Intro: Indexing

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).
Imagine we have recorded the ages of many people; say, voters:

How many voters are aged 27?
To find out, we have no choice but to scan $n = 1\text{M}$ records

*Order* to the rescue

---

Suppose instead our list is ordered by age

How many voters are aged 27?

Binary search

More specifically?

2 searches, one for the first age-27 person, one for last
Indexing

Simply *ordering* the data allows us to query it more efficiently

From $n$-item scan to two $\log_2 n$ binary searches

Did it also improve our ability to *compress* the age data?

Yes; we now have "runs" of same value, monotonicity, etc
Indexing

Grouping

Ordering
Indexing

We are working with a text. We want to know if some word occurs. The text is big but an excerpt is:

```
... order is good ...
```

Ordering words alphabetically: good < is < order
Can we still use binary search?

Yes, but what's the cost of comparing 2 words?

Several character comparisons needed to get relative order of dinosaur & dinosaurs

Again, we've improved queryability & compressibility
Indexing

Queries only on *words* is limiting

Texts might not consist of words e.g. DNA

Word matches might not be the right query e.g. autocomplete e.g. inexact matching

What if we'd like to be able to query *any substring*?
Indexing

... order _ is _ good ...

Use underscore (_) for space, assume it comes first alphabetically

Put all suffixes in order...
Indexing

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_good...</td>
<td>(This is just the relative order of the order_is_good suffixes)</td>
</tr>
<tr>
<td>_is_good...</td>
<td></td>
</tr>
<tr>
<td>d...</td>
<td></td>
</tr>
<tr>
<td>der_is_good...</td>
<td>Can we use binary search?</td>
</tr>
<tr>
<td>er_is_good...</td>
<td>Yes; still might need several character comparisons to get relative order of suffixes</td>
</tr>
<tr>
<td>good...</td>
<td></td>
</tr>
<tr>
<td>is_good...</td>
<td></td>
</tr>
<tr>
<td>od...</td>
<td></td>
</tr>
<tr>
<td>ood...</td>
<td></td>
</tr>
<tr>
<td>ood...</td>
<td></td>
</tr>
<tr>
<td>order_is_good...</td>
<td></td>
</tr>
<tr>
<td>r_is_good...</td>
<td></td>
</tr>
<tr>
<td>rder_is_good...</td>
<td></td>
</tr>
<tr>
<td>s_good...</td>
<td></td>
</tr>
</tbody>
</table>
Motivating questions

How do we measure the amount of redundant information in a string?

How do we represent strings so that redundant information takes minimal space?

How can orderings "reveal" structure and make strings compressible?

How can ordering make strings fast to search, faster than binary search?