Suffix Sorting

Michael Liut

University of Toronto

Overview

1. Introduction/Background of the Audience
2. Suffix Sorting
3. Pedagogical Approach Taken
4. Question & Answer

Introduction/Background

Why Suffix Sorting?

- It’s a topic that is more commonly taught in Europe and Japan
  - I want our students to be exposed to this important topic too!

- It was related to my thesis
  - I am well versed to teach cutting edge content in this space

- It’s an introductory topic which makes for a good segue into the importance of the Suffix Array!
  - Searching large corpuses (e.g. Google), data compression, finding all occurrences of a particular substring, computational biology, etc.

Who is the intended talk for?

- Students enrolled in CSC-373 (Algorithm Design and Analysis)
  - ➢ This can be the introduction to divide and conquer algorithms.

<u>Background</u>

- Have taken programming classes (e.g. CSC-209, CSC-148)
- Have taken data structures classes (e.g. CSC-263, CSC-265)

Layout of Lesson

- ~5 minutes of the end of the previous class
- Assign the reading for next class
- ~20 minutes for this class
  - Recap of reading
  - Active Learning Exercise

Intended Learning Objectives

End of Class 1 + Homework

1. Students should be able to construct a Trie and Radix Tree
2. Students should understand the difference (spatially) between a Trie and a Radix Tree.

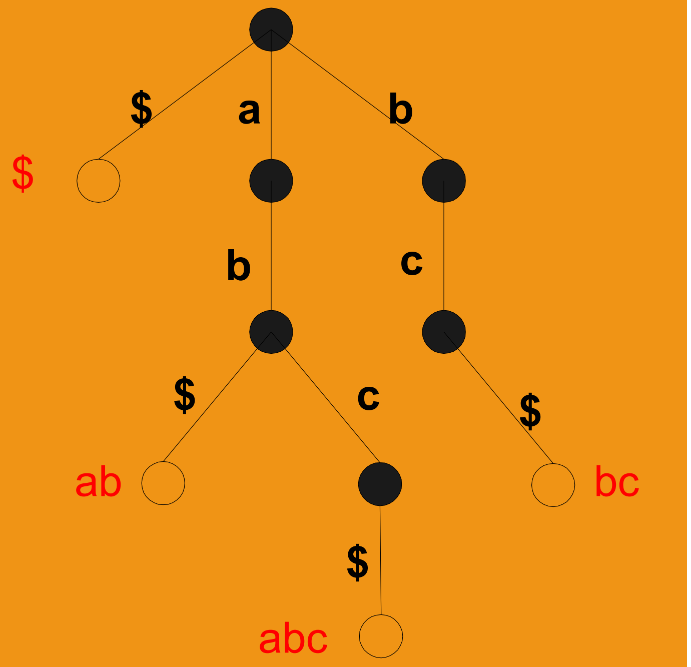
Beginning of Class 2

1. Students should be able to construct a Suffix Tree and Suffix Array
2. Students should understand the difference (spatially) between a Suffix Tree and a Suffix Array.

Trie (pronounced 'try')

- a dictionary tree (prefix tree)
  - Composed of Nodes and Links
  - Stores a set of words, each Node representing a character

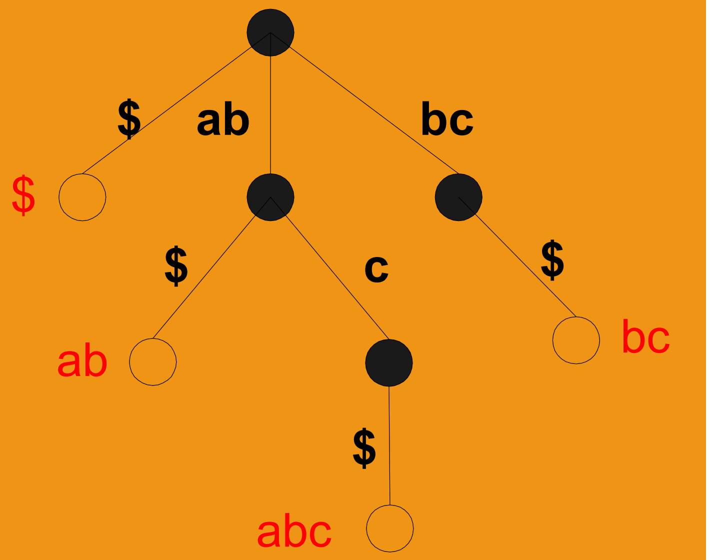
*note: the sentinel symbol $ is used to terminate the string, it is lexicographically smallest.



A Trie on *X* = { ab, abc, bc }

Radix Tree

- a Trie with a compressed chain of nodes
  - Each internal node having at least 2 children
  - *AKA*: Patricia Trie, Compacted Trie, and Radix Trie



A Radix Tree on *X* = { ab, abc, bc }

Homework

- Algorithms, 4th edition by Sedgewick and Wayne.

- Read Chapters:
  - 5.1 (String Sorting)
  - 5.2 (Tries)
  - 6. Pages: 875-878 (Sorting Suffixes and Suffix Arrays)

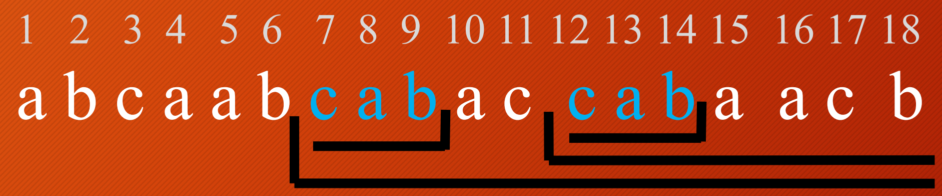
- After reading, check to ensure you've me today's intended learning objectives!

Suffix Sorting

What is the goal?

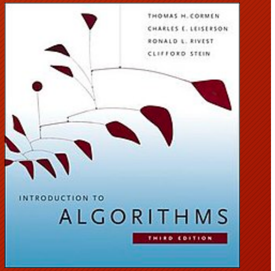
- To identify all occurrences of a substring fast and efficiently.
  - Think of trying to catalogue all the substrings of your favourite CS textbook!

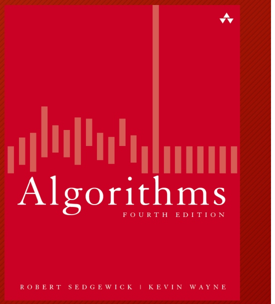
- Instead of re-scanning the string every time we are looking for a pattern, we “prepare” a data structure to do the search easily.
  - The idea is that any substring is a prefix of a suffix!



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

a b c a a b c a b a c c a b a a c b





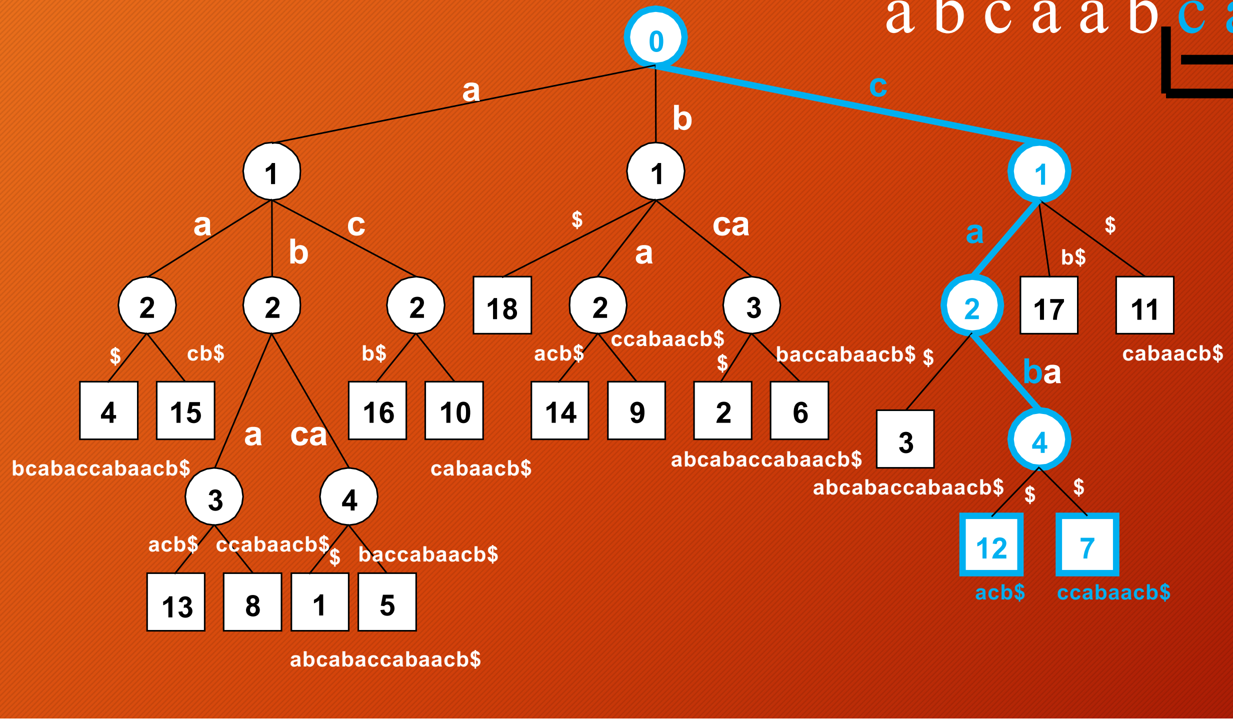
UNIVERSITY OF TORONTO

Suffix Tree

- Suffix Tree: a Suffix Radix Tree

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

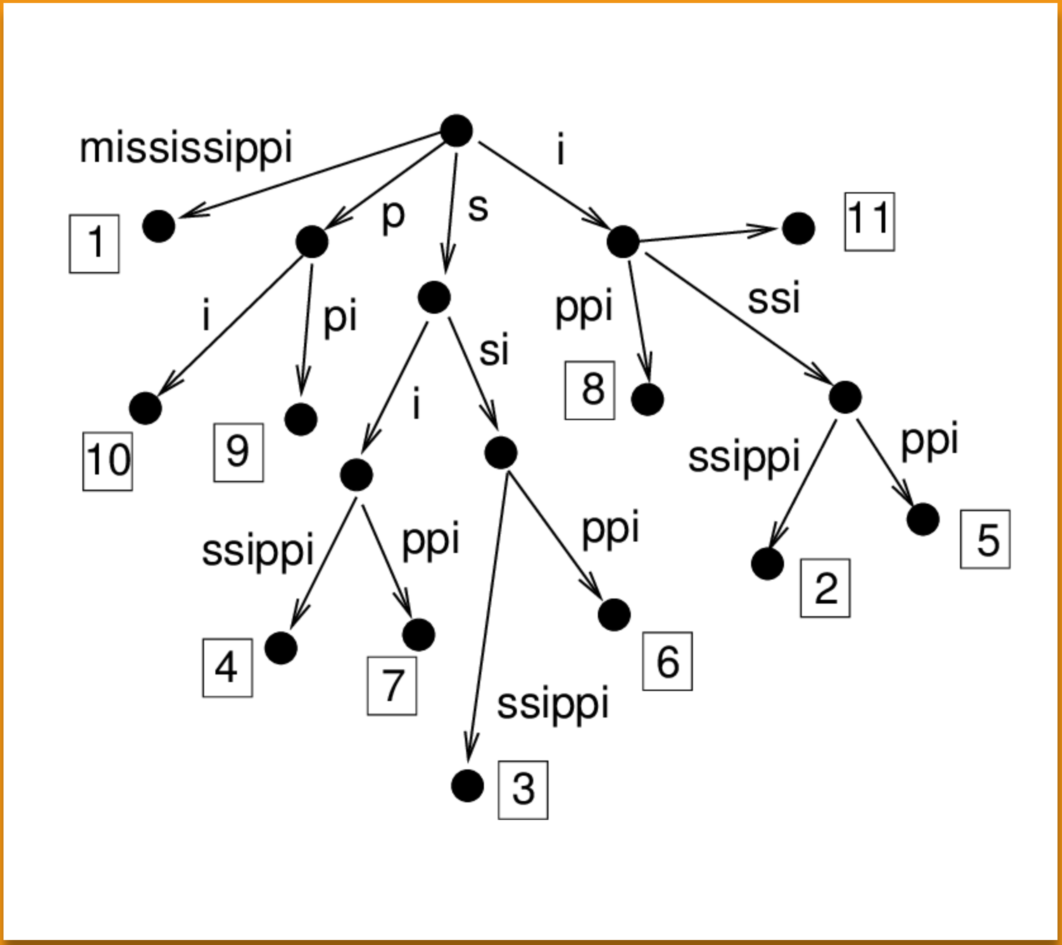
a b c a a b c a b a c c a b a a c b



Worksheet: Task 1!

- Let’s construct a Suffix Tree for the word “Mississippi”

1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i



Issues with Suffix Trees

- Require a lot of space! Typically 10-20x more space than the original string!
- Even using some compression techniques, it’s still ~5x bigger than the original string!

Suffix Array

- Introduced by Manber & Myers (1990).
- Sorted array of all suffixes of a particular string.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

a b c a a b c a b a c c a b a a c b

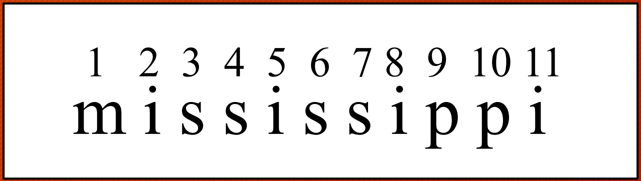
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SA	4	15	13	8	1	5	16	10	18	14	9	2	6	3	12	7	17	11

Suffix Array

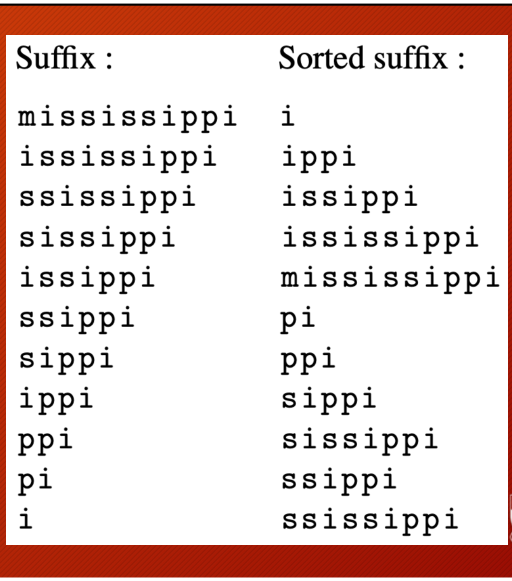
- Best algorithms were O (n log n)
- In 2003, several researchers emulated Farach's approach to provide a recursive linear algorithm for suffix sorting.
- In 2015 Baier introduced a non-recursive linear suffix sorting algorithm.

Worksheet: Task 2!

- Let’s construct a Suffix Array from “Mississippi”



1	2	3	4	5	6	7	8	9	10	11
m	i	s	s	i	s	s	i	p	p	i



Suffix :	Sorted suffix :
mississippi	i
ississippi	ippi
ssissippi	issippi
sissippi	ississippi
issippi	mississippi
ssippi	pi
sippi	ppi
ippi	sippi
ppi	sissippi
pi	ssippi
i	ssissippi

	1	2	3	4	5	6	7	8	9	10	11
SA	5	4	11	9	3	10	8	2	7	6	1
SA^{-1}	11	8	5	2	1	10	9	7	4	6	3

Question: if we included $ where would it go?

Answer: at the beginning, it’s the smallest!

The Agenda for Next Week

- Suffix Array + Longest Common Prefix (LCP)

- Suffix Tree Algorithms
  1. Weiner, then McCreight 1973/1976
  2. Ukkonen, 1995
  3. Farach, 1997

- Recursive vs. Iterative implementation

UNIVERSITY OF TORONTO

Pedagogical Approach

Pedagogical Approach

- The 3 “P”s: Prepare, Practice, and Perform
  - Similar to that in CSC-108 and CSC-148
- Actively learning in the classroom, but also applying these experientially through homework assignments and weekly labs.
- Breaking up topics into foundational building blocks for them to tackle one step at a time (divide and conquer ☺).

Q & A

Thanks for listening! ☺

Does anyone have any questions?

References

Baier, U. Linear-time Suffix Sorting. Ulm University, Germany. November 2015.

Franek, F. Suffix-based text indices, construction algorithms, and applications.
2nd CanaDAM Conference, Centre de Recherches, Mathématiques in Montréal. May 2009.

Liut, M. Computing Lyndon Arrays. McMaster University, Canada. September 2019.

Sedgewick, R. and Wayne, K. Algorithms (4th ed.). Addison-Wesley. March 2011.

Yang, J. Algorithm of Suffix Tree. Osaka University, Japan. November 2011.