Ukkonen's Algorithm

EXAMPLE BASED SUPPLEMENTARY MATERIAL

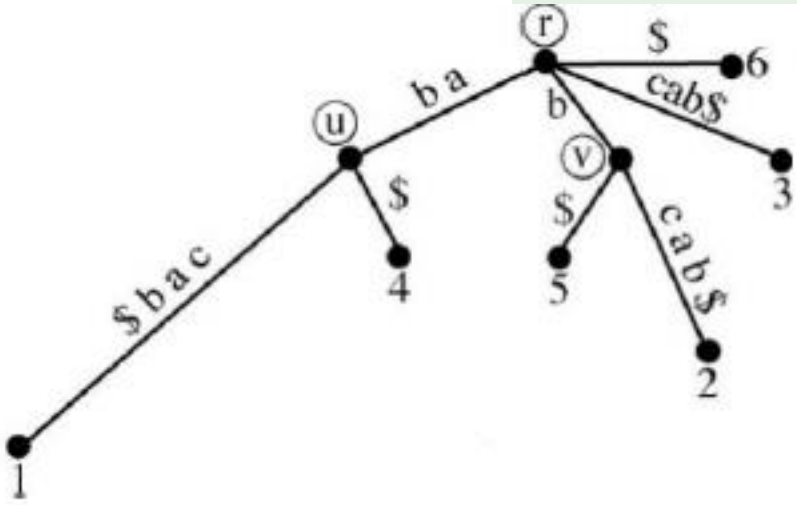
ANUJA DHARMARATNE
MONASH UNIVERSITY MALAYSIA

Suffix Tree- naïve approach

Consider the suffixes of a string as given and constructing a suffix tree to represent them.

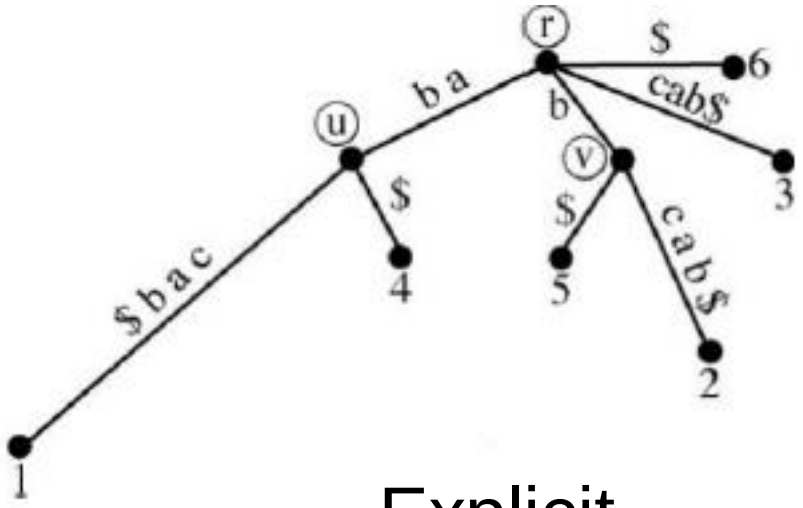
1	2	3	4	5	6
a	b	c	a	b	$

```
SUFFIX 1: str[1..6] = a b c a b $
SUFFIX 2: str[2..6] = b c a b $
SUFFIX 3: str[3..6] = c a b $
SUFFIX 4: str[4..6] = a b $
SUFFIX 5: str[5..6] = b $
SUFFIX 6: str[6..6] = $
```

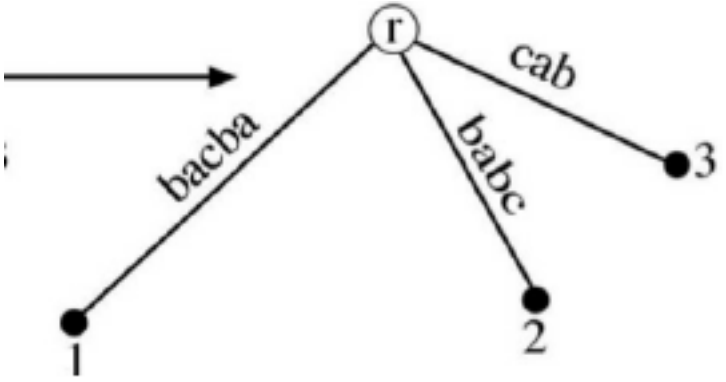


Explicit vs Implicit

1	2	3	4	5	6
a	b	c	a	b	$



Explicit

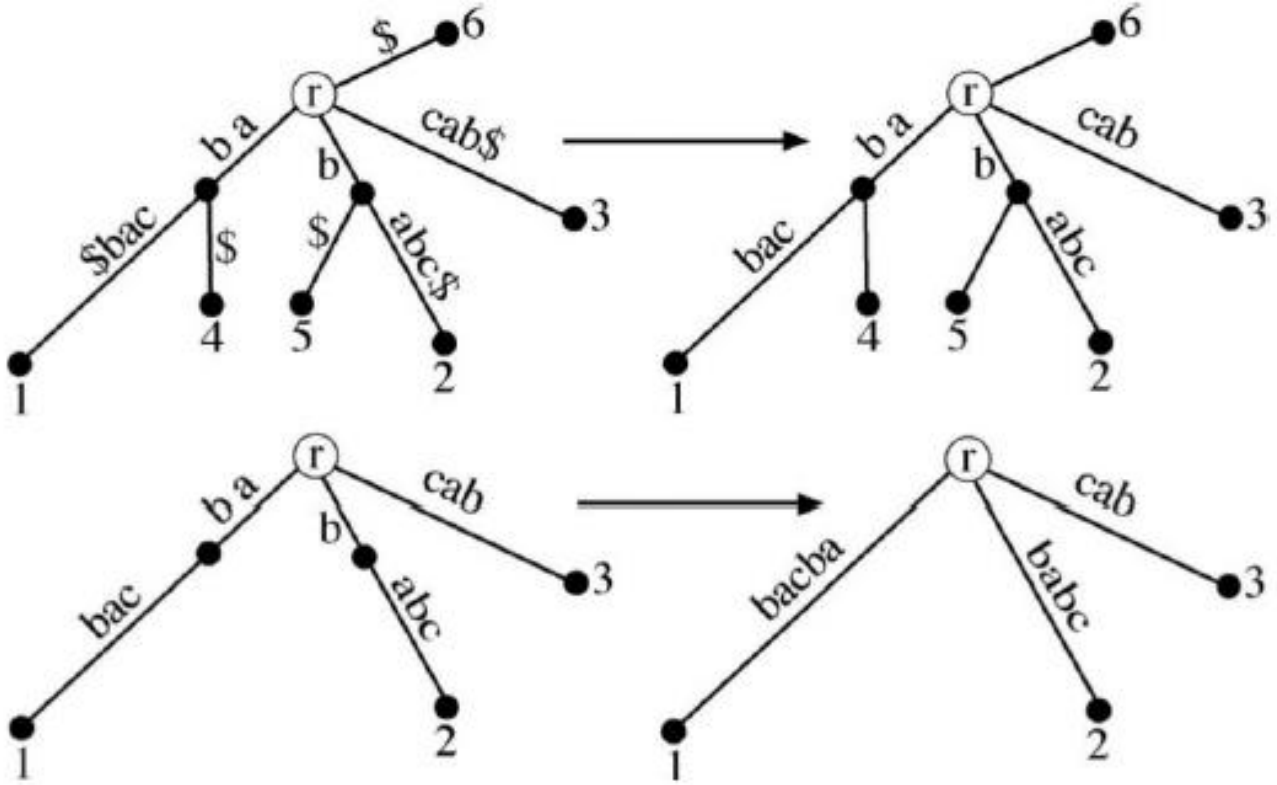


Implicit

How do we convert this Explicit suffix tree to an implicit one?

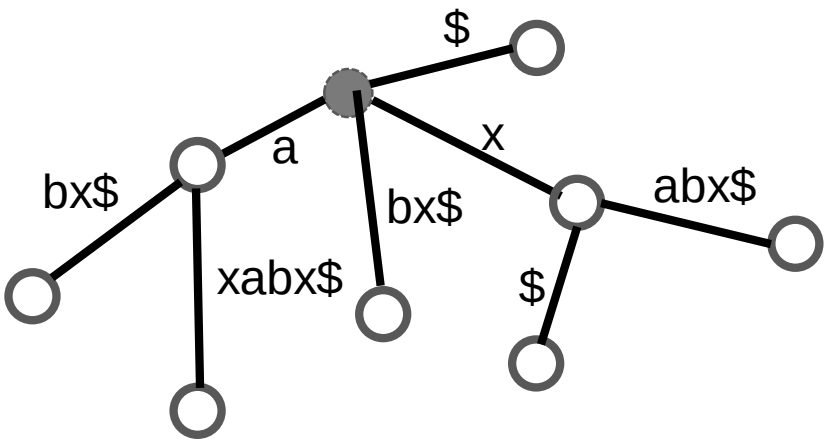
Constructing the implicit tree

- First, remove all terminal ($) characters in the regular suffix tree.
- Then, remove all edges without edge labels (i.e. substrings).
- Then, path compress the tree by removing all nodes that do not have at least two children.

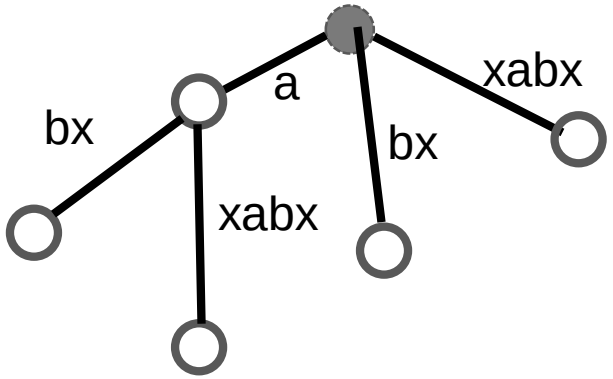


Another example

String= axabx$



Explicit



Implicit

Ukkonen's algorithm

- Constructs implicit suffix trees in phases.

Ukkonen's algorithm

- Constructs implicit suffix trees in phases.
- If there are n characters in the string, there will be n phases to construct the implicit suffix tree.
  - Let's call implicit suffix tree as **implicitST**.
  - The $(i+1)^{th}$ implicitST for the prefix string[1…i+1] is constructed at the $(i+1)^{th}$ phase.
  - $implicitST_{i+1}$ is incrementally computed using the $implicitST_i$ from the previous phase.

Ukkonen's algorithm

- Constructs implicit suffix trees in phases.
- If there are n characters in the string, there will be n phases to construct the implicit suffix tree.
  - Let's call implicit suffix tree as **implicitST**.
  - The $(i+1)^{th}$ implicitST for the prefix string[1…i+1] is constructed at the $(i+1)^{th}$ phase.
  - $implicitST_{i+1}$ is incrementally computed using the $implicitST_i$ from the previous phase.
- For example, if you got a string "abcab$", at 3^{rd} phase, it constructs the implicitST of "abc".
- Also, when constructing implicitST("abc"), it's previous phase, i.e. implicitST("ab") is used.

Ukkonen's algorithm

- Constructs implicit suffix trees in phases.
- If there are n characters in the string, there will be n phases to construct the implicit suffix tree.
  - Let's call implicit suffix tree as **implicitST**.
  - The $(i+1)^{th}$ implicitST for the prefix string[1…i+1] is constructed at the $(i+1)^{th}$ phase.
  - $implicitST_{i+1}$ is incrementally computed using the $implicitST_i$ from the previous phase.
- For example, if you got a string "abcab$", at 3^{rd} phase, it constructs the implicitST of "abc".
- Also, when constructing implicitST("abc"), it's previous phase, i.e. implicitST("ab") is used.
- Each phase has extensions.

Ukkonen's algorithm- contd..

- Each phase has suffix extensions.
- What are these extensions?

Ukkonen's algorithm- contd..

- Each phase has suffix extensions.
- What are these extensions?
  - When we are constructing the implicitST_{i+1}, what we do is extending implicitST_i by one additional character.

Ukkonen's algorithm- contd..

- Each phase has suffix extensions.
- What are these extensions?
  - When we are constructing the implicitST_{i+1}, what we do is extending implicitST_i by one additional character.
  - Therefore, when extending some suffix in the phase i + 1, the extension steps have to:
    - Find the path from the root node r, corresponding to the (current) suffix and extending it by adding the additional character.

Ukkonen's algorithm- contd..

- Each phase has suffix extensions.
- What are these extensions?
  - When we are constructing the implicitST_{i+1}, what we do is extending implicitST_i by one additional character.
  - Therefore, when extending some suffix in the phase i + 1, the extension steps have to:
    - Find the path from the root node r, corresponding to the (current) suffix and extending it by adding the additional character.
    - i.e. In extension j of phase i + 1, we need to find the end of the path from the root labeled with substring S[j..i] and extend the substring by adding the character S(i + 1) to its end

Ukkonen's algorithm- contd..

- Each phase has suffix extensions.
- What are these extensions?
  - When we are constructing the implicitST_{i+1}, what we do is extending implicitST_i by one additional character.
  - Therefore, when extending some suffix in the phase i + 1, the extension steps have to:
    - Find the path from the root node r, corresponding to the (current) suffix and extending it by adding the additional character.
    - i.e. In extension j of phase i + 1, we need to find the end of the path from the root labeled with substring S[j..i] and extend the substring by adding the character S(i + 1) to its end
- When extending suffixes, we have to follow 3 suffix extension rules.

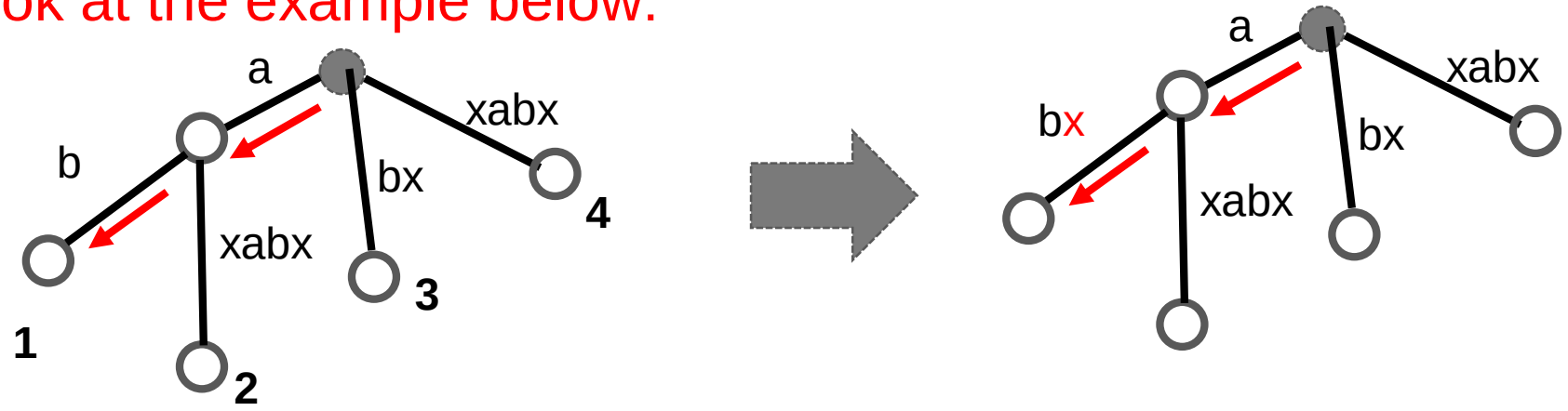
Suffix extension rule 1

If the path str[j..i] in implicitST_i ends at a leaf, adjust the label of the edge to that leaf to account for the added character str[i + 1].

Suffix extension rule 1

If the path str[j..i] in implicitST_i ends at a leaf, adjust the label of the edge to that leaf to account for the added character str[i + 1].

Look at the example below:



Assume the path of implicitST_i is ‘ab’ which ends at leaf node 1. Now we need to add str[i+1]= ‘x’. According to Rule 1, just add ‘x’ to the leaf node.

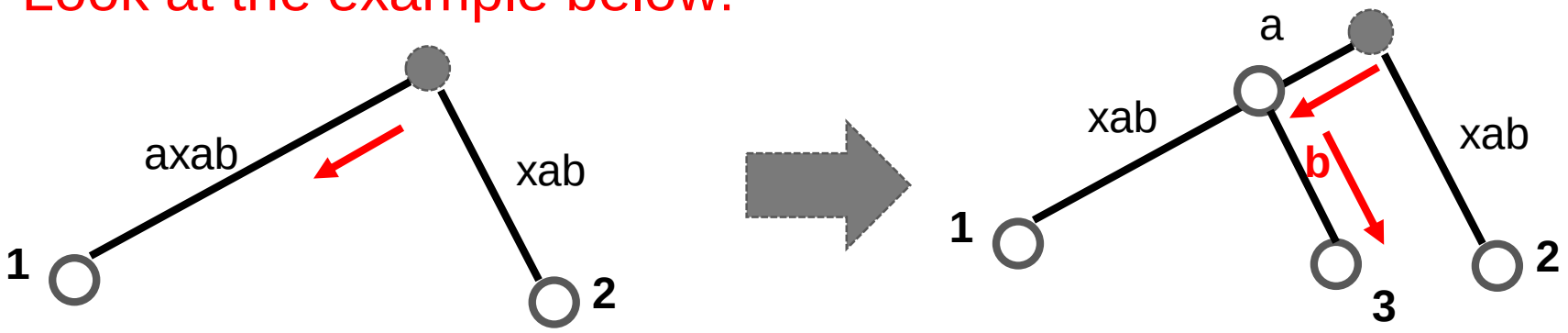
Suffix extension rule 2

If the path str[j..i] in implicitST_i does not end at a leaf, and the next character in the existing path is different to str[i+1], then split the edge and create a new node, followed by a new leaf; assign character str[i + 1] as the edge label between the new node and the leaf.

Suffix extension rule 2

If the path str[j..i] in implicitST_i does not end at a leaf, and the next character in the existing path is different to str[i+1], then split the edge and create a new node, followed by a new leaf; assign character str[i + 1] as the edge label between the new node and the leaf.

Look at the example below:



Assume the path of implicitST_i starts with an ‘a’ and now we need to add str[i+1]= ‘b’. According to Rule 2, split the edge, create a new node and a leaf and label the new edge.

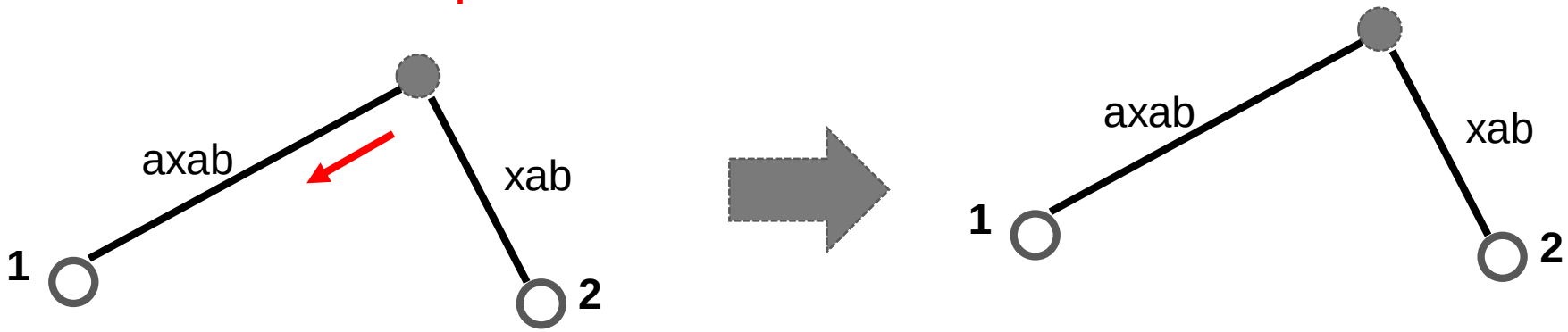
Suffix extension rule 3

If the path str[j..i] in implicitST_i does not end at a leaf, but is within some edge label, and the next character in that path is str[i+1], then str[i + 1] is already in the tree. Do nothing.

Suffix extension rule 3

If the path str[j..i] in implicitST_i does not end at a leaf, but is within some edge label, and the next character in that path is str[i+1], then str[i + 1] is already in the tree. Do nothing.

Look at the example below:



Assume the path of implicitST_i starts with an ‘a’ and now we need to add str[i+1]= ‘x’. According to Rule 3, do nothing since ‘ax’ already exists in path 1.