Suffix Arrays: building
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).
Building the suffix array is fundamentally about **sorting** the suffixes.
Suffix array: sorting suffixes

Should we use our favorite sort, e.g., **quicksort**?

<table>
<thead>
<tr>
<th>0</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a$</td>
</tr>
<tr>
<td>2</td>
<td>aaba$</td>
</tr>
<tr>
<td>3</td>
<td>aba$</td>
</tr>
<tr>
<td>4</td>
<td>abaaba$</td>
</tr>
<tr>
<td>5</td>
<td>ba$</td>
</tr>
<tr>
<td>6</td>
<td>baaba$</td>
</tr>
</tbody>
</table>

```python
def quicksort(q):
    lt, gt = [], []
    if len(q) <= 1:
        return q
    for x in q[1:]:
        if x < q[0]:
            lt.append(x)
        else:
            gt.append(x)
    return quicksort(lt) + q[0:1] + quicksort(gt)
```

We learn this is \( O(m \log m) \) expected time.

Quicksort is \( O(m \log m) \) (expected) when sorting items that can be compared in constant time.

But our **lexicographic** comparisons are \( O(m) \) time!

So \( O(m^2 \log m) \) overall 😞
Can a suffix **tree** help us build a suffix **array**?

Recall the leaves of the tree are also the elements of the array. It's a matter of visiting in the right order.
Suffix array: sorting suffixes

Try: depth-first traversal, always visiting children in lexicographical order
Suffix array: sorting suffixes

Try: depth-first traversal, always visiting children in lexicographical order
Suffix array: sorting suffixes

Suffix tree & array are both $O(m)$ space
Suffix tree construction with Ukkonen is $O(m)$ time
Traversal is $O(m)$ time
So this is $O(m)$ time & space✅
Suffix array: sorting suffixes

To avoid overhead of building a tree first, we can choose a "direct" algorithm

Fast-in-practice $O(m \log m)$ algorithms


Newer $O(m)$ algorithms as well!

