Intro: Indexing

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Department of Computer Science

Indexing

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

Index	Age
1	78
2	19
3	20
4	50
⋮	
500,000	54
500,001	50
500,002	19
500,003	77
⋮	
999,997	40
999,998	27
999,999	71
1,000,000	44

Example modeled on: Prezza, Nicola. Compressed Computation for Text Indexing. Diss. PhD thesis, University of Udine, 2016.

Indexing

17	18
33	18
39	18
60	18
⋮	
999,905	49
999,985	49
4	50
18	50
⋮	
999,649	101
999,811	101
433,034	103
377,003	104

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

17	18
33	18
39	18
60	18
⋮	
999,905	49
999,985	49
4	50
18	50
⋮	
999,649	101
999,811	101
433,034	103
377,003	104

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

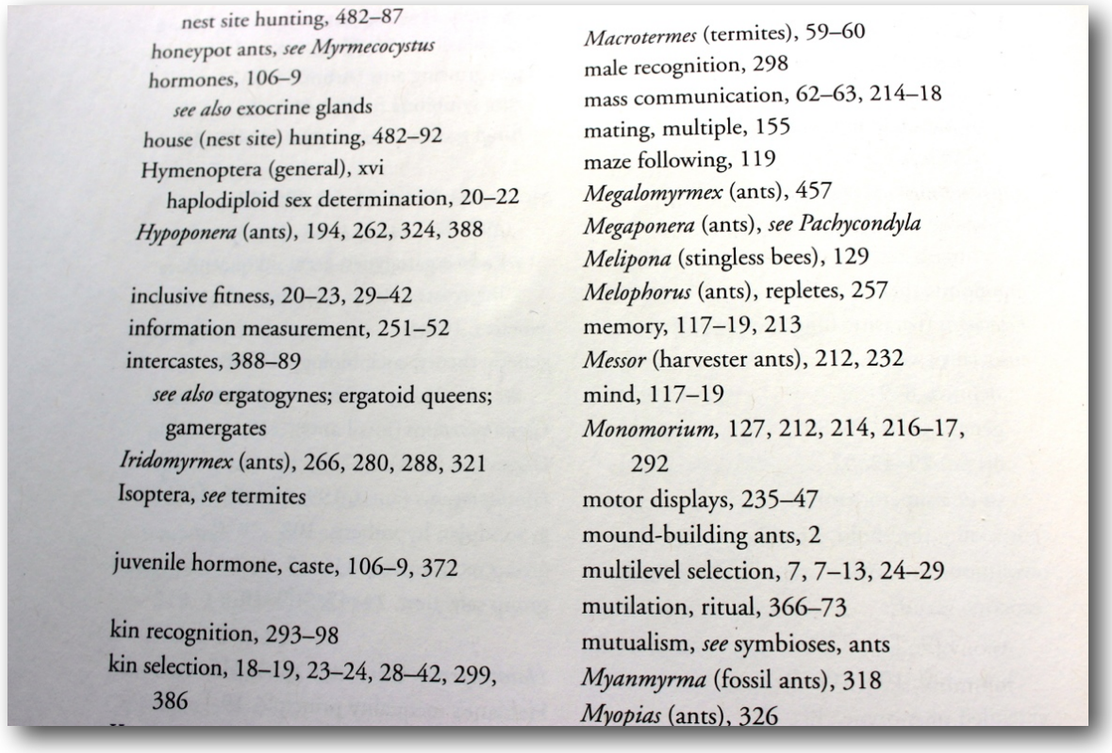
From n-item scan to two $\log_2 n$ binary searches

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

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

Indexing

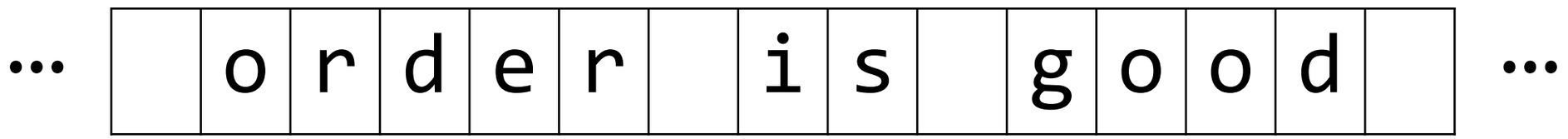
Grouping



Ordering

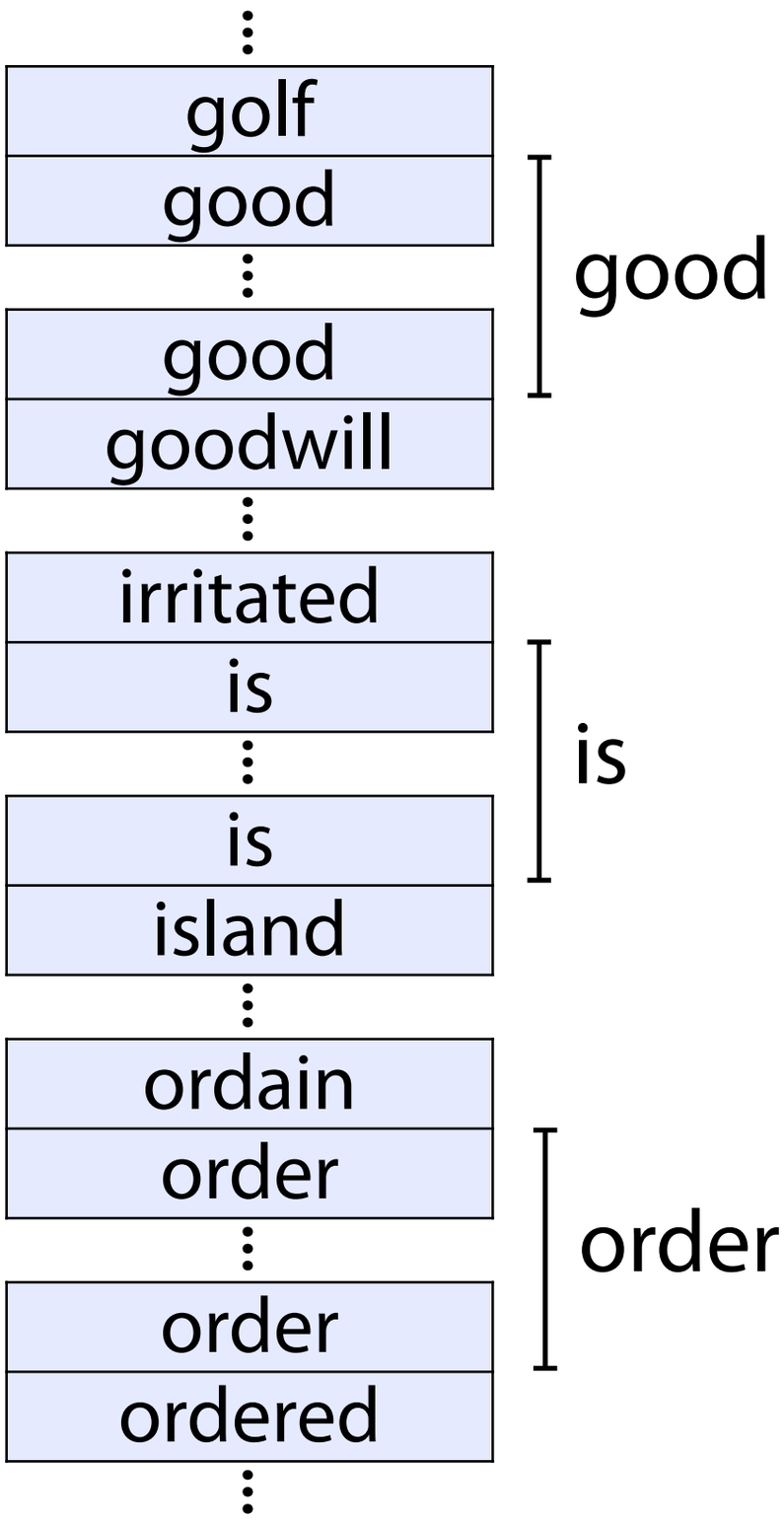
Indexing

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



Ordering words alphabetically: good < is < order

Indexing



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

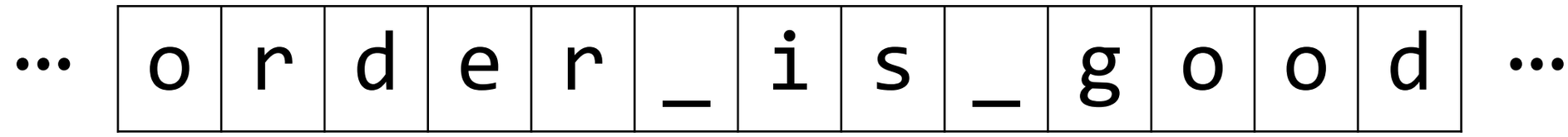
Word matches might not be the right query

e.g. autocomplete

e.g. inexact matching

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

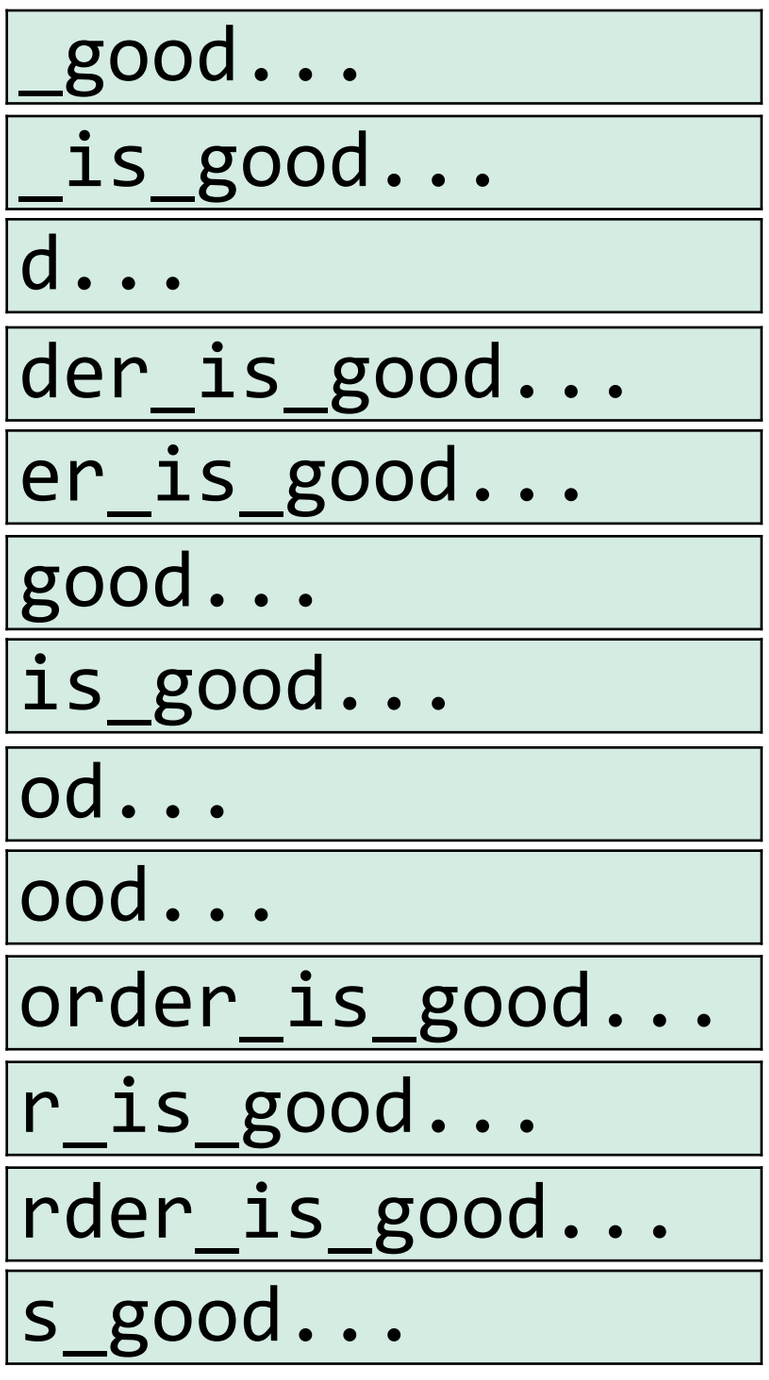
Indexing



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

Put all suffixes in order...

Indexing



(This is just the relative order of the `order_is_good` suffixes)

Can we use binary search?

Yes; still might need several character comparisons to get relative order of suffixes

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?