Suffix arrays

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

$T = \text{abaaba}\$

$SA(T) = \begin{array}{c|c}
0 & 6 \\
1 & 5 \\
2 & a $ \\
3 & a a b a $ \\
4 & a b a $ \\
5 & b a $ \\
6 & b a a b a $
\end{array}$

Suffix array of $T$ is an array of integers in $[0, m)$ specifying lexicographic (alphabetical) order of $T$'s suffixes
Suffix array

$O(m)$ space, like suffix tree

Is “constant factor” worse, better, same?

![Diagram showing peak memory usage and time for suffix tree and suffix array algorithms.](chart.png)
Suffix array

32-bit integers sufficient for human genome, so fits in
~4 bytes/base × 3 billion bases ≈ 12 GB. Suffix tree is >45 GB.
Suffix array: querying

Is $P$ 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

Use binary search

$T = \text{abaaba}$

<table>
<thead>
<tr>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$</td>
<td>a $</td>
<td>b a $</td>
<td>a b a $</td>
<td>a a b a $</td>
<td>b a a b a</td>
</tr>
</tbody>
</table>

$P = \text{aba}$
Suffix array: querying

Is \( P \) a substring of \( T \)?

Do binary search, check whether \( P \) is a prefix of the suffix there

Query time is \( O(\ ? \ ) \ldots \)

\( O(\log_2 m) \) bisections, \( O(n) \) comparisons per bisection, so \( O(n \log m) \)

\( T = \text{abaaba}\$ \)

<table>
<thead>
<tr>
<th>6</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>a $</td>
</tr>
<tr>
<td>2</td>
<td>a a b a a $</td>
</tr>
<tr>
<td>3</td>
<td>a b a $</td>
</tr>
<tr>
<td>0</td>
<td>a b a a b a $</td>
</tr>
<tr>
<td>4</td>
<td>b a $</td>
</tr>
<tr>
<td>1</td>
<td>b a a b a $</td>
</tr>
</tbody>
</table>
Suffix array: querying

Contrast suffix array query time: $O(n \log m)$ with suffix tree: $O(n)$

Time can be improved to $O(n + \log m)$, but we won’t discuss here (See Gusfield 7.17.4). For this class, we'll consider it $O(n \log m)$. 
Suffix array: sorting suffixes

Use your favorite sort, e.g., quicksort

|   | a b a a b a $ | def quicksort(q):
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>b a a b a $</td>
<td>lt, gt = [], []</td>
</tr>
<tr>
<td>1</td>
<td>a a b a $</td>
<td>if len(q) &lt;= 1:</td>
</tr>
<tr>
<td>2</td>
<td>a b a $</td>
<td>return q</td>
</tr>
<tr>
<td>3</td>
<td>b a $</td>
<td>for x in q[1:]:</td>
</tr>
<tr>
<td>4</td>
<td>a $</td>
<td>if x &lt; q[0]:</td>
</tr>
<tr>
<td>5</td>
<td>$</td>
<td>lt.append(x)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>else:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gt.append(x)</td>
</tr>
</tbody>
</table>

Expected time: $O(m^2 \log m)$

Not $O(m \log m)$ because a suffix comparison is $O(m)$ time
Suffix array: building

How to build a suffix array?

(a) Build suffix tree, (b) traverse in alphabetical order, (c) upon reaching leaf, append suffix to array
Suffix array: sorting suffixes

Another idea: Use a sort algorithm that’s aware that the items being sorted are all suffixes of the same string

Original suffix array paper suggested an $O(m \log m)$ algorithm


Other popular $O(m \log m)$ algorithms have been suggested


More recently $O(m)$ algorithms have been demonstrated!


Suffix array: summary

Just $m$ integers, with $O(n \log m)$ query time

Constant factor greatly reduced compared to suffix tree: human genome index fits in ~12 GB instead of > 45 GB