Suffix Trees

Pavel Pevzner

Department of Computer Science and Engineering
University of California at San Diego

Algorithms on Strings
Data Structures and Algorithms

This slide desk is incomplete.
For the complete set of frames,
please see our videos in the
Algorithms on Strings course on Coursera
(Algorithms and Data Structures Specialization)

Outline

- **From Genome Sequencing to Pattern Matching**
- Brute Force Approach to Pattern Matching
- Herding Patterns into Trie
- Herding Text into Suffix Trie
- From Suffix Tries to Suffix Trees

The Newspaper Problem

stack of NY Times, June 27, 2000

The Newspaper Problem

stack of NY Times, June 27, 2000

stack of NY Times, June 27, 2000
on a pile of dynamite

The Newspaper Problem

The Newspaper Problem

The Newspaper Problem

The Newspaper Problem

The Newspaper Problem as an Overlapping Puzzle

The Newspaper Problem as an Overlapping Puzzle

Millions of Copies of a Genome

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

Breaking the Genomes at Random Positions

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTATCGGATCAGCTACCACATCGTAGC

Generating Short Substring (Reads)

CTGATGA TGGACTACGCTAC TACTGCTAG CTGTATTACG ATCAGCTACCACA TCGTAGCTACG ATGCATTAGCAA GCTATCGGA TCAGCTACCA CATCGTAGC

CTGATGATG GACTACGCT ACTACTGCTA GCTGTATTACG ATCAGCTACC ACATCGTAGCT ACGATGCATTA GCAAGCTATC GGATCAGCTAC CACATCGTAGC

CTGATGATGG ACTACGCTAC TACTGCTAGCT GTATTACGATC AGCTACCAC ATCGTAGCTACG ATGCATTAGCA AGCTATCGG A TCAGCTACCA CATCGTAGC

CTGATGATGGACT ACGCTACTACT GCTAGCTGTAT TACGATCAGC TACCACATCGT AGCTACGATGCA TTAGCAAGCT ATCGGATCA GCTACCACATC GTAGC

Burning Some Reads

CTGATGA TGGACTACGCTAC TACTGCTAG CTGTATTACG ATCAGCTACCACA TCGTAGCTACG ATGCATTAGCAA GCTATCGGA TCAGCTACCA CATCGTAGC

CTGATGATG GACTACGCT ACTACTGCTA GCTGTATTACG ATCAGCTACC ACATCGTAGCT ACGATGCATTA GCAAGCTATC GGATCAGCTAC CACATCGTAGC

CTGATGATGG ACTACGCTAC TACTGCTAGCT GTATTACGATC AGCTACCAC ATCGTAGCTACG ATGCATTAGCA AGCTATCGG A TCAGCTACCA CATCGTAGC

CTGATGATGGACT ACGCTACTACT GCTAGCTGTAT TACGATCAGC TACCACATCGT AGCTACGATGCA TTAGCAAGCT ATCGGATCA GCTACCACATC GTAGC

Assembling Genome

CTGATGA TGGACTACGCTAC TACTGCTAG CTGTATTACG ATCAGCTACCACA TCGTAGCTACG ATGCATTAGCAA GCTATCGGA TCAGCTACCA CATCGTAGC

CTGATGATG GACTACGCT ACTACTGCTA GCTGTATTACG ATCAGCTACC ACATCGTAGCT ACGATGCATTA GCAAGCTATC GGATCAGCTAC CACATCGTAGC

CTGATGATGG ACTACGCTAC TACTGCTAGCT GTATTACGATC AGCTACCAC ATCGTAGCTACG ATGCATTAGCA AGCTATCGG A TCAGCTACCA CATCGTAGC

CTGATGATGGACT ACGCTACTACT GCTAGCTGTAT TACGATCAGC TACCACATCGT AGCTACGATGCA TTAGCAAGCT ATCGGATCA GCTACCACATC GTAGC

cow
2009
horse
2007
opossum
2007
macaque
2006
dog
2005
chimpanzee
2005
rat
2004
mouse
2002
human
2001

Why Do We Sequence 1000s of Species?

Applications in

- **medicine** (mouse genome)
- **agriculture** (rice genome)
- **biotechnology** (genomes of energy-producing bacteria),
- etc., etc., etc.

Few Mutations Can Make a Big Difference…

- Different people have slightly different genomes: on average, roughly 1 mutation in 1000 nucleotides.

- The 1 in 1000 nucleotides difference accounts for height, high cholesterol susceptibility, and 7000+ genetic diseases.

```
CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGA
TCAGCTACCACATCGTAGCTACGATGCATTAGCAAGCTAT
CGATCGATCGATCGATTATCTACGATCGATCGATCGATCA
CTATACGAGCTACTACGTACGTACGATCGCGGGACTATTA
TCGACTACAGATAAAACATGCTAGTACAACAGTATACATA
GCTGCGGGATACGATTAGCTAATAGCTGACGATATCCGAT

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGA
TCAGCTACAACATCGTAGCTACGATGCATTAGCAAGCTAT
CGATCGATCGATCGATTATCTACGATCGATCGATCGATCA
CTATACGAGCTACTACGTACGTACGATCGCGTGACTATTA
TCGACTACAGATGAAACATGCTAGTACAACAGTATACATA
GCTGCGGGATACGATTAGCTAATAGCTGACGATATCCGAT
```

Emergence of Personalized Medicine

- **2010:** Nicholas Volker became the first child to be saved by genome sequencing.
  - Doctors could not diagnose his condition; he went through dozens of surgeries.
  - Sequencing revealed a mutation in a gene linked to a defect in his immune system.
  - This led doctors to use immunotherapy, which saved the child.

From Reference Genome to Personal Genomes

- **Reference human genome** assembled in 2000.

```
CTGAGGATGGACTACGCTACTACTGATAGCTGTTT
```

reference genome

```
GAGGA        CCACG        TGA-AG
CTGA     GGACC      ACTAC  A-AGCT
      GATGG    ACGCT              TGTTT
```

reads

```
CTGAGGATGGACCACGCTACTACTGA-AGCTGTTT
```

personal genome

Exact Pattern Matching

- Where does a read match the reference genome *exactly*?

- **Pattern Matching Problem:**
  - **Input:** A string *Pattern* (read) and a string *Text* (genome).
  - **Output:** All positions in *Text* where *Pattern* appears as a substring.

Approximate Pattern Matching

- Where does a read match the reference genome *approximately*?

- **Approximate Pattern Matching Problem:**
  - **Input:** A string *Pattern*, a string *Text,* and an integer *d*
  - **Output:** All positions in *Text* where *Pattern* appears as a substring with at most *d* mismatches.

Multiple Pattern Matching

- Where do billions of reads match the reference genome?

- **Multiple Pattern Matching Problem:**
  - **Input:** A set of strings *Patterns* and a string *Text.*
  - **Output:** All positions in *Text* where a string from *Patterns* appears as a substring.

Outline

- From Genome Sequencing to Pattern Matching
- **Brute Force Approach to Pattern Matching**
- Herding Patterns into Trie
- Herding Text into Suffix Trie
- From Suffix Tries to Suffix Trees

A Brute Force Approach to Exact Pattern Matching

Genome

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTAC**C**ACATCGTAGCTAC

Pattern drives along *Text*

p a n a m a b a n a n a s *Text*

n a n a *Pattern*

panamabananas *Text*

nana *Pattern*

p a n a m a b a n a n a s *Text*

n a n a *Pattern*

p a **n** a m a b a n a n a s *Text*

n a n a *Pattern*

p a **n a** m a b a n a n a s *Text*

n a n a *Pattern*

p a **n a m** a b a n a n a s *Text*

n a n a *Pattern*

p a n **a** m a b a n a n a s *Text*

n a n a *Pattern*

p a n a **m** a b a n a n a s *Text*

n a n a *Pattern*

p a n a m **a** b a n a n a s *Text*

n a n a *Pattern*

p a n a m a **b** a n a n a s *Text*

n a n a *Pattern*

p a n a m a b **a** n a n a s *Text*

n a n a *Pattern*

p a n a m a b a **n** a n a s *Text*

n a n a *Pattern*

p a n a m a b a **n a** n a s *Text*

n a n a *Pattern*

p a n a m a b a **n a n** a s *Text*

n a n a *Pattern*

Pattern Found!

panamaba**nana**s *Text*

nana *Pattern*

p a n a m a b a n **a** n a s *Text*

n a n a *Pattern*

Brute Force Approach Is Fast!

- **single** *Pattern*: O(|*Text*| • |*Pattern*|)

The runtime of the **Knuth-Morris-Pratt** algorithm: O(|*Text*|)

(wait for the lecture on algorithmic challenges in pattern matching)

Brute Force Approach is Slow for **Billions** of Patterns

- **single** *Pattern*: $O(|Text| \bullet |Pattern|)$
- **multiple** *Patterns*:

$$O\Big(\sum_{\substack{\text{all strings } Pattern \\ \text{In } Patterns}} |Text| \bullet |Pattern|\Big)$$

Brute Force Approach is Slow for Billions of Patterns

- **single** *Pattern*: $O(|Text| \bullet |Pattern|)$
- **multiple** *Patterns*:

$$O\Big(\sum_{\substack{\text{all strings } Pattern \\ \text{In } Patterns}} |Text| \bullet |Pattern|\Big) =$$

$$O(|Text| \bullet |Patterns|)$$

For human genome:

- $|Text| \approx 3*10^9$
- $|Patterns| \approx 10^{12}$

Outline

- From Genome Sequencing to Pattern Matching
- Brute Force Approach to Pattern Matching
- **Herding Patterns into Trie**
- Herding Text into Suffix Trie
- From Suffix Tries to Suffix Trees

Patterns Travel One at a Time

Genome

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTAC**C**ACATCGTAGCTAC

Herding Patterns onto a Bus

Genome

CTGATGATGGACTACGCTACTACTGCTAGCTGTATTACGATCAGCTAC**C**ACATCGTAGCTAC

Patterns

banana
pan
nab
antenna
bandana
ananas
nana

Patterns

banana

pan

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana

pan

and

nab

antenna

bandana

ananas

nana

Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana

Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana

Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana

Trie(*Patterns*)

Patterns

banana
pan
and
nab
antenna
bandana
ananas
nana

p a n a m a b a n a n a s

TrieMatching(*Text, Patterns*):

drive Trie(*Patterns*) along *Text* at each position of *Text*

- walk down Trie(*Patterns*) by spelling symbols of *Text*
- a pattern from *Patterns* matches *Text* each time you reach a leaf!

For simplicity, we assume that no pattern is a substring of another pattern

p a n a m a b a n a n a s

p a n a m a b a n a n a s

panamabananas

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a b a n a n a s

p a n a m a **b** a n a n a s

p a n a m a **b a** n a n a s

p a n a m a **b a n** a n a s

p a n a m a **b a n a** n a s

p a n a m a **b a n a n** a s

panama**bananas**

p a n a m a b a n a n a s

p a n a m a b **a n** a n a s

p a n a m a b **a n a** n a s

p a n a m a b **a n a n** a s

p a n a m a b **a n a n a** s

p a n a m a b **a n a n a s**

p a n a m a b a **n** a n a s

p a n a m a b a **n a** n a s

p a n a m a b a **n a n** a s

p a n a m a b a **n a n a** s

p a n a m a b a n **a** n a s

p a n a m a b a n **a n** a s

p a n a m a b a n **a n a** s

p a n a m a b a n **a n a s**

p a n a m a b a n a **n** a s

p a n a m a b a n a **n a** s

p a n a m a b a n a **n a s**

p a n a m a b a n a n **a** s

p a n a m a b a n a n **a** **s**

p a n a m a b a n a n a **s**

Our Bus Is Fast!

- Runtime of brute force approach:
  - O(|*Text*| • |*Patterns*|)

Here patterns is the sum of length of all patterns

- Runtime of **TrieMatching**:
  - O(|*Text*| * |*LongestPattern*|)

If we shorten the text each time we found a match

Trie construction takes O(|*Patterns*|) time

Memory Footprint of **TrieMatching**

hence using tries dramatically improves on the brute force approach with respect to speed but becomes impractical with respect to memory.

- Our trie has 30 edges
- # edges = O(|*Patterns*|)!
- For human genome: |*Patterns*|≈ 10^{12}

Outline

- From Genome Sequencing to Pattern Matching
- Brute Force Approach to Pattern Matching
- Herding Patterns into Trie
- **Herding Text into Suffix Trie**
- From Suffix Tries to Suffix Trees

New Idea: Packing *Text* onto a Bus

- Generate all suffixes of *Text*
- Form a trie out of these suffixes (**suffix trie**)
- For each *Pattern*, check if it can be spelled out from the root downward in the suffix trie

Instead of packing patterns into a bus,let's pack text, into bus. Let me explain how we can do this. We'll generate all suffixes of text. And form a trie out of all the suffixes. It will be larger. A rather large suffix trie. For each pattern, we can check ifit can be spelled out from the root downward in the suffix trie. So we are building a verylarge bus this time.

Root

p a n a m a b a n a n a s

Root

p a n a m a b a n a n a s $

Adding “$” sign in the end (we’ll explain later why)

p a n a m a b a n a n a s $

panamabananas$
Root
a
n
a
m
a
b
a
n
a
n
a
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$

panamabananas$
Root
a
n
a
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$

panamabananas$

panamabananas$

panamabananas$

panamabananas$

panamabananas$

p a n a m a b a n a n a s $
Root
a
b
m
n
p
a
n
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
m
n
a
b
a
n
a
n
a
s
$
a
s
$
b
m
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
n
a
n
a
m
n
a
b
a
n
a
n
a
s
$
a
m
a
b
a
n
a
n
a
s
$

panamabanas$

p a n a m a b a n a n a s $
Root
a
b
m
n
b
m
p
a
n
n
a
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
n
s
$
a
b
a
n
a
n
a
s
$
a
s
$
a
m
a
b
a
n
a
n
a
s
$

panamabanan**as$**
Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
n
a
n
a
s
$
n
a
s
$
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$

panamabanana s $
Root
a
b
m
n
s
$
p
a
n
s
$

SuffixTrie(*Text*)

nana
a
b
m
n
s
$
b
m
n
p
a
n
s
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
n
s
a
b
a
n
a
n
a
s
$
m
n
s
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$

Root
a
b
m
n
s
$
p
n
nana

Root
a
b
m
n
s
$
p
a
n
a
nana
n
a
m
n
s
$
b
m
n
a
m
n
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$

Root
a
b
m
n
s
$
p
a
n
a
m
n
s
nana

Root
nana
Match!

banana
Root
a
b
m
n
s
$
p
a
n
a
$
b
m
n
s
$
a
n
a
s
$
a
b
a
n
a
s
$
a
m
n
s
$
a
b
a
n
a
s
$
a
m
n
s
$
a
b
a
n
a
s
$
a
m
n
s
$
a
b
a
n
a
s
$

banana
Root
a
b
m
n
s
$
p
a
n
a
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
a
s
$
b
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
s
$

banana
Root
a
b
m
n
s
$
p
a
n
a
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
s
$
b
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$

banana
Root
a
b
m
n
s
$
p
a
n
a
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
s
$
b
a
n
a
s
$
m
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
s
$

banana
Root
a
b
m
n
s
$
p
a
n
a
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
s
$
b
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
m
a
b
a
n
a
n
a
s
$
s
$

banana

Root
a
b
m
n
s
$
p
a
n
a
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
n
a
n
a
s
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$

Root
nab

Root
nab

Root
nab

antenna
Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
m
n
s
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$

antenna
Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
m
a
b
a
n
a
n
a
s
$

antenna
Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
s
$
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$

Where Are the Matches???

b a n a n a s $

Where Are the Matches???

p a n a m a **b a n a n a s $**

Idea: to find the positions of matches, add some information to leaves

But there is one important question we forgot to answer. Where are the matches? How do we find our patterns match the text? There's no information in the suffix trie yet that allows us to answer this question. Here's an idea. Let's try to add some information to the leaves of our tree. But what information to add? Let's say for every leaf,let's add information about the starting position of the suffix,that generated this leaf. For example, for bananas,we will let position six, because bananas, start at position six of the text.

Root
a
b
m
n
s
$
b
m
p
a
n
s
a
n
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$

panamabananas$
Root
0

panamabananas$
Root
1
0

Root
p a n a m a b a n a n a s $
a
b
m
n
s
$
p
a
n
a
n
a
s
$
1
2
0

panamabananas$
Root
3
1
2
0

panamabananas$
Root
a
b
m
n
s
$
p
3
1
4
2
0

panamabananas$

panama**bananas$**

panamabananas$
Root
a
b
m
n
s
$
p
5
3
1
7
6
4
2
0

panamabananas$
Root
a
b
m
n
s
$
p
5
3
1
7
6
4
2
8
0

panamabanas$
Root
a
b
m
n
s
$
p
a
n
a
m
a
b
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
m
n
s
a
s
$
7
$
9
a
n
a
n
a
s
$
6
a
b
a
n
a
n
a
s
$
4
m
n
s
a
b
a
n
a
n
a
s
$
2
a
s
$
8
$
a
b
a
n
a
n
a
s
$
0

p a n a m a b a n a n a s $
Root
a
b
m
n
s
$
p
s
5
3
1
7
9
6
4
2
0
10
8

Root
p a n a m a b a n a n a s $
a
b
m
n
s
$
p
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
m
n
s
a
s
$
7
$
9
11
b
m
a
n
a
n
a
s
$
6
a
b
a
n
a
n
a
s
$
4
m
n
s
a
b
a
n
a
n
a
s
$
2
a
s
$
8
$
10
a
m
a
b
a
n
a
n
a
s
$
0

p a n a m a b a n a n a s $
Root
a
b
m
n
s
$
p
a
n
a
m
n
s
$
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
a
s
$
7
$
9
11
b
m
a
n
a
n
a
s
$
6
a
b
a
n
a
n
a
s
$
4
a
b
a
n
a
n
a
s
$
2
a
s
$
8
$
10
a
b
a
n
a
n
a
s
$
0
12

Root
SuffixTrie(*Text*)

Walking Down to the Leaves to Find Matches

- Once we find a match, we “walk down” to the leaf (or leaves) in order to find the starting position of the match.

banana
Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
a
n
a
s
$
7
$
9
11
a
n
a
n
a
s
$
6
a
b
a
n
a
n
a
s
$
4
a
b
a
n
a
n
a
s
$
2
a
s
$
10
$
8
a
b
a
n
a
n
a
s
$
0
$
12

banana

Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
a
s
$
7
$
9
11
a
n
a
n
a
s
$
D
6
a
b
a
n
a
n
a
s
$
4
a
b
a
n
a
n
a
s
$
2
a
s
$
10
$
8
a
b
a
n
a
n
a
s
$
0
12

banana

Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
m
n
s
a
s
$
7
$
9
11
b
m
a
n
a
n
a
s
D
$
6
n
a
b
a
n
a
n
a
s
$
4
a
m
n
s
a
b
a
n
a
n
a
s
$
2
a
s
$
10
$
8
12
a
m
a
b
a
n
a
n
a
s
$
0

panamabanana s$

a n a

"ana" occurs three times but to find its starting position we have to walk down to the end of each branch

ana
Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
a
s
$
7
$
9
11
a
n
a
n
a
s
$
6
a
b
a
n
a
n
a
s
$
4
a
b
a
n
a
n
a
s
$
2
a
s
$
8
$
10
12
a
b
a
n
a
n
a
s
$
0

ana
Root
a
b
m
n
s
$
p
a
n
a
n
a
s
$
5
a
b
a
n
a
n
a
s
$
3
a
b
a
n
a
n
a
s
$
1
m
n
s
a
s
$
7
$
9
11
a
n
a
n
a
s
$
6
a
b
a
n
a
n
a
s
$
4
m
n
s
a
b
a
n
a
n
a
s
$
2
a
s
$
8
$
10
12
a
b
a
n
a
n
a
s
$
0

ana
Root
a
b
m
n
s
$
p
a
n
a
n
a
$
12
11
9
7
10
8
5
3
1
6
4
2
0

p a n a m a b a n a n a s $
Root
"ana" occurs three times but to find its starting position we have to walk down to the end of each branch

Memory Footprint of Suffix Trie

The suffix trie is formed from |*Text*| suffixes with total length:

|*Text*| symbols

```
p a n a m a b a n a n a s $
  a n a m a b a n a n a s $
    n a m a b a n a n a s $
      a m a b a n a n a s $
        m a b a n a n a s $
          a b a n a n a s $
            b a n a n a s $
              a n a n a s $
                n a n a s $
                  a n a s $
                    n a s $
                      a s $
                        s $
                          $
```

|*Text*| suffixes

$$|Text| * (|Text| - 1)/2$$

This is O(|Text|^2)

Because the average length of the text suffix is approximately text/2

For human genome:

- $|Text| \approx 3*10^9$

Outline

- From Genome Sequencing to Pattern Matching
- Brute Force Approach to Pattern Matching
- Herding Patterns into Trie
- Herding Text into Suffix Trie
- **From Suffix Tries to Suffix Trees**

Root
a
b
m
n
s
$
b
m
p
a
n
s
a
n
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
s
$
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
s
$
a
m
a
b
a
n
a
n
a
s
$

Root
bananas$
a
m
n
s
$
b
p
a
n
a
$
s
a
m
n
s
a
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
n
s
a
b
a
n
a
n
a
s
$
a
$
a
b
a
n
a
n
a
s
$
To reduce memory we can compress each "nonbranching path" of the tree

Root
a
bananas$
m
n
s
$
b
m
n
p
a
n
s
$
a
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
s
a
$
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
m
n
s
a
$
a
b
a
n
a
n
a
s
$
m
n
s
a
$
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
n
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
$
b
m
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
p
a
n
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
s
$
n
a
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
n
s
$
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
$
b
m
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
p
a
n
a
m
n
s
a
b
a
n
a
n
a
s
$
a
s
$
$
s
n
a
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
n
s
a
$
s
$
b
m
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
p
a
n
a
m
n
s
a
b
a
n
a
n
a
s
$
a
$
s
$
n
a
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
n
s
a
$
s
$
b
m
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
p
a
n
a
m
n
s
a
b
a
n
a
n
a
s
$
a
$
s
$
s
$
a
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
s
$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
p
a
n
a
m
n
s
a
b
a
n
a
n
a
s
$
a
$
s
$
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
s
$
b
a
n
a
n
a
s
$
m
a
a
b
a
n
a
n
a
s
$
p
a
n
a
m
n
s
a
b
a
n
a
n
a
s
$
a
a
$
s
$
n
a
a
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
$$
b
m
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
p
a
n
a
m
n
s
a
b
a
n
a
n
a
s
$
$
n
a
$
s
$
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
n
s
$
a
mabananas$
nas$
$$
b
m
p
a
n
s
$
n
a
m
n
s
$
a
b
a
n
a
n
a
s
$
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
na
s
$
mabananas$
nas$
$$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
a
n
s
a
a
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s
$
mabananas$
nas$
$$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
$$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
a
n
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
$$
b
a
n
a
n
a
s
$
m
a
b
a
n
a
n
a
s
$
n
a
m
a
b
a
n
a
n
a
s
$
a
n
s
a
$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
m
a
b
a
n
a
n
a
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
n
a
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
m
p
a
n
a
m
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
b
a
n
a
n
a
s
$
a
a
s
$
$
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
$
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
Bananas$
mabananas$
n
p
a
n
a
m
a
b
a
n
a
n
a
s
$
n
a
s
$
s
$
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
p
a
n
a
mabananas$
n
s
a
$
s
$
s
n
a
$
m
a
b
a
n
a
n
a
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
n
s
a
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
nas$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
n
a
mabananas$
nas$
s
$
p
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
$$
bananas$
mabananas$
p
a
n
a
mabananas$
nas$
$$
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
$$
bananas$
mabananas$
p
a
n
a
mabananas$
nas$
$$
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
$$
Bananas$
mabananas$
p
na
mabananas$
nas$
$$
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
$$
bananas$
mabananas$
p
na
mabananas$
nas$
$$
a
n
a
m
a
b
a
n
a
n
a
s
$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
na
mabananas$
nas$
s$
panamabananas$
s
$

Root
a
bananas$
mabananas$
na
s$
mabananas$
nas$
s$
bananas$
mabananas$
na
mabananas$
nas$
s$
panamabananas$
s
$

SuffixTree(*Text*)

Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)

Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- **Cheating!!!** - how do we store all edge labels?

Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- **Cheating!!!** - how do we store all edge labels?

Meaning that the memory required to store all strings will still be same as suffix trie as here also we are storing all the strings

Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- storing edge labels

hence we will be storing just lengths and starting positions as labels

Since each suffix adds one leaf and at most one internal vertex to the suffix tree:

- # vertices < 2|*Text*|
- memory footprint of the suffix tree: O(|*Text*|)
- storing edge labels

This is not true (IR). Just given as an example of how tree is constructed where memory footprint is O(|Text|) because we still need sub-strings in the tree to compare it to pattern

Why did we bother to add "**$**" to "panamabananas"?

- to make sure that each suffix corresponds to a leaf

Why do we want to make sure that each suffix correspond to a leaf?

- construct suffix tree for "papa"(without adding "$") and compare it with the suffix tree for "papa$"

Constructing Suffix Tree: **Naive Approach**

because with naive algo we have to first construct suffix trie and then we would construct suffix tree

- **Quadratic** runtime:
  - $O(|Text|^2)$

$O(|Genome| + |Patterns|)$ to find pattern matches

Constructing Suffix Tree: **Linear-Time Algorithm**

- **Linear** runtime (for a constant-size alphabet):
  - O(|*Text*|)

Too complex to cover in this course

Linear-time algorithm (Weiner, 1973) was simplified by Ukkonen (1995) but it is still too complex to cover in this course

Big Secret of the Big O Notation

- Suffix trees enable fast **Exact** Multiple Pattern Matching: First we need O(|Text|) to build the suffix tree. Then for each pattern Pattern in Patterns we need additional O (|Pattern|) to match this pattern against the Text. The total time for all the patterns is thus O(|Patterns|), and the overall running time is O(|Text| + |Patterns|). We only need O(|Text|) additional memory to store the suffix tree and all the positions where at least one of the Patterns occurs in the Text.
  - Runtime: O(|*Text*| + |*Patterns*|)
    TO Construct Suffix Tree
  - Memory: O(|*Text*|)
- However, big-O notation hides constants!
  - suffix tree algorithms has large memory footprint ~ 20•|*Text*| for long texts like human genome
- We want to find mutations!
  - it is unclear how to develop fast **Approximate** Multiple Pattern Matching using suffix trees