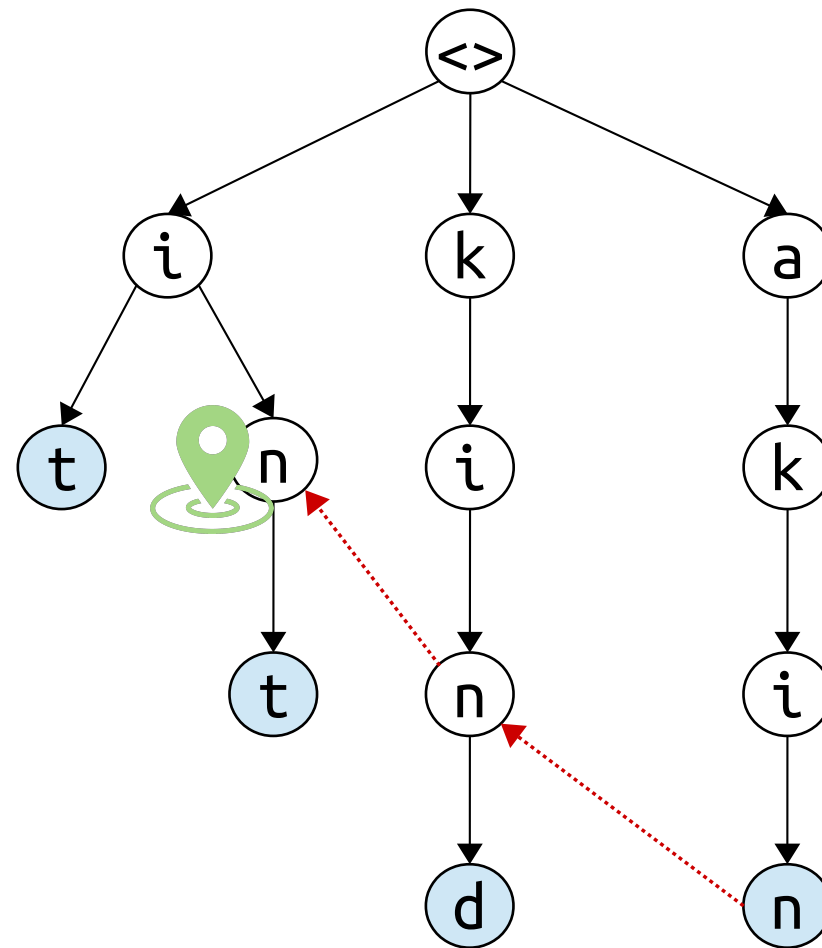
k i n t

BUT, suffix “i n” of the sequence we have visited appears in another branch of the trie.

Aho-Corasick – Suffix Links

patterns:

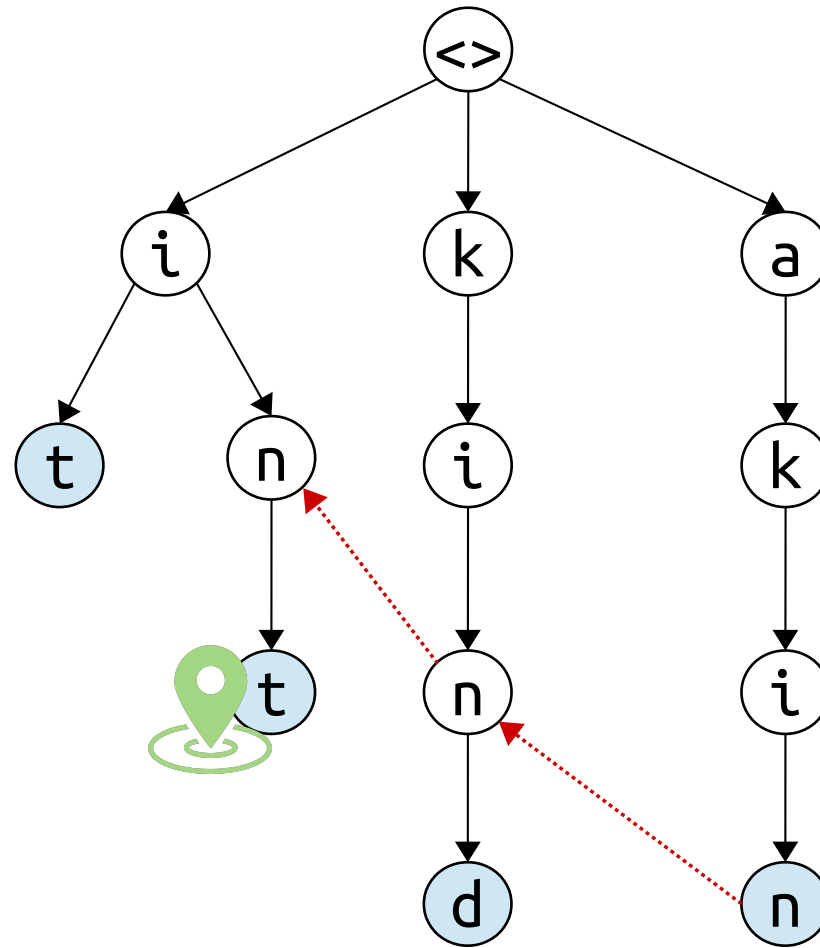
i	t		
i	n	t	
k	i	n	d
a	k	i	n



Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



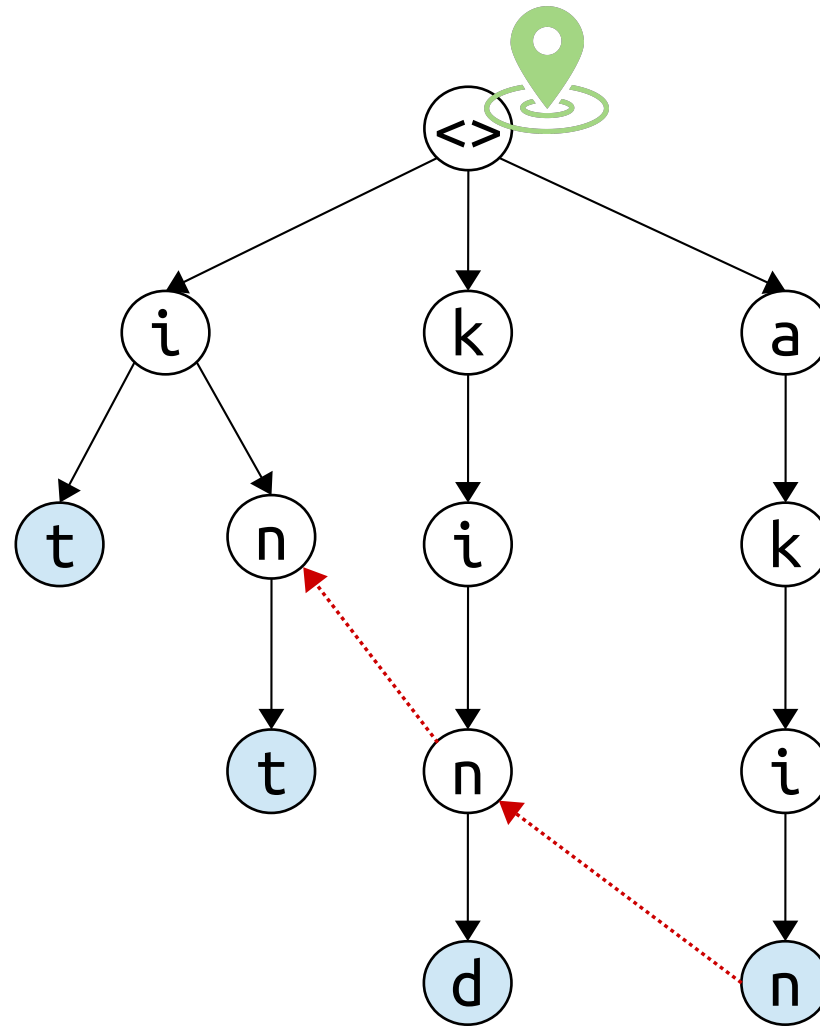
text: k i n t


The text 'k i n t' is shown in a sequence of four boxes. The characters 'i', 'n', and 't' are highlighted in green. A blue arrow points down from the trie diagram to the text, and a green arrow points up from the text to the 't' character.

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n

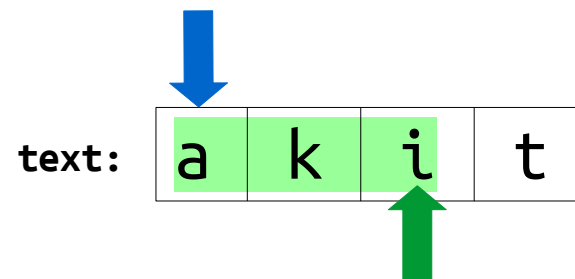
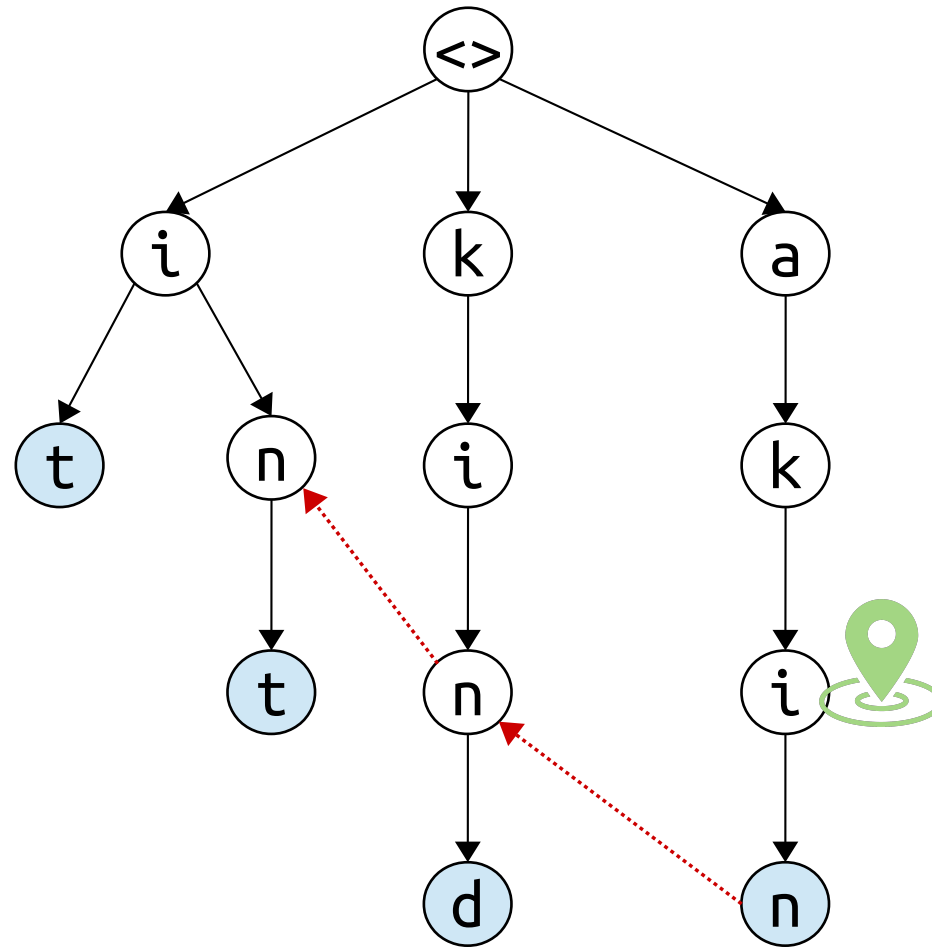


text:  a k i t

Aho-Corasick – Suffix Links

patterns:

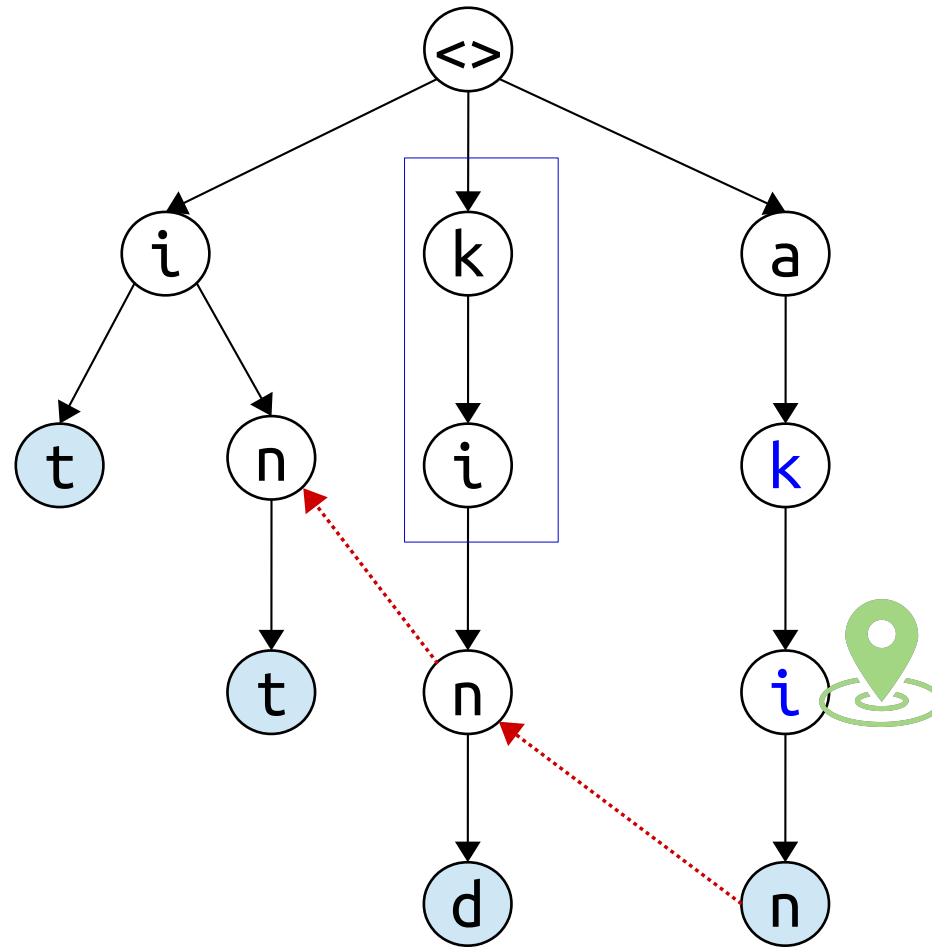
i	t		
i	n	t	
k	i	n	d
a	k	i	n



Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



text:

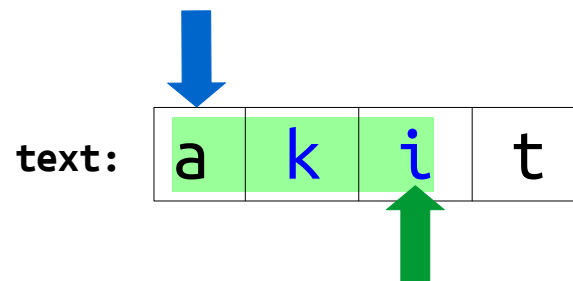
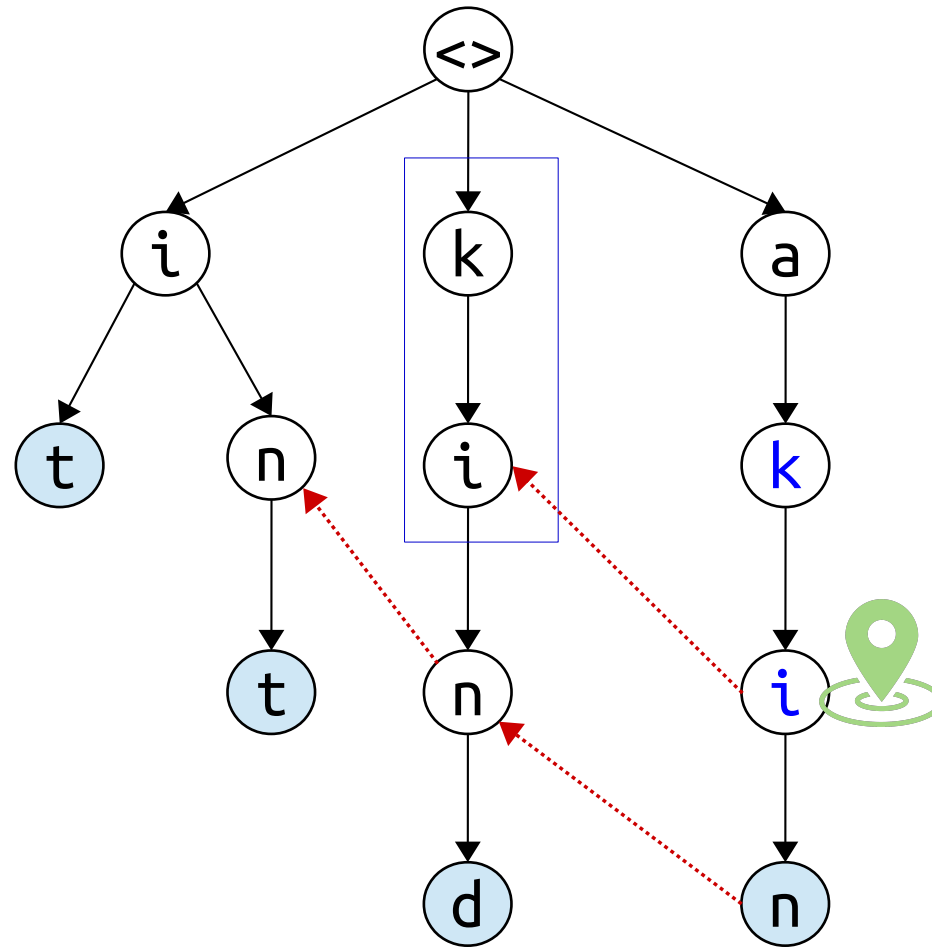
a	k	i	t
---	---	---	---

A blue arrow points down to the 'a' character, and a green arrow points up to the 'i' character.

Aho-Corasick – Suffix Links

patterns:

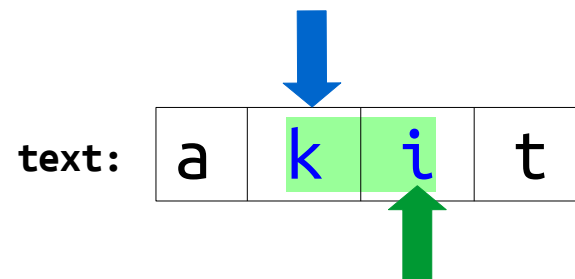
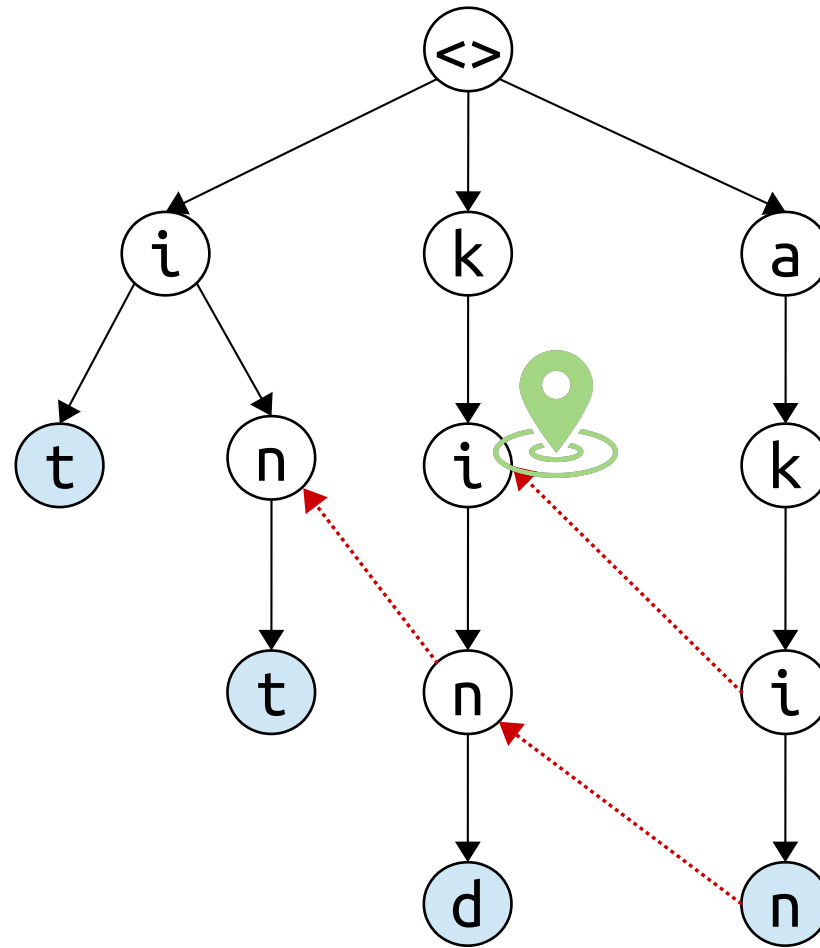
i	t		
i	n	t	
k	i	n	d
a	k	i	n



Aho-Corasick – Suffix Links

patterns:

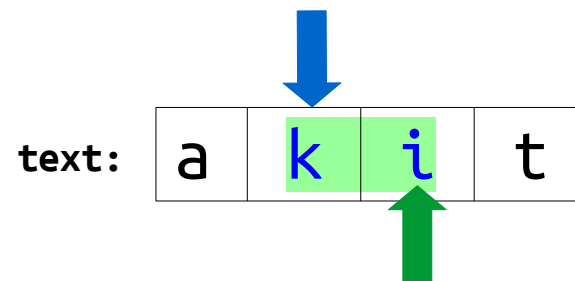
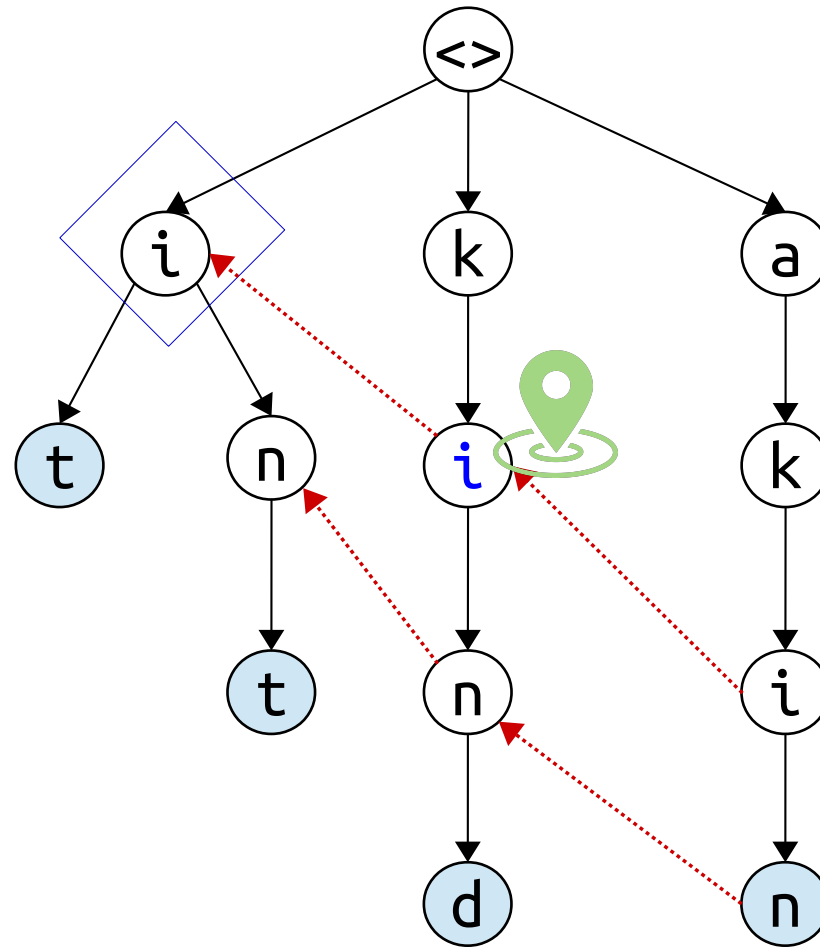
i	t		
i	n	t	
k	i	n	d
a	k	i	n



Aho-Corasick – Suffix Links

patterns:

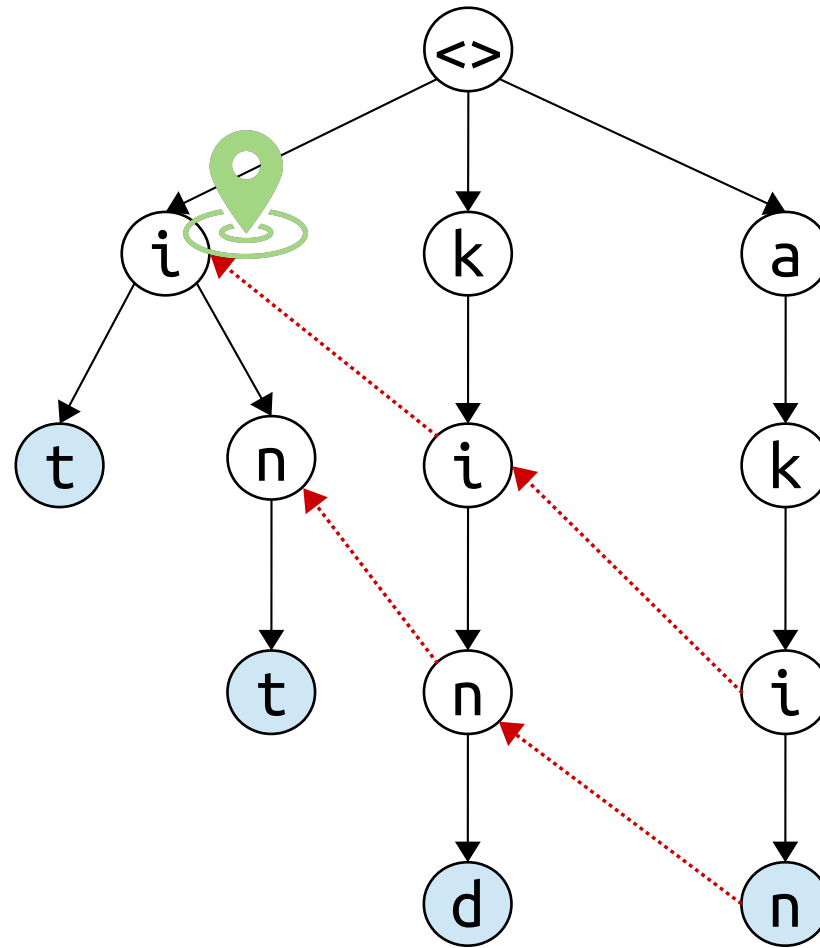
i	t		
i	n	t	
k	i	n	d
a	k	i	n



Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



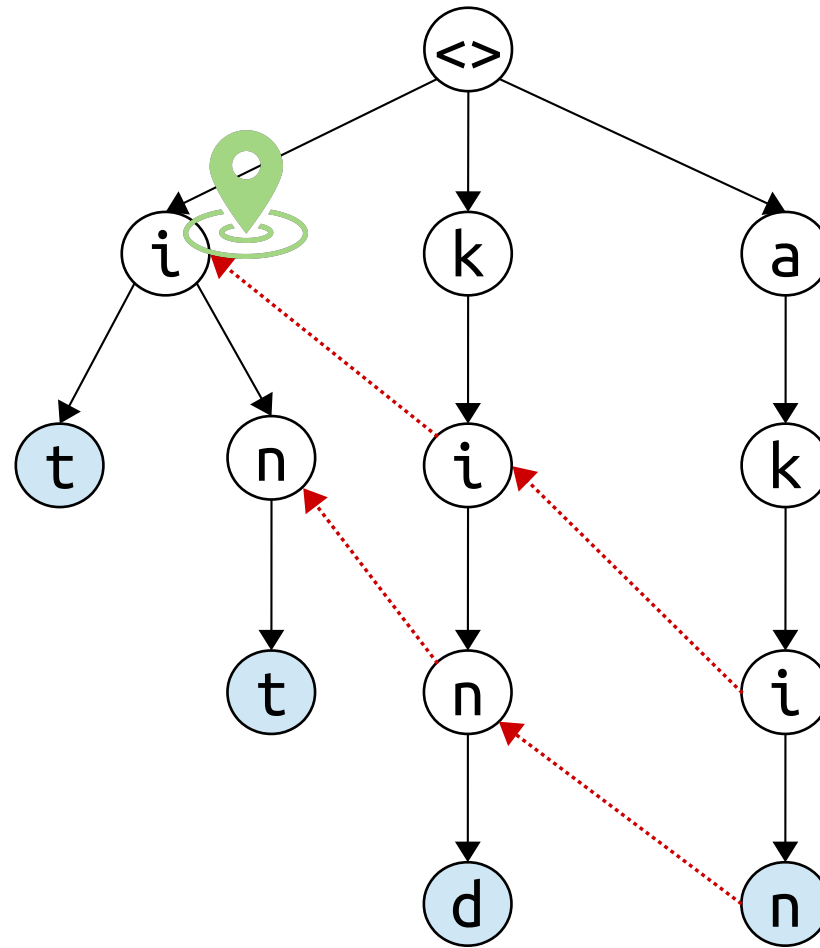
text: a k i t

A blue arrow points down to the 'i' in the text, and a green arrow points up to the 'i' in the text.

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



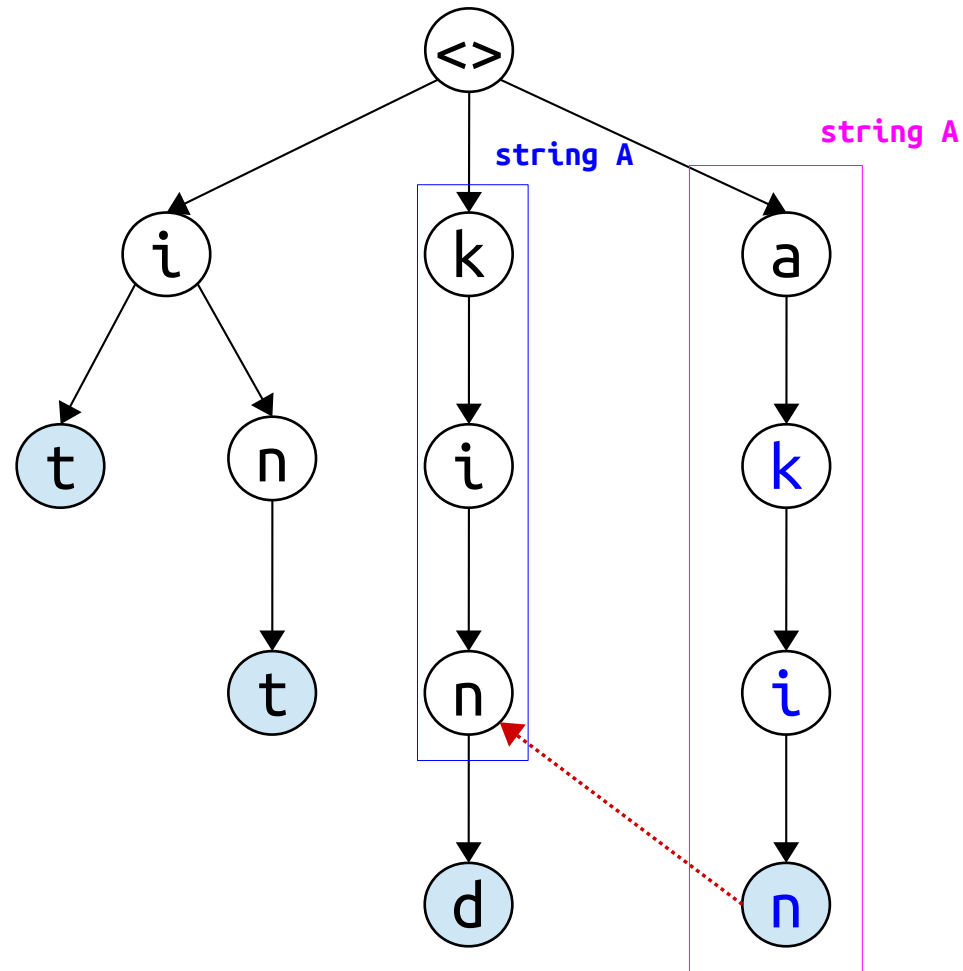
text: a k i t

A blue arrow points to the 'i' character in the text, and a green arrow points to the 't' character in the text.

Aho-Corasick – Suffix Links

patterns:

i	t		
i	n	t	
k	i	n	d
a	k	i	n



Suffix link leads from a node corresponding to **string A** to the node corresponding to **string B**

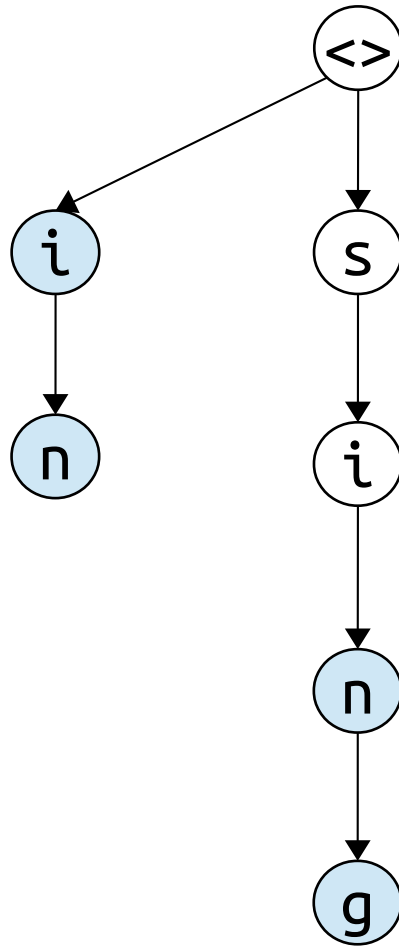
string B is in a branch of the trie and it is the **longest (proper) suffix** of **string A**.

Every node in the trie has a non-null suffix link (except the root).

Aho-Corasick – Output Links

patterns:

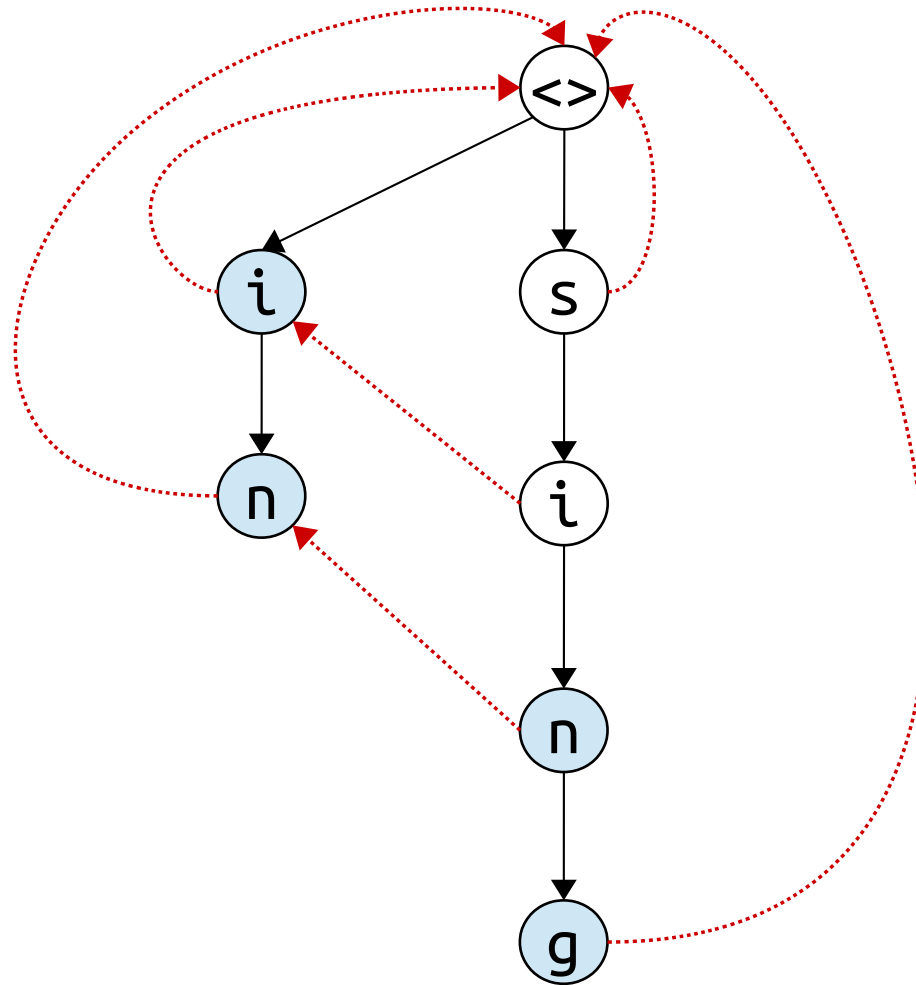
i			
i	n		
s	i	n	
s	i	n	g



Aho-Corasick – Output Links

patterns:

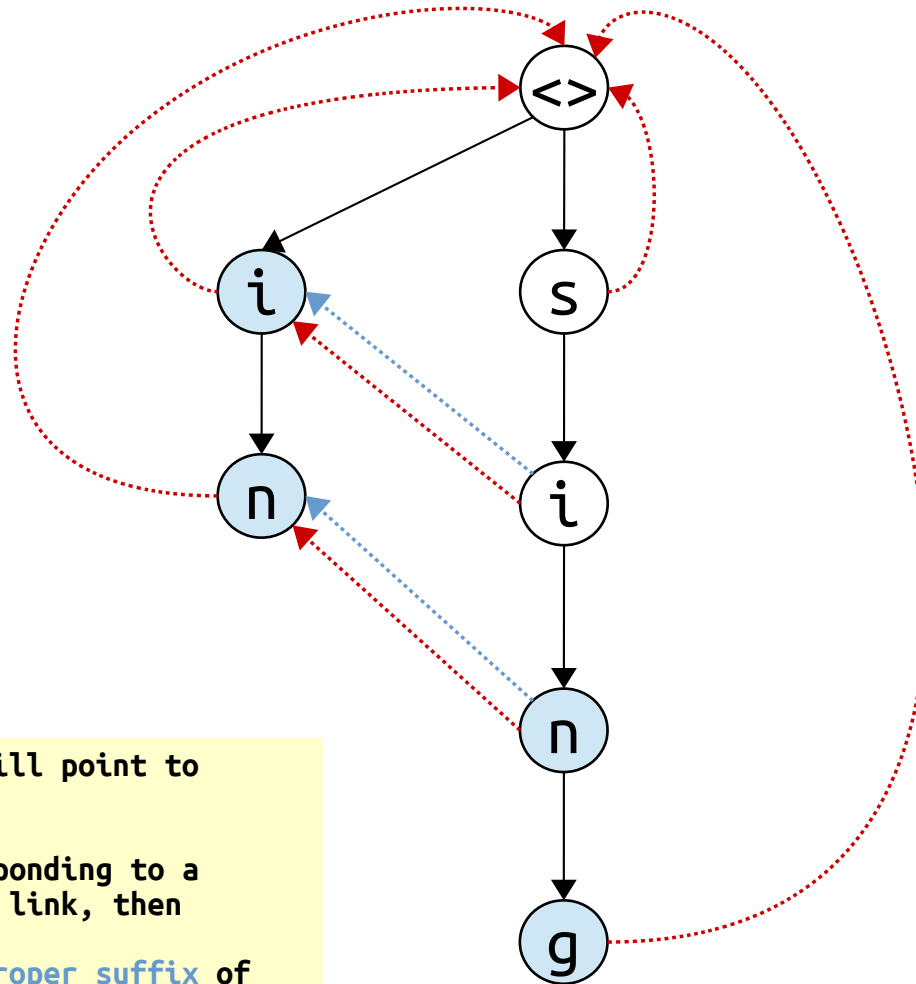
i			
i	n		
s	i	n	
s	i	n	g



Aho-Corasick – Output Links

patterns:

i			
i	n		
s	i	n	
s	i	n	g



The **output link** of a node will point to A **endOfWord** node.

When we visit a node corresponding to a **string A** that has an output link, then

It means that the **longest proper suffix** of **string A** is a pattern in the trie, so we “print” it.

Aho-Corasick – Output Links

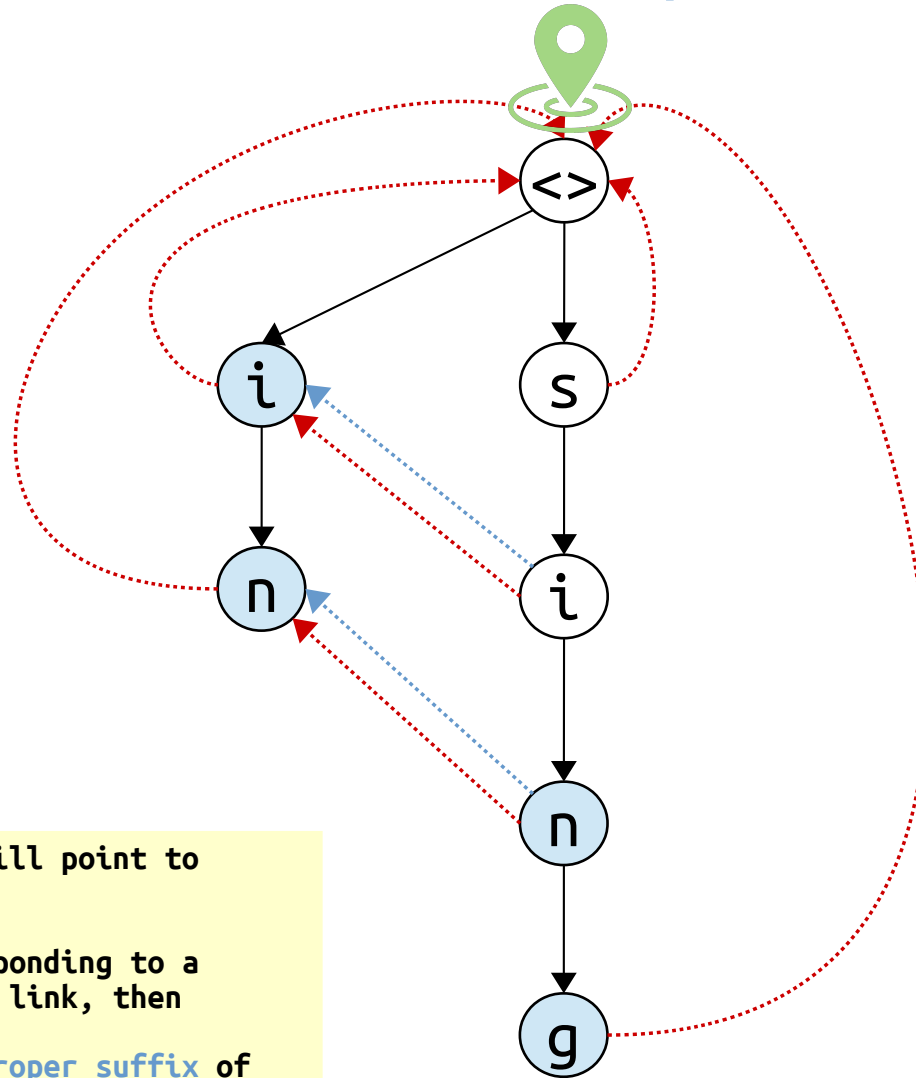
patterns:

i			
i	n		
s	i	n	
s	i	n	g

The **output link** of a node will point to A **endOfWord** node.

When we visit a node corresponding to a **string A** that has an output link, then

It means that the **longest proper suffix** of **string A** is a pattern in the trie, so we “print” it.



text:

s i n g

Aho-Corasick – Output Links

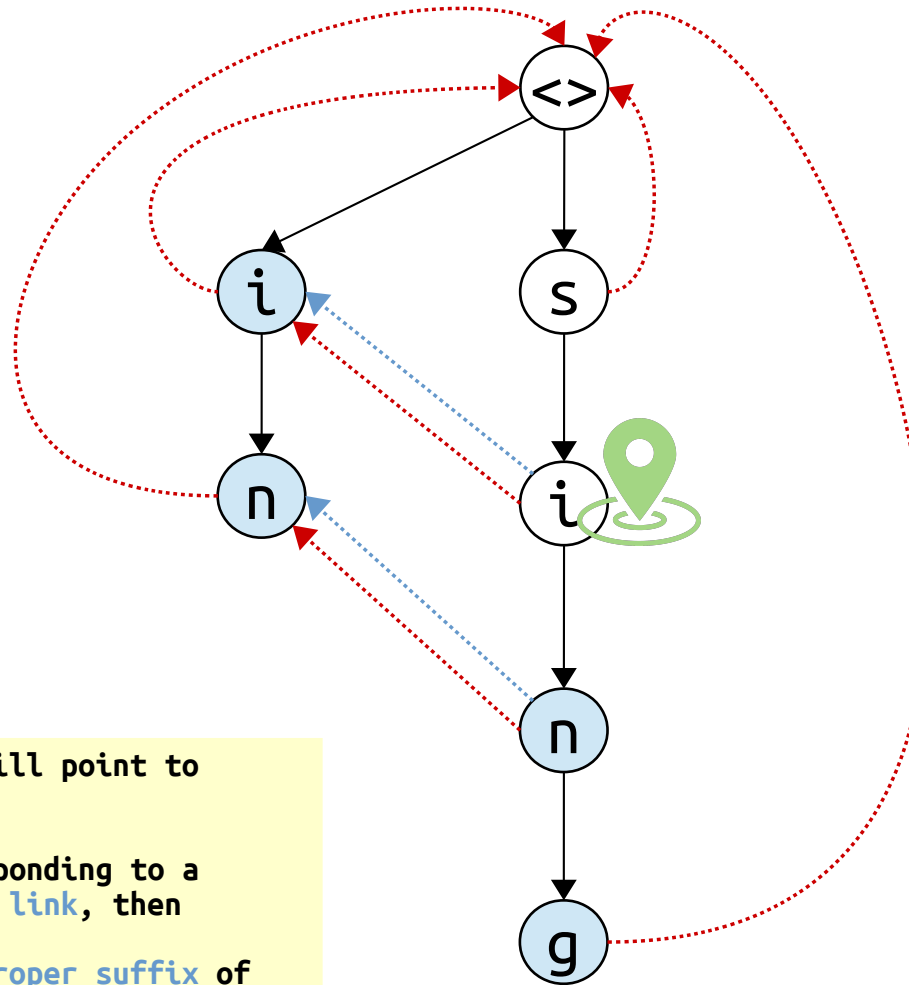
patterns:

i			
i	n		
s	i	n	
s	i	n	g

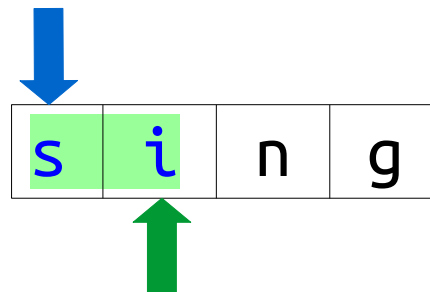
The **output link** of a node will point to A **endOfWord** node.

When we visit a node corresponding to a **string A** that has an **output link**, then

It means that the **longest proper suffix** of **string A** is a pattern in the trie, so we “print” it.



text:



pattern “i” found in the text

Aho-Corasick – Output Links

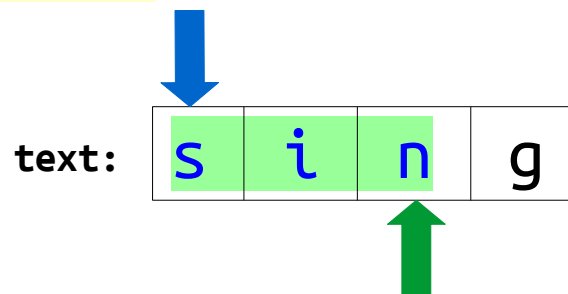
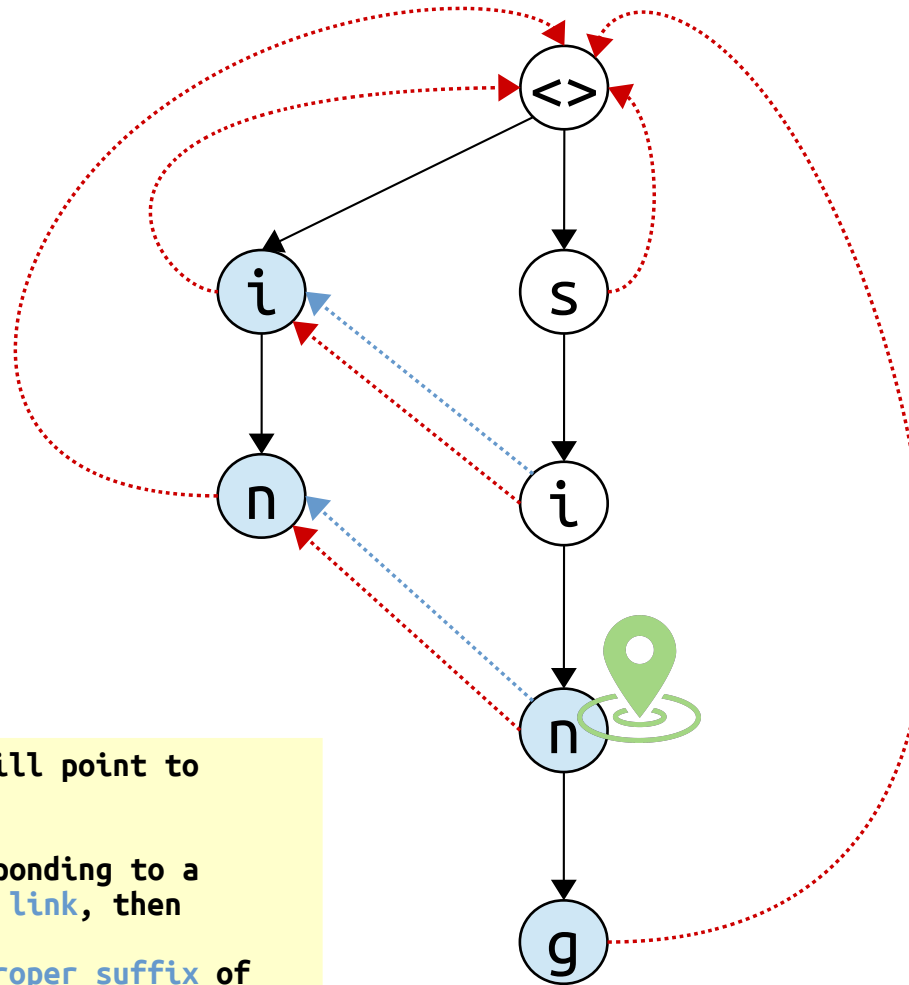
patterns:

i			
i	n		
s	i	n	
s	i	n	g

The **output link** of a node will point to A **endOfWord** node.

When we visit a node corresponding to a **string A** that has an **output link**, then

It means that the **longest proper suffix** of **string A** is a pattern in the trie, so we “print” it.



pattern “**i**” found in the text
pattern “**in**” found in the text
pattern “**sin**” found in the text

Aho-Corasick – Output Links

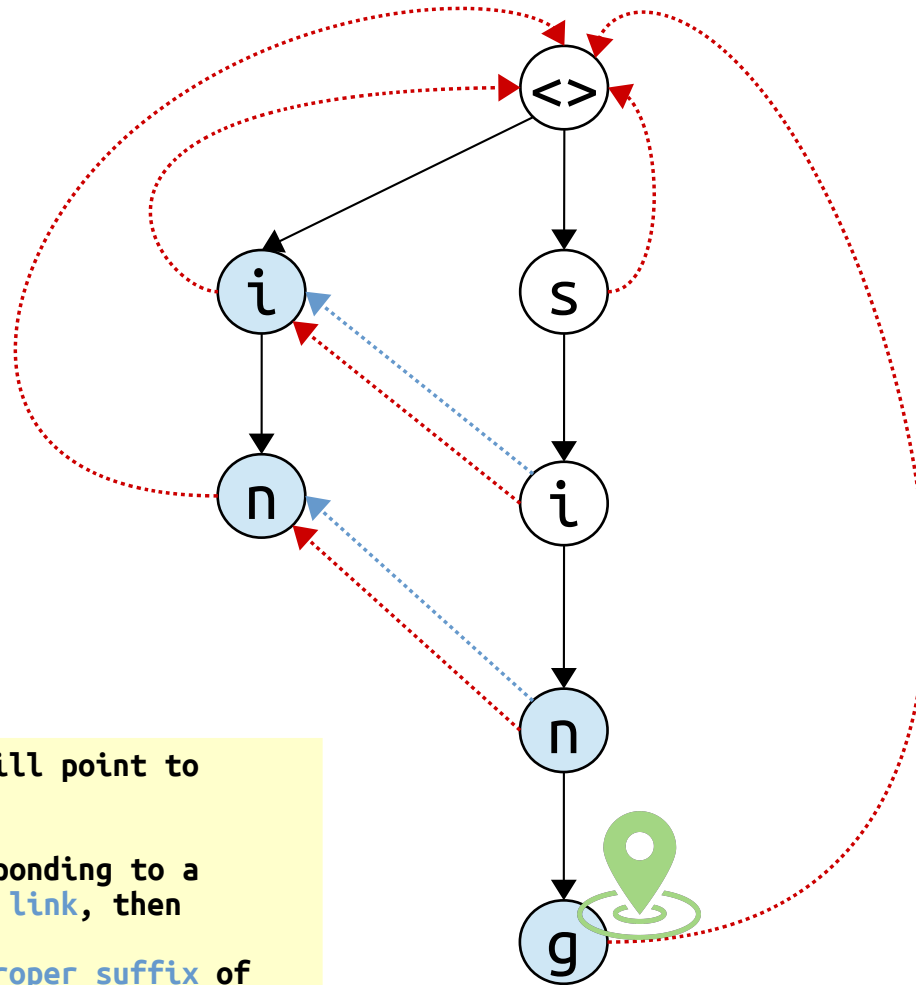
patterns:

i			
i	n		
s	i	n	
s	i	n	g

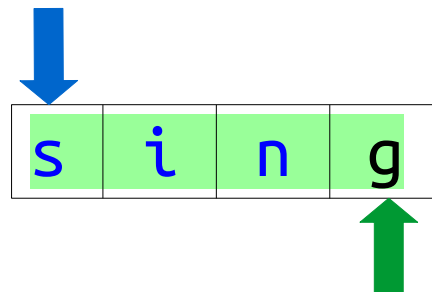
The **output link** of a node will point to A **endOfWord** node.

When we visit a node corresponding to a **string A** that has an **output link**, then

It means that the **longest proper suffix** of **string A** is a pattern in the trie, so we “print” it.



text:



pattern “i” found in the text
pattern “in” found in the text
pattern “sin” found in the text
pattern “sing” found in the text