Running slow queries with Apache Lucene

Adrien Grand
@jpountz
Berlin Buzzwords 2017

Working with Lucene since 2010
Lucene committer since 2012
Lucene PMC since 2013
Elastic employee since 2013

Fast queries are cool

- SIMD instructions for faster decoding of postings
  - Ivan Mamontov & Mikhail Khludnev
    - https://berlinbuzzwords.de/15/session/fast-decompression-lucene-codec
- MaxScore: faster disjunctions
  - Stefan Pohl
    - http://2012.berlinbuzzwords.de/sessions/efficient-scoring-lucene

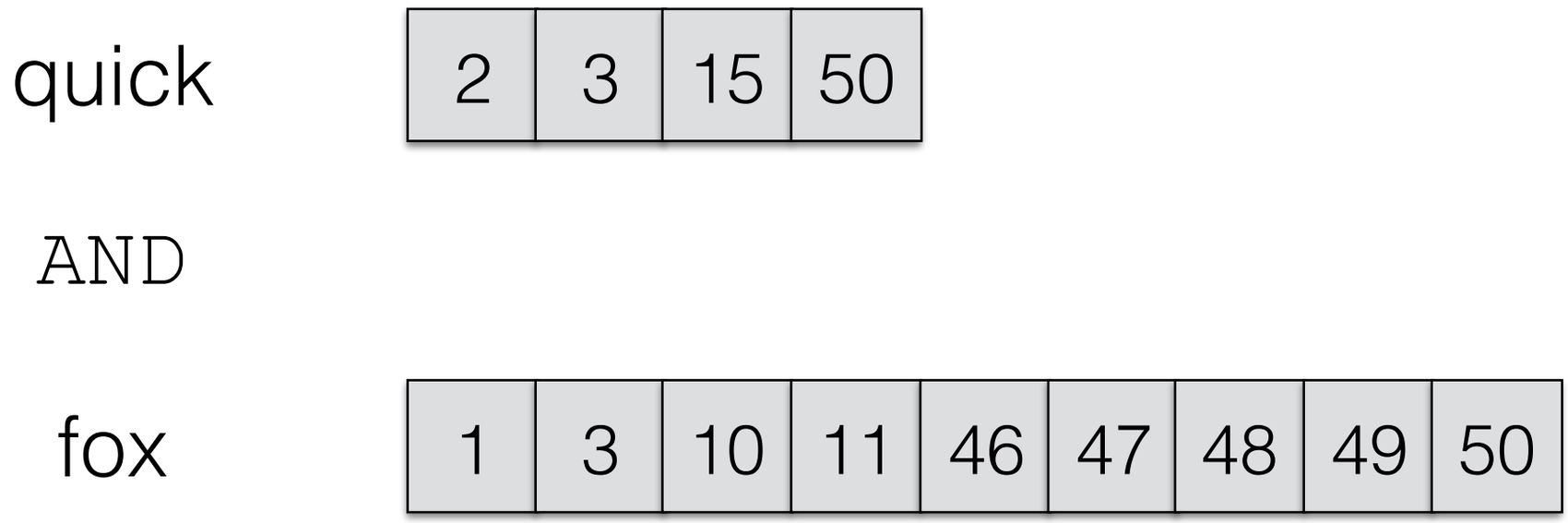
Query responsibilities

- Queries produce sorted iterators over doc IDs
- A fast query can efficiently:
  - iterate over matches
  - skip over arbitrary ranges of doc IDs

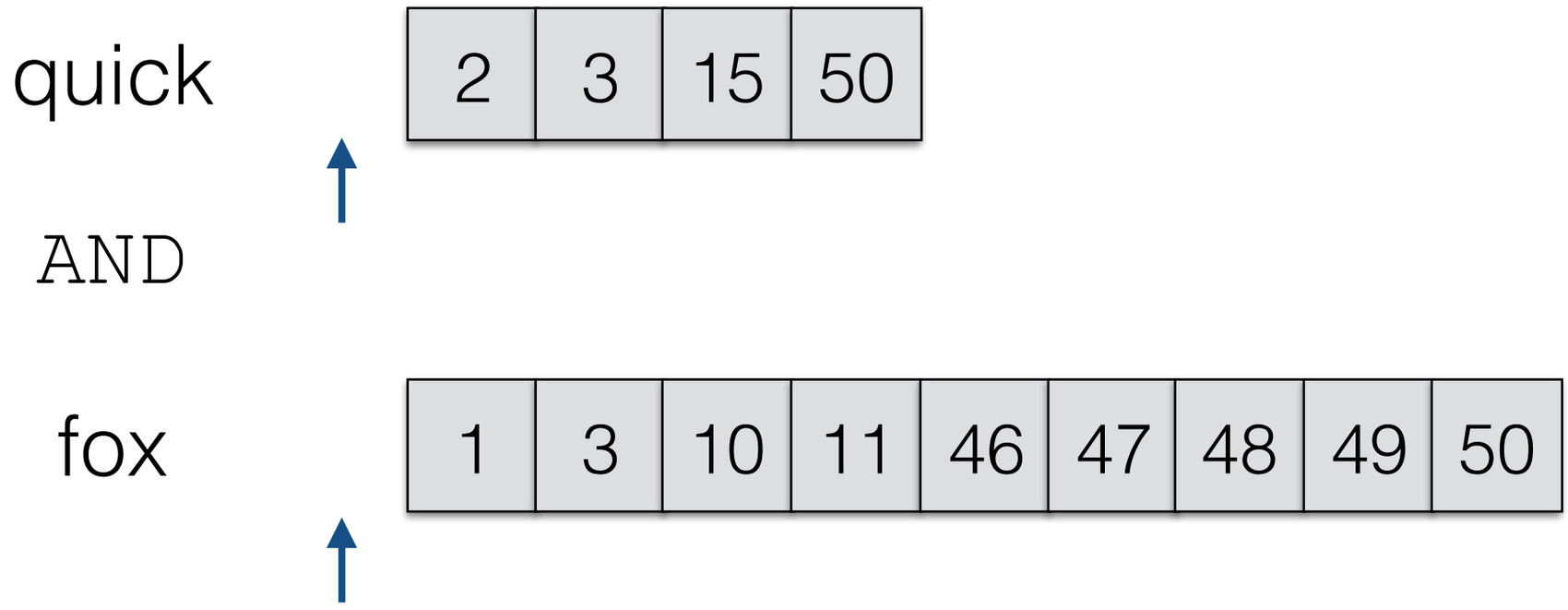
Slow queries

- Can’t iterate efficiently
  - Phrase queries
  - Script queries
- Can’t skip efficiently
  - Numeric range queries
  - Geo bounding box queries
  - Geo distance queries

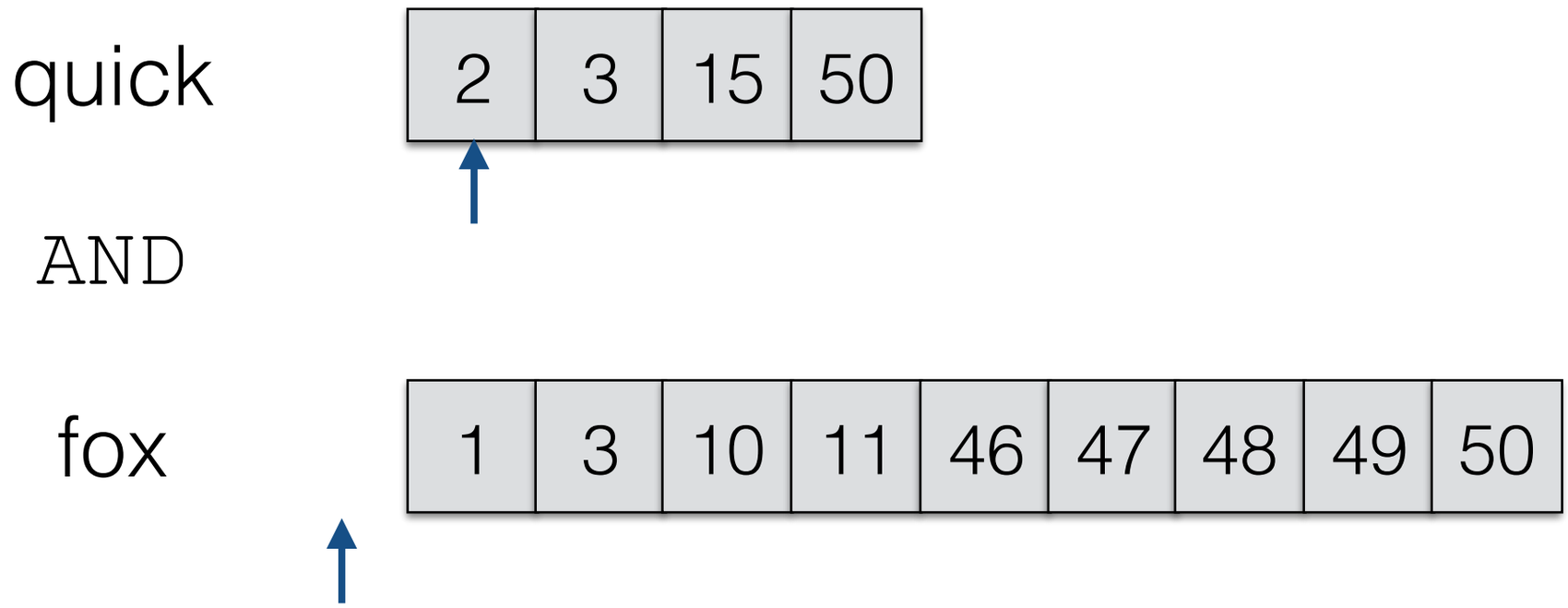
Conjunctions



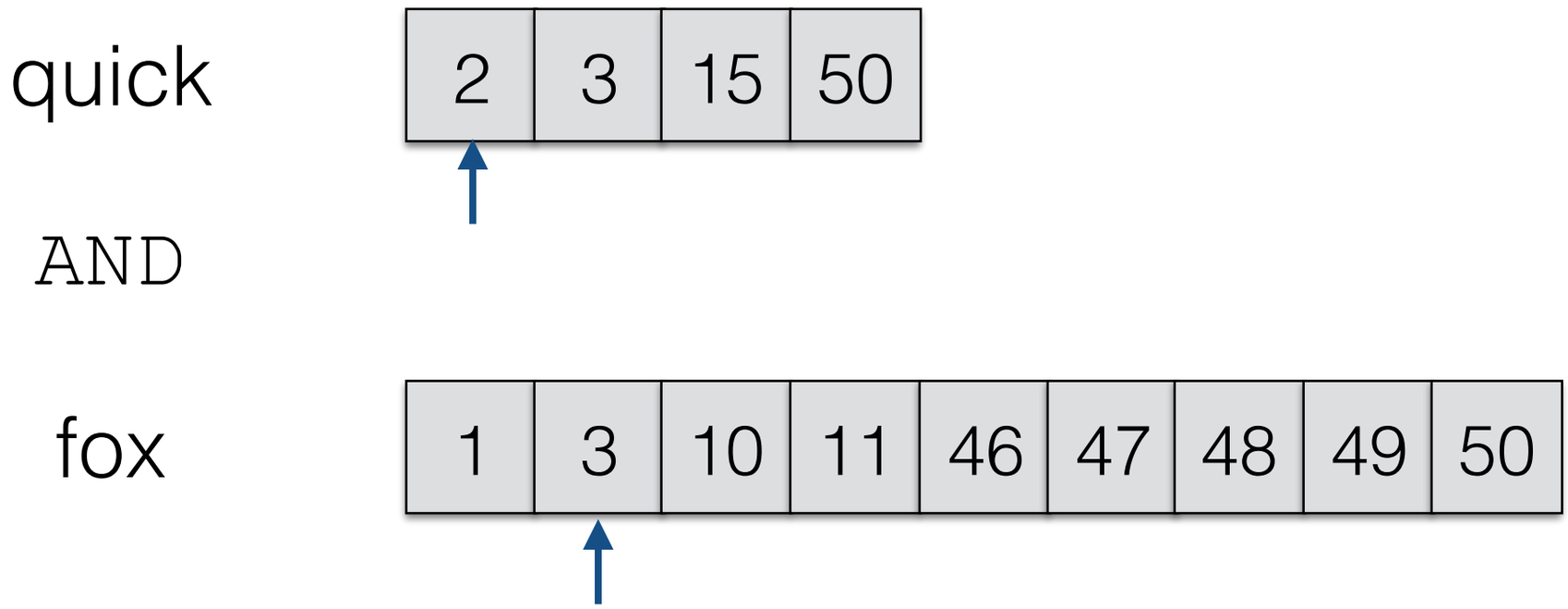
Conjunctions



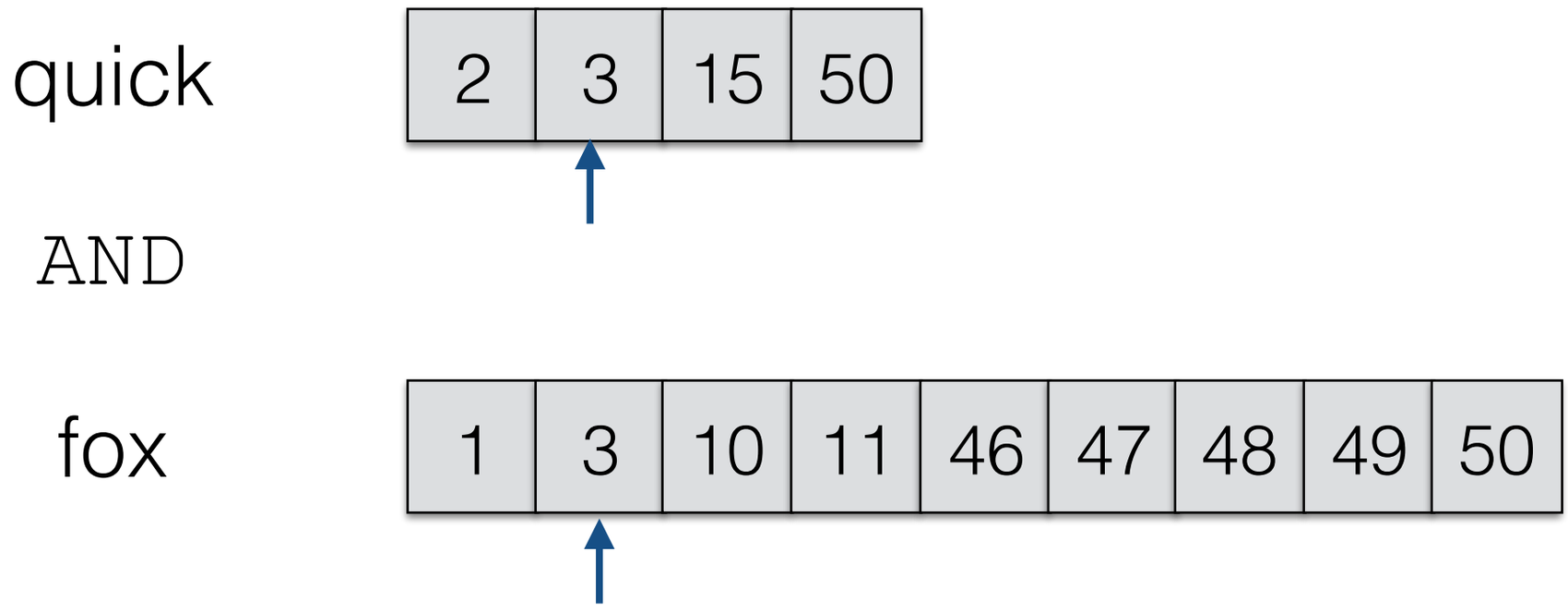
Conjunctions



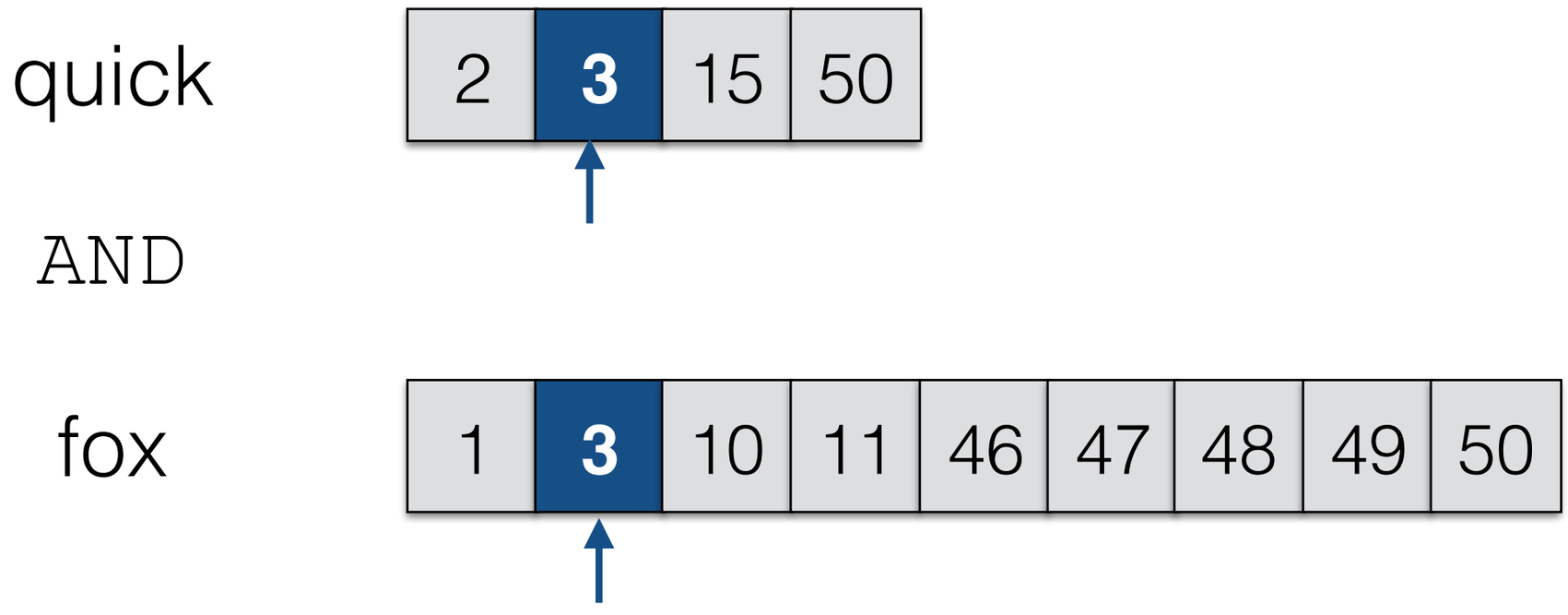
Conjunctions



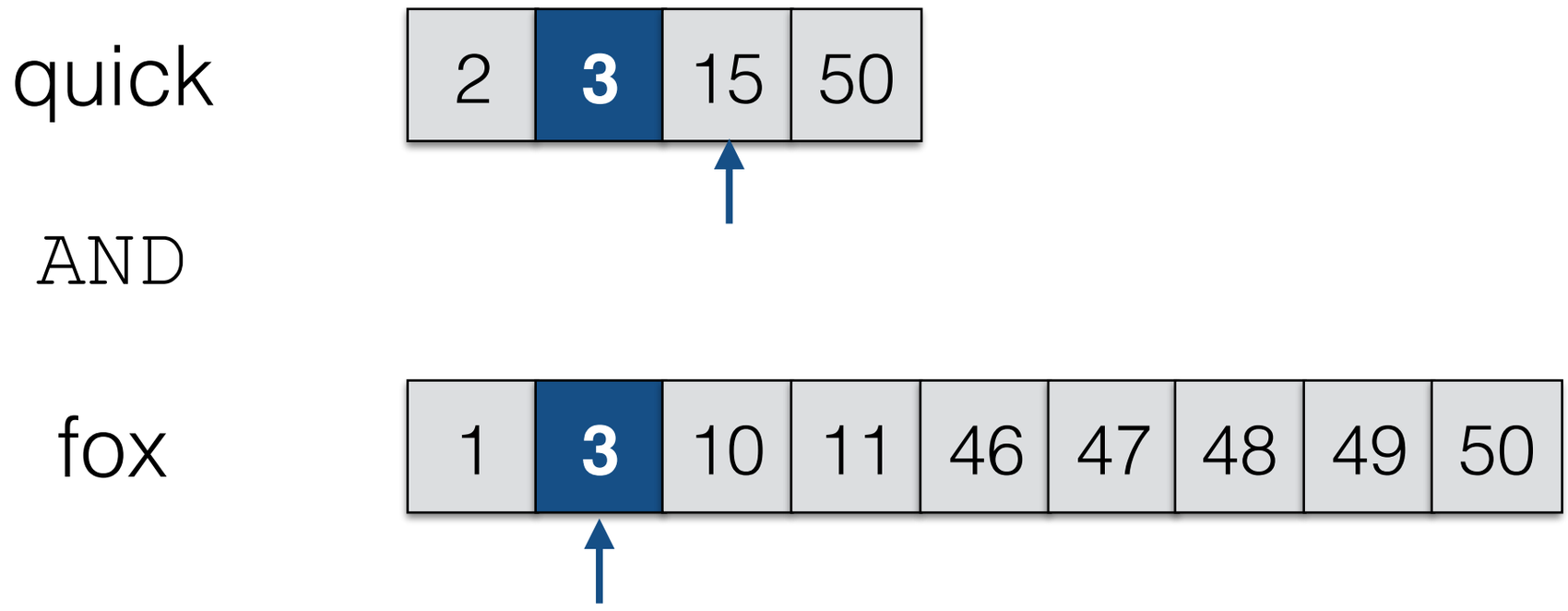
Conjunctions



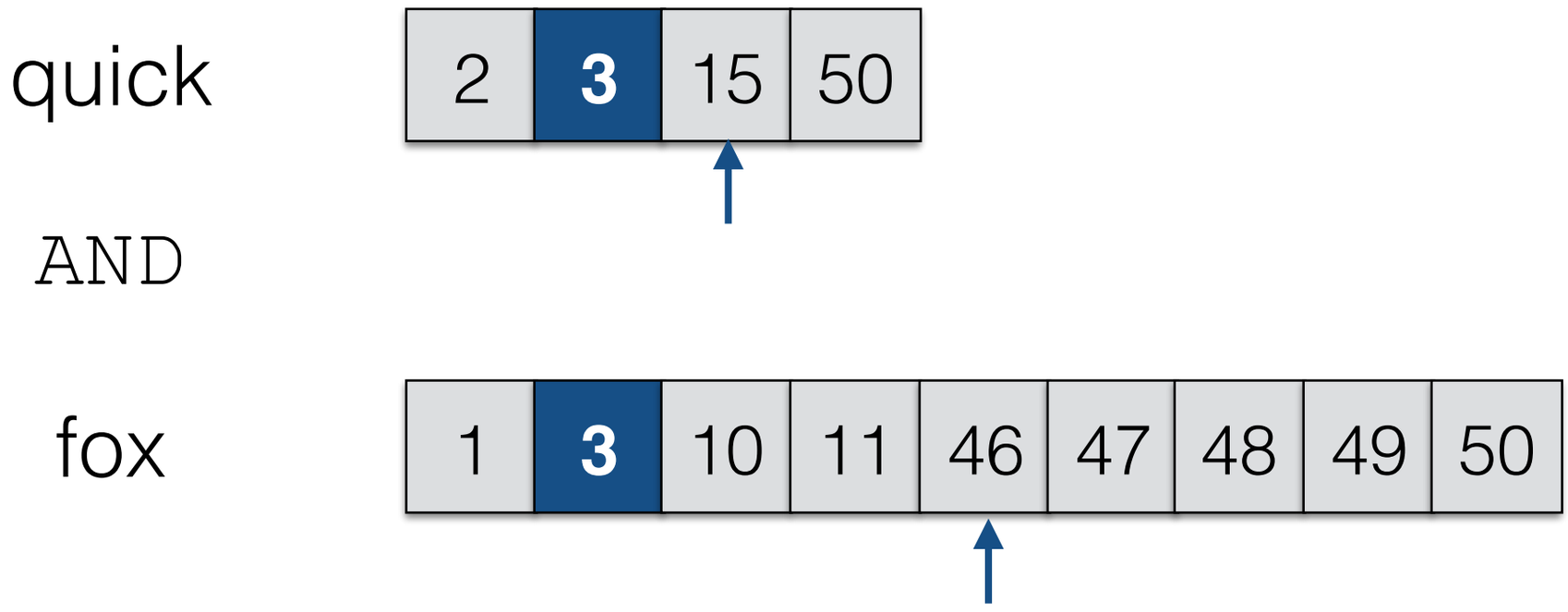
Conjunctions



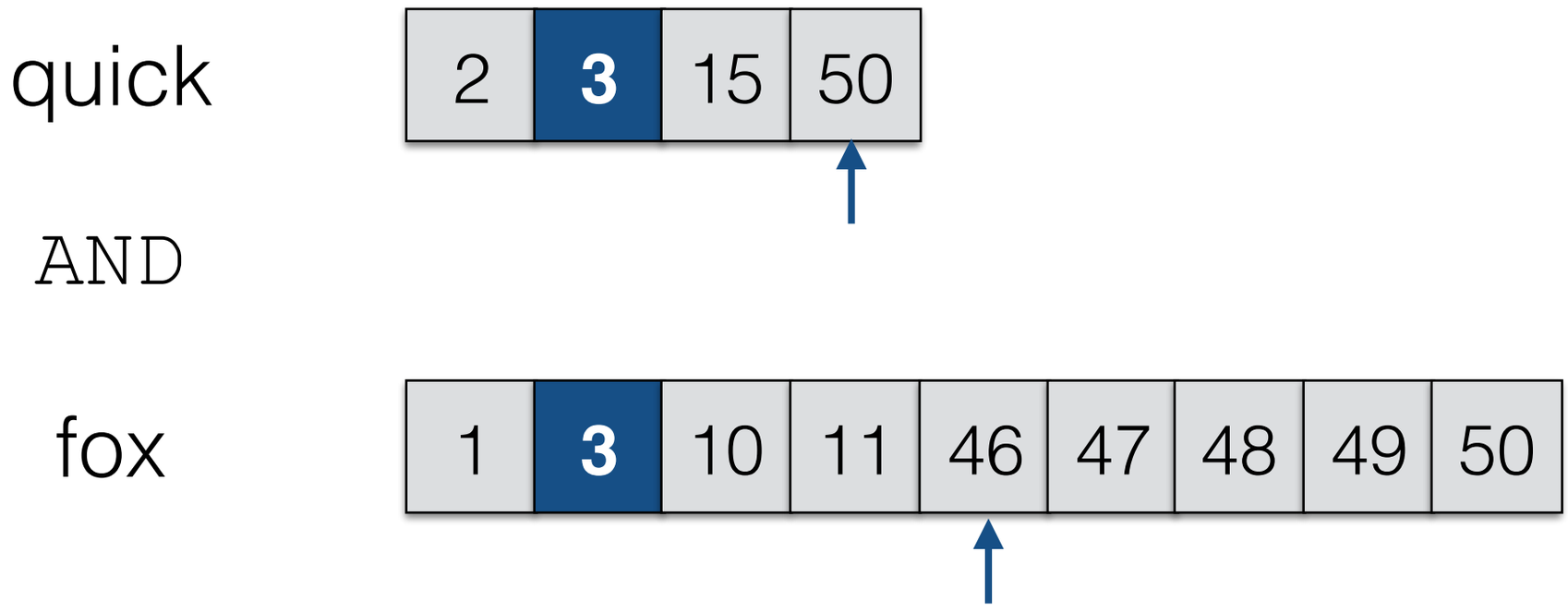
Conjunctions



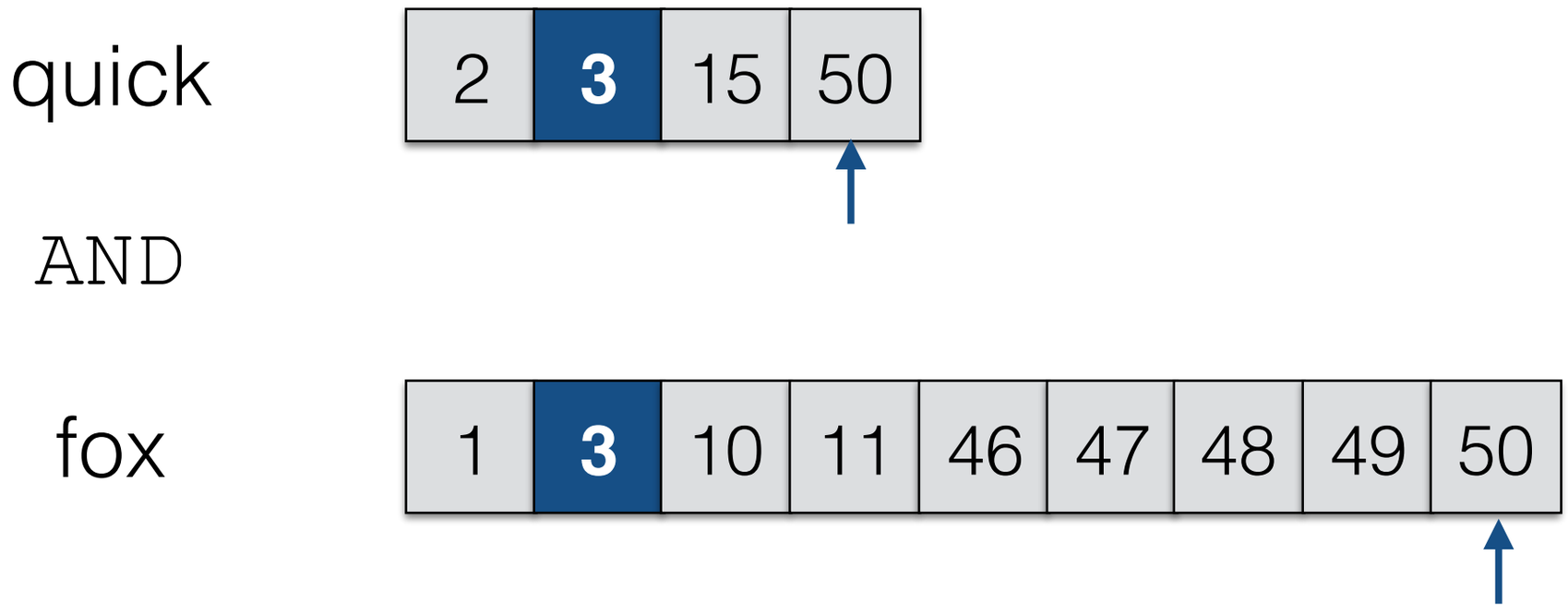
Conjunctions



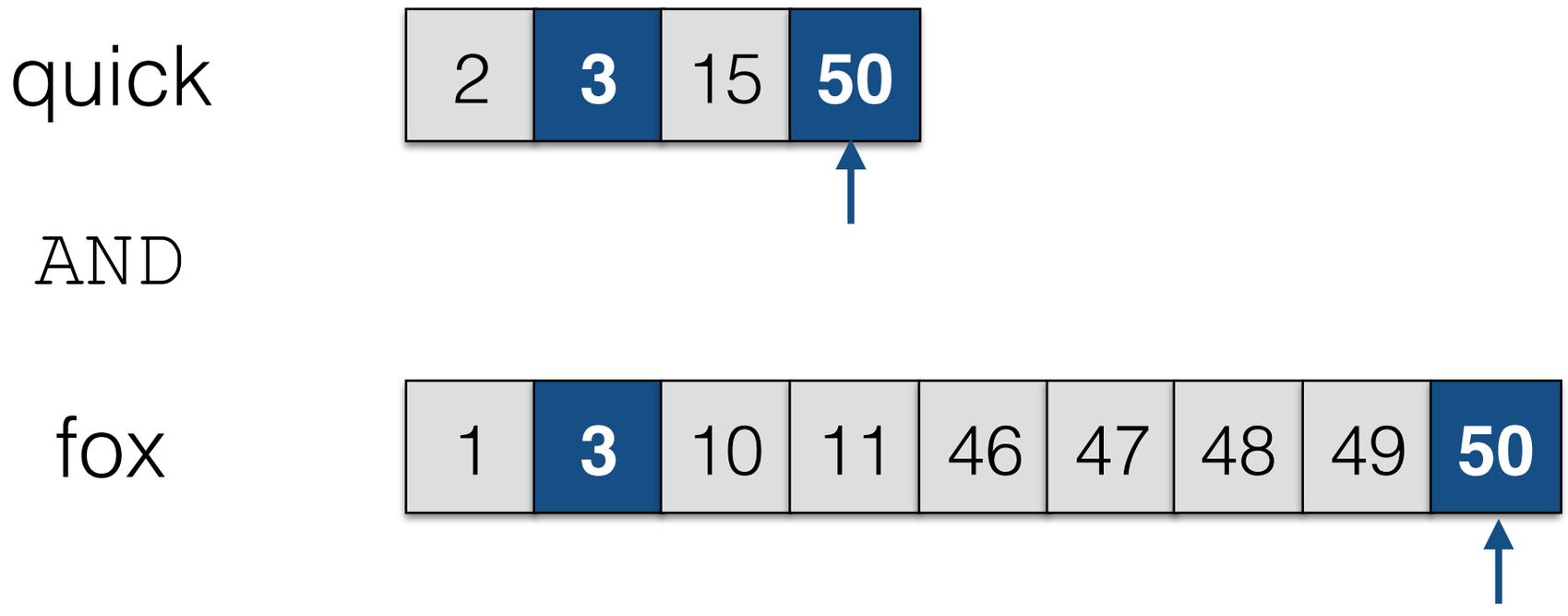
Conjunctions



Conjunctions

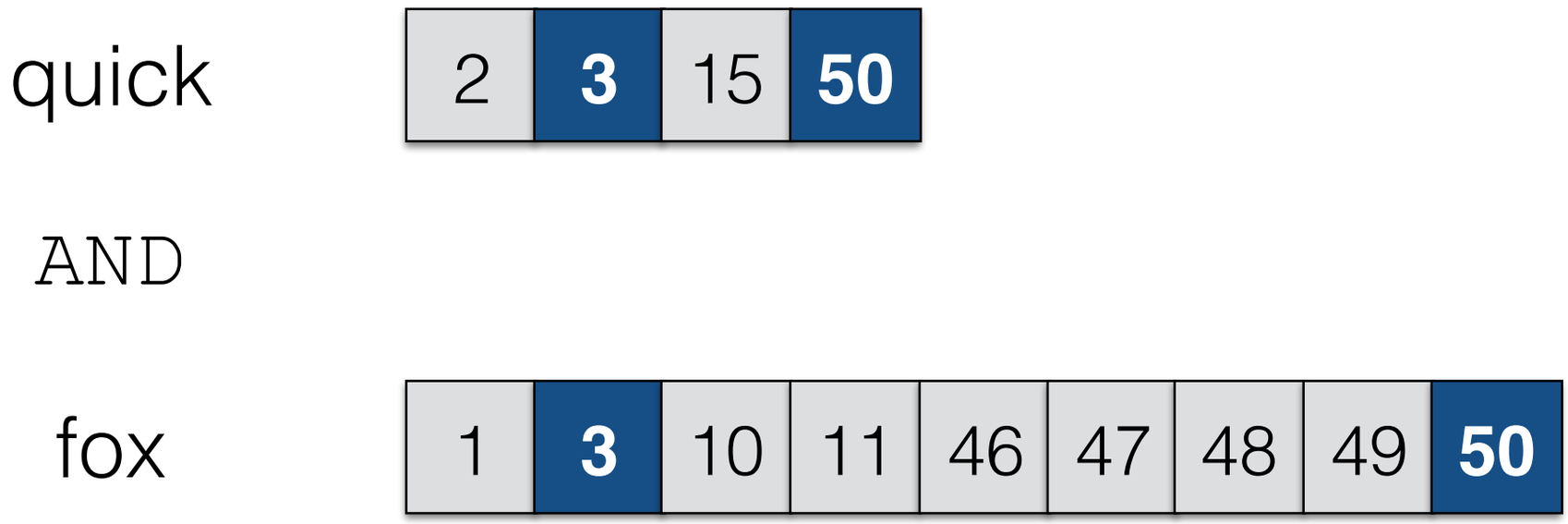


Conjunctions



Conjunctions

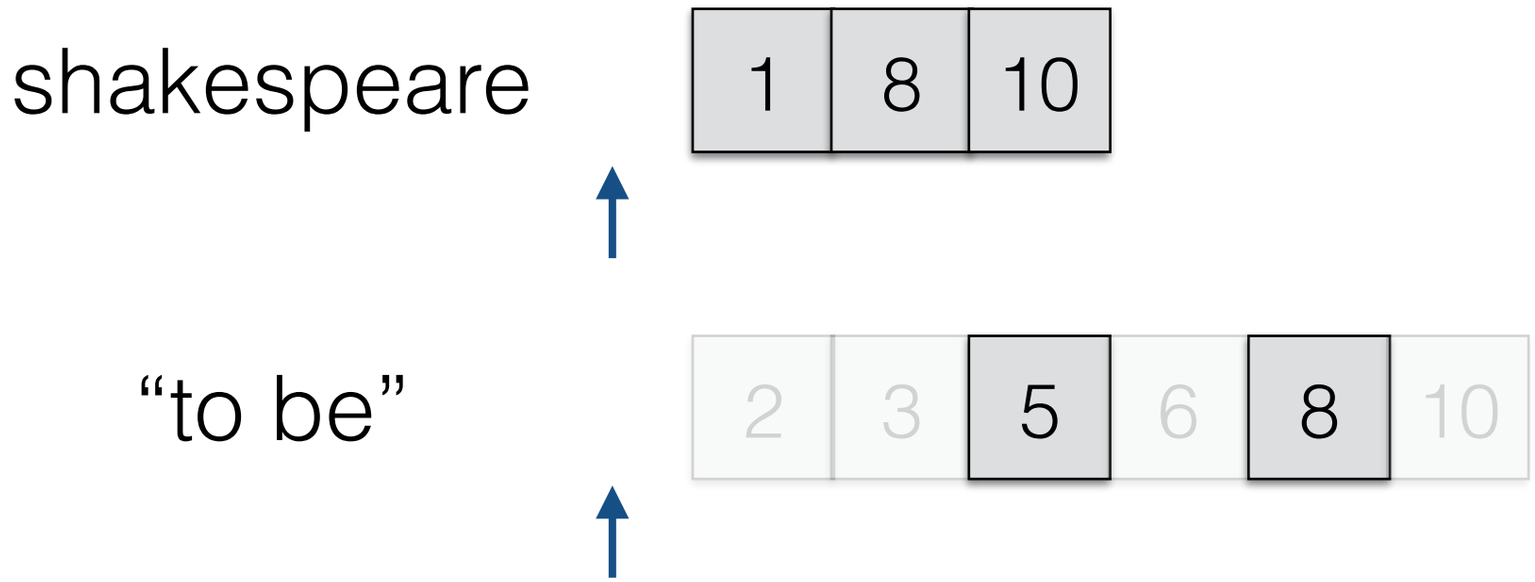
Leapfrog



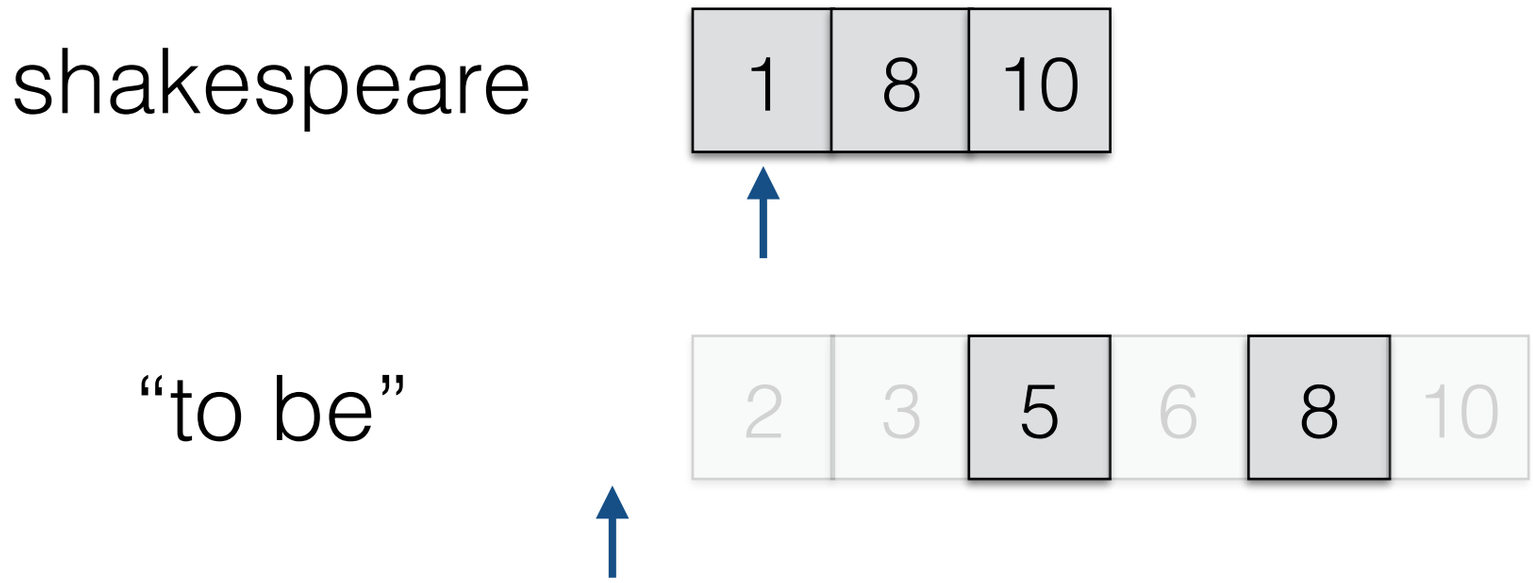
Slow iterators

- “to be”
- finding the next match is not cheap!
- “to be” AND shakespeare

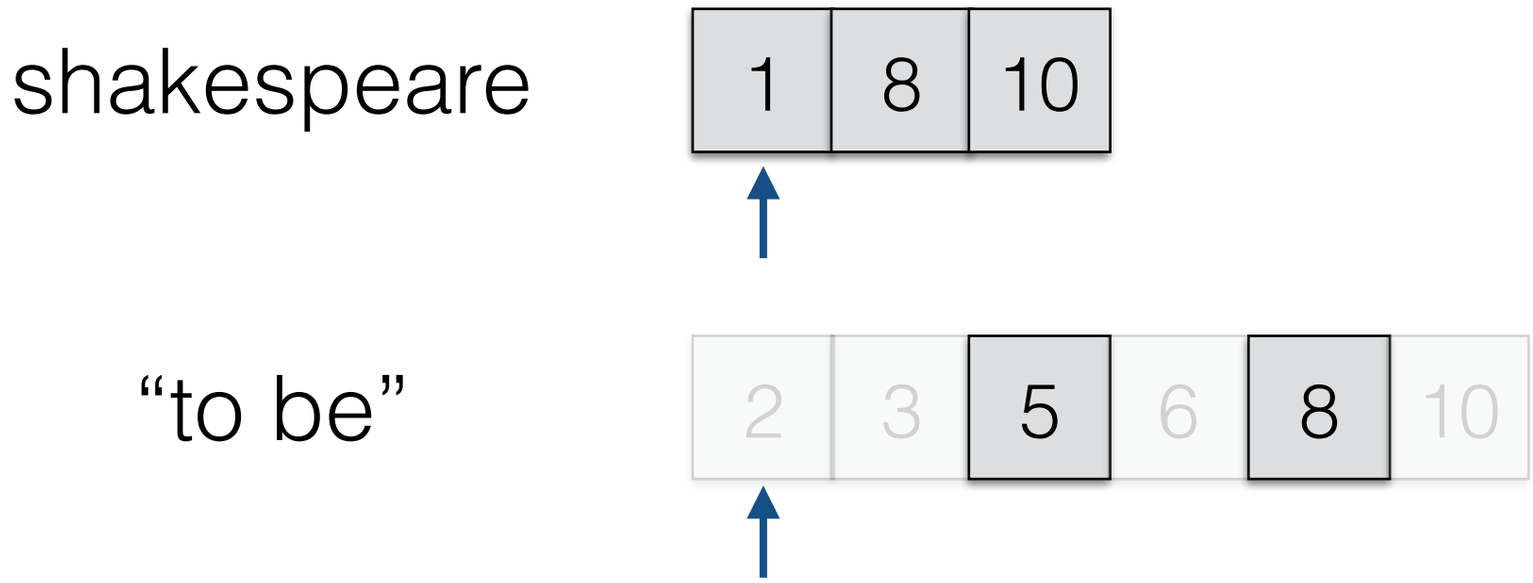
Slow iterators



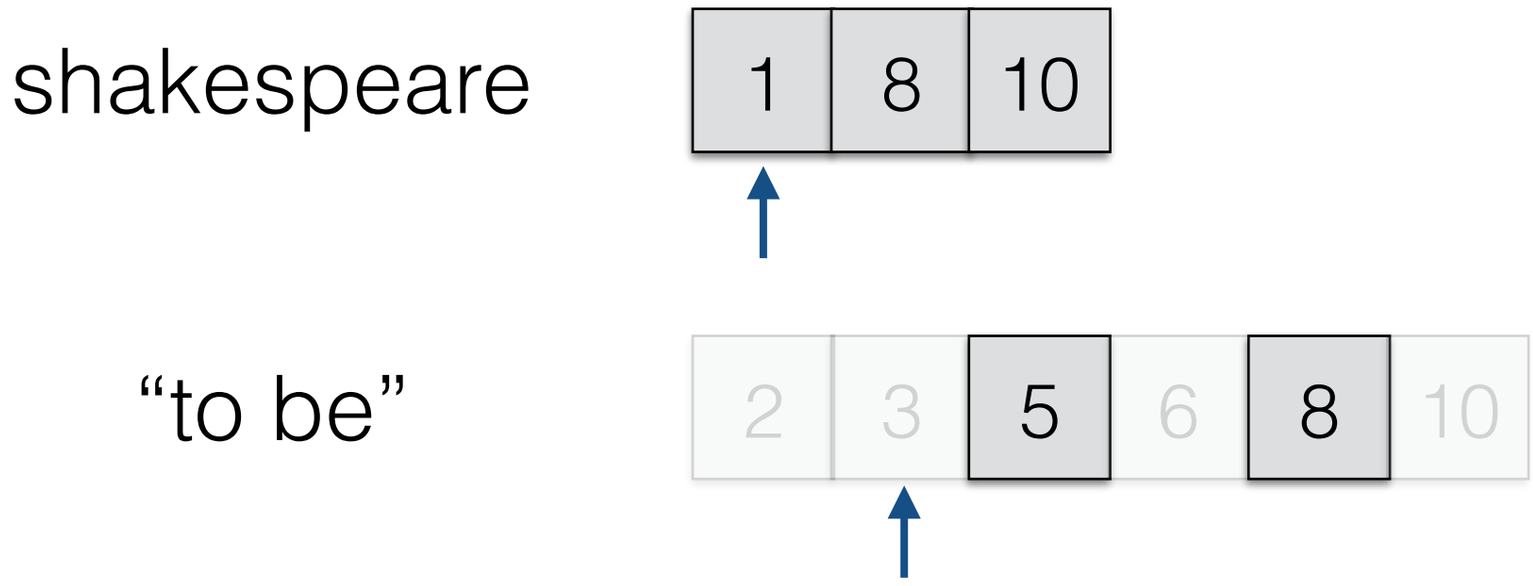
Slow iterators



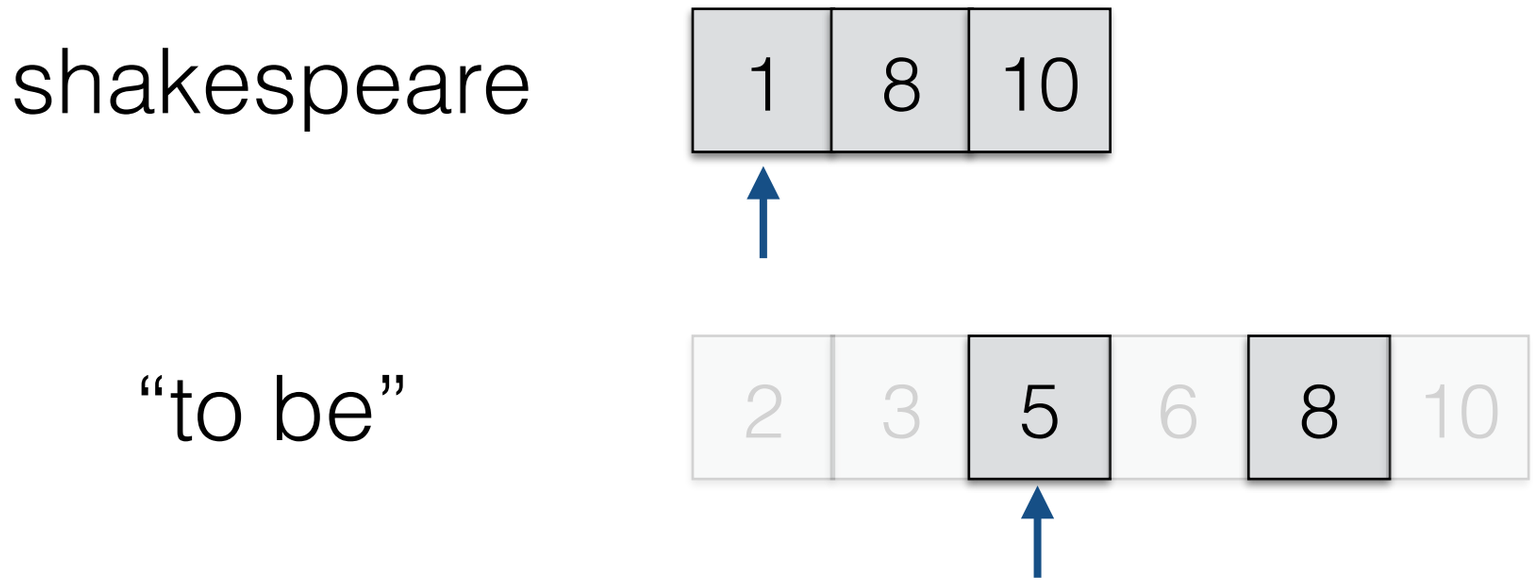
Slow iterators



Slow iterators



Slow iterators



Two-phase iteration

- Lucene 5.1
- Iterator “to be” split into:
  - approximation: to AND be
  - confirmation: check positions

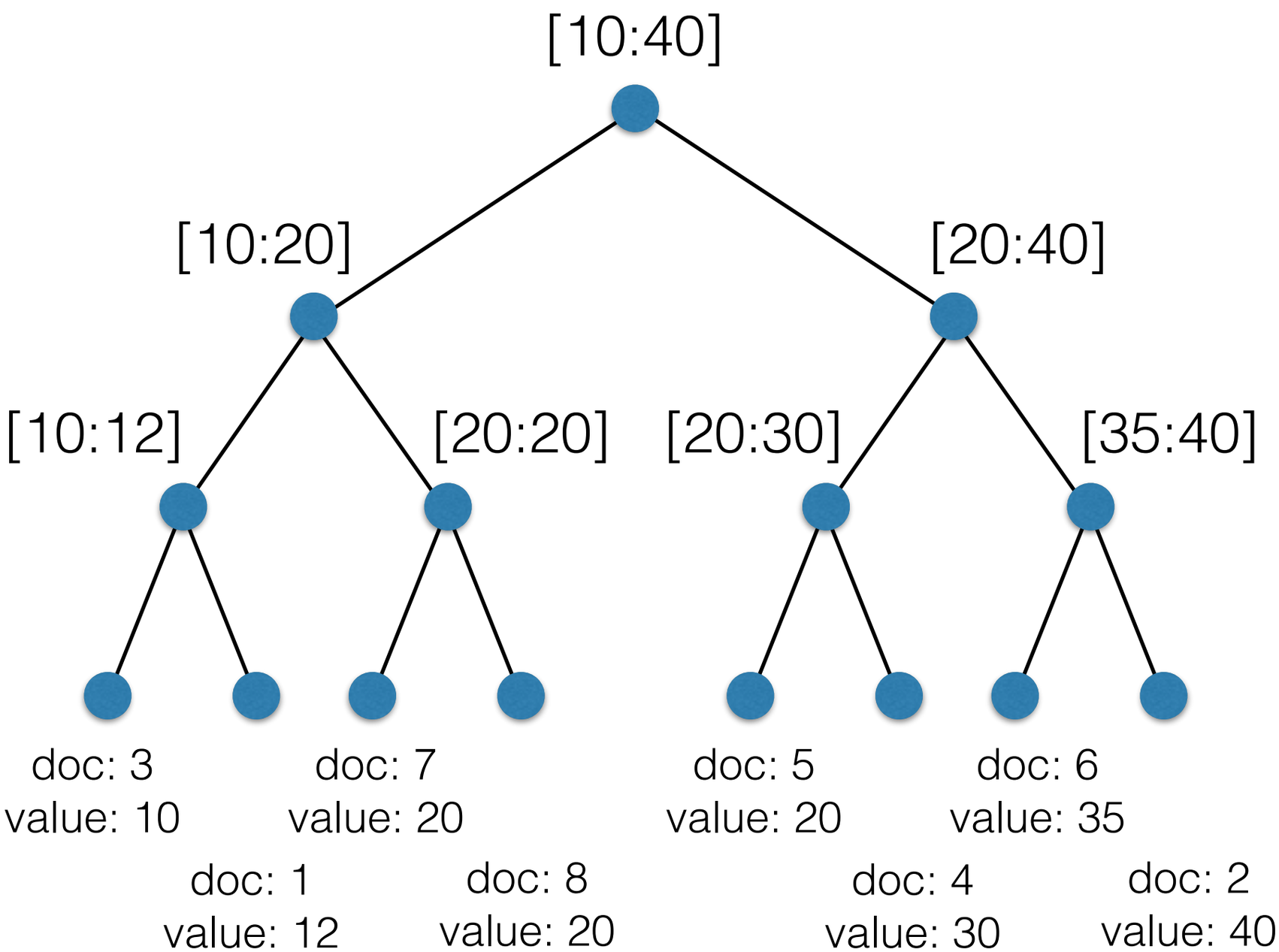
Two-phase iteration

- Reach agreement between all approximations first
- Then start confirming
- “match cost” API for confirming cheaper bits first

Two-phase iteration

- Phrase query
  - Approximation = conjunction
  - Confirmation = check positions
- Script query
  - Approximation = match_all
  - Confirmation = run script
- Compound queries (bool, constant_score) propagate

Query planning: ranges



Query planning: ranges

id:foo42 AND numeric_field:[0 TO 10000000]

Doc values range

- Doc values = column store
- Two-phase iteration:
  - approximation = match_all
  - confirmation = check value against range

Points vs. doc values

- Cost of conjunction with range?
- Points
  - Cost = number of docs that match the range
- Doc values
  - Cost = number of checked documents

Best of both worlds

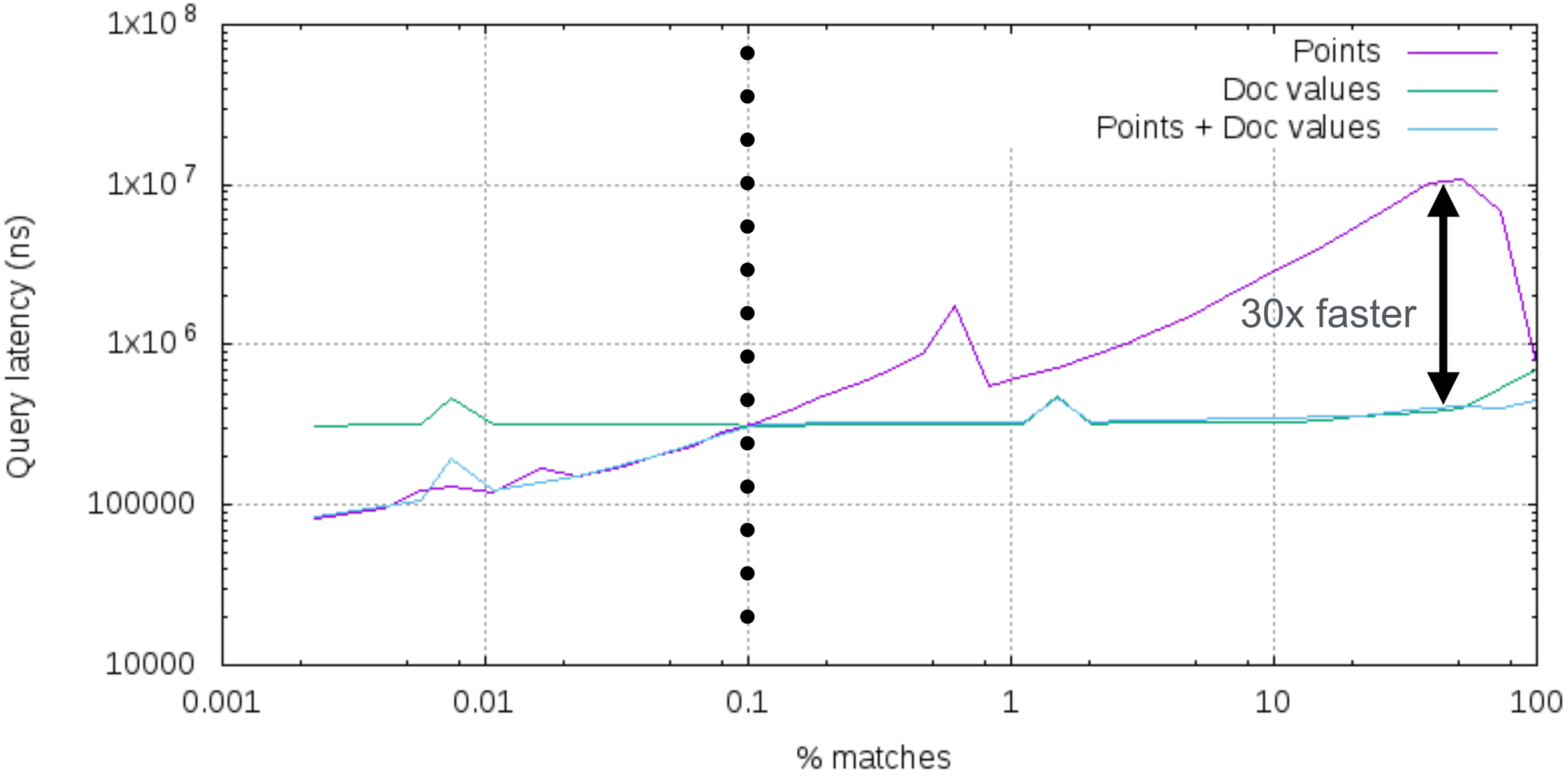
- Since Lucene 6.5

```
Query pointQuery =
    LongPoint.newRangeQuery(“elevation”, 100, 1000);

Query docValueQuery =
    NumericDocValues.newRangeQuery(“age”, 100, 1000);

Query idvQuery = new
    IndexOrDovValuesQuery(pointQuery, docValuesQuery);
```

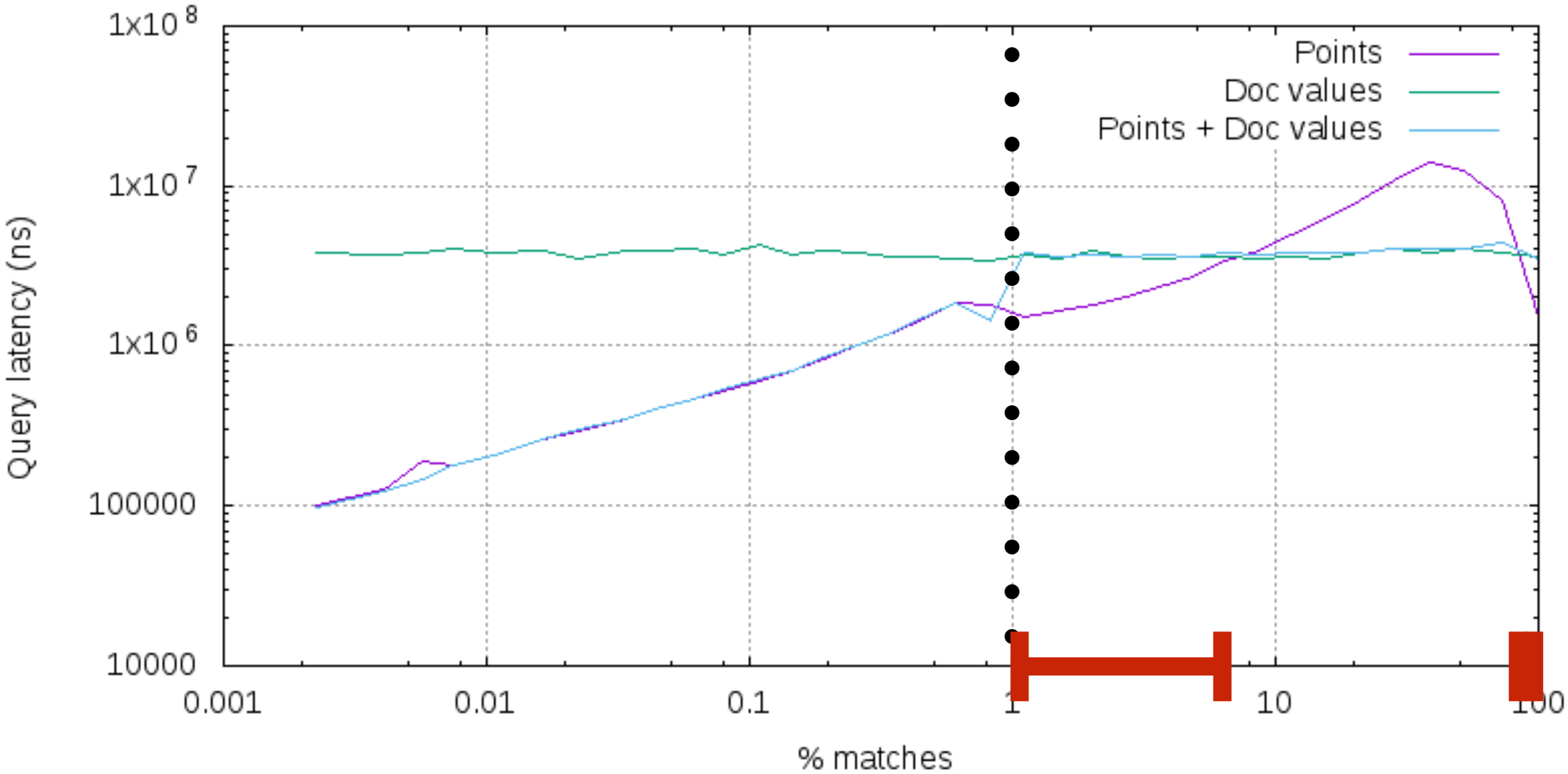
Benchmark

- 10M docs wikipedia subset
- `body:some_term AND last_modification: [2013-10-02 TO 2017-06-11]`
- Beware: log scale!

Points vs. doc values



Points vs. doc values



Query planning

- Good on average, but can make the wrong decision sometimes! Query planning is hard.
- Optimization also applies to
  - geo bounding box queries
  - geo distance queries

TODO

- Prefix/wildcard/regex queries
- Range queries on range fields
- Improve the heuristic!

Questions?