Suffix Arrays: basic queries
Ben Langmead

Please sign guestbook (www.langmead-lab.org/teaching-materials) to tell me briefly how you are using the slides. For original Keynote files, email me (ben.langmead@gmail.com).
Suffix array: querying

Suffix array for $T = abracadabra\text{d}ad$
How to check if a query string $P$ a substring of $T$?

1. For $P$ to be a substring, it must be a **prefix** of $\geq 1$ of $T$'s **suffixes**
2. Suffixes sharing a prefix are **consecutive** in the suffix array
Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end.

If characters are equal, we advance to the right in both.
If characters are not equal, we have our answer.

A: abracadabra
   
B: abradad
Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end.

If characters are equal, we advance to the right in both. If characters are not equal, we have our answer.

A: abracadabradad

= - - - - - -

B: abradad
Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end.

If characters are **equal**, we advance to the right in both. If characters are **not equal**, we have our answer.

A: \texttt{abracadabra\texttt{d d a d d}}

\texttt{== - - - - - - -}

B: \texttt{abradadad}
Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end.

If characters are equal, we advance to the right in both. If characters are not equal, we have our answer.

A: abracadabra dads
   ===-=-

B: abradadad
Suffix array: lexicographical comparisons

When comparing two strings, we start at the left end.

If characters are equal, we advance to the right in both. If characters are not equal, we have our answer.

A: abracadabradadad
   ===< --
B: abradadad

Therefore A < B
Say query is **dad**  Which suffixes (if any) have **dad** as a prefix? Let's apply binary search.
Suffix array: querying

Iteration 1: compare dad to pivot bracadabradad$

dad is alphabetically after; recurse on right half
Suffix array: querying

SA:

$ 14 0 7 3 12 5 10 1 8 4 13 6 11 2 9
Suffix array: querying

SA:

14 0 7 3 12 5 10 1 8 4 13 6 11 2 9
Suffix array: querying

Iteration 2: compare dad to pivot dabradad$

dad is alphabetically after; recurse on bottom half
Suffix array: querying
Suffix array: querying

SA: 14 0 7 3 12 5 10 1 8 4 13 6 11 2 9
Suffix array: querying

Iteration 3: compare \textbf{dad} to pivot \textbf{racadabradad$}
\textbf{dad} is before
Suffix array: querying

SA: 14 0 7 3 12 5 10 1 8 4 13 6 11 2 9
Suffix array: querying

Iteration 4: compare dad to pivot dad$

Answer: yes, dad appears
Suffix array: querying

Usually binary search is \( O(\log m) \)...

\( O(\log m) \) bisections, \( O(1) \) per bisection to compare

But we need \( O(n) \) to lexicographically compare the pivot suffix with or length-\( n \) query. So \( O(n \log m) \) time overall.
Suffix array: querying

SA:

14 0 7 3 12 5 10 1 8 4 13 6 11 2 9
Suffix array: querying
Suffix array: querying

SA:

<table>
<thead>
<tr>
<th></th>
<th>14</th>
<th>0</th>
<th>7</th>
<th>3</th>
<th>12</th>
<th>5</th>
<th>10</th>
<th>1</th>
<th>8</th>
<th>4</th>
<th>13</th>
<th>6</th>
<th>11</th>
<th>2</th>
<th>9</th>
</tr>
</thead>
</table>

$ | abracadabradad$ | abradad$ | acadabradad$ | ad$ | adabradad$ | adad$ | bracadabradad$ | bradad$ | cadabradad$ | d$ | dabradad$ | dad$ | radad$ | racadabradad$ | radad$ | d$ | dabradad$ | dad$ | radad$ | racadabradad$ | radad$ | d$ | dabradad$ | dad$ | radad$ | racadabradad$ |

Diagram showing the suffix array and the corresponding suffixes.
Suffix array
Suffix array

SA: 14 0 7 3 12 5 10 1 8 4 13 6 11 2 9
Suffix array

Suffix tree is an **explicit** tree

SA binary search tree is **implicit**
...but has same function: narrow the space of suffixes that might have our query as a prefix
Suffix array

Presence/absence is $O(n)$ time for length-$n$ query

Presence/absence is $O(n \log m)$ time for length-$n$ query