4. Tolerant retrieval
What we have seen so far
Boolean retrieval

Input
Set of documents

Output
Subset of documents

query

lawyer AND Penang AND NOT silver
### Standard Inverted Index

<table>
<thead>
<tr>
<th>a</th>
<th>→</th>
<th>6</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>→</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>→</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>→</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>→</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>→</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>g</td>
<td>→</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Index construction in reality...

1. Collect documents
2. Tokenizing
3. Linguistic preprocessing
4. Build the index (postings list)
Stop words

a
an
and
are
as
at
be
by
for
from
has
he
in

is
it
its
of
on
that
the
to
was
were
will
with
Equivalence classes of terms (types)
Porter Stemmer

(m>0) ENCI -> ENCE valenci -> valence
(m>0) ANCI -> ANCE hesitanci -> hesitatecance
(m>0) IZER -> IZE digitizer -> digitize
(m>0) ABLI -> ABLE conformabli -> conformable
(m>0) ALLI -> AL radicallli -> radical
(m>0) ENTLI -> ENT differentli -> different
(m>0) ELI -> E vileli - > vile
(m>0) OUSLI -> OUS analogousli -> analogous
(m>0) IZATION -> IZE vietnamization -> vietnamize
(m>0) ATION -> ATE predication -> predicate
(m>0) ATOR -> ATE operator -> operate
(m>0) ALISM -> AL feudalism -> feudal
(m>0) IVENESS -> IVE decisiveness -> decisive
(m>0) FULNESS -> FUL hopefulness -> hopeful
(m>0) OUSNESS -> OUS callousness -> callous

https://tartarus.org martin/PorterStemmer/
Skip pointers (performance improvement)

In practice $\sqrt{\text{Number of postings}}$
Help ETH Zurich to flexibly react to new challenges and to set new accents in the future.
Positional index (phrase search feature)

"ETH Zurich"

Help
ETH
Zurich
to
flexibly
react
Search structures
## Inverted Index

<table>
<thead>
<tr>
<th>Term</th>
<th>Document 1</th>
<th>Document 2</th>
<th>Document 3</th>
<th>Document 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>almost</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>be</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>carefully</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>come</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>fair</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hour</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>it</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laertes</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>merely</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>most</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>my</td>
<td>1</td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>possess</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>should</td>
<td>1</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>take</td>
<td>1</td>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>that</td>
<td>1</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>thine</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>this</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thy</td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>1</td>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>to</td>
<td>1</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>upon</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>you</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>your</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Inverted index

But how do we lookup terms based on a query?
Looking up a term

1. Does it exist?
Looking up a term

1. Does it exist?

- almost
- be
- carefully
- come
- fair
- hour
- is
- it
- Laertes
- merely
- most
- my
- possess
- should
- take
- that
- thine
- this
- thy
- time
- to
- upon
- you
- your

should
Looking up a term

1. Does it exist?

almost
be
carefully
come
fair
hour
is
it

Laertes
merely
most
my
possess
should
take
that

thine
this
thy
time
to
upon
you
your

should
Looking up a term

1. Does it exist?

should

could
Looking up a term

1. Does it exist?

should
could
Looking up a term

2. Locate the postings list

should
Looking up a term

2. Locate the postings list

almost 1 3
be 1 2
carefully 1 1
come 3 1
fair 1 2
hour 3 1
is 1 3
it 1 4
Laertes 1 2
merely 1 4
most 1 1
my 1 3
possess 1 4
shall 1 1
take 1 2
that 1 4
thine 1 2
this 1 4
thy 1 2
time 1 2
to 1 4
upon 1 1
you 1 1
your 1 1
The missing part
The missing part

e?
The missing part

e?
The missing part

e?
### Solution 1: hash table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
# Solution 1: hash table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Solution 1: hash table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

a \(\text{hash()}\)
### Solution 1: hash table

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a

hash()
## Solution 1: hash table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>e</td>
</tr>
<tr>
<td>3</td>
<td>h</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
</tr>
<tr>
<td>8</td>
<td>d</td>
</tr>
<tr>
<td>9</td>
<td>c</td>
</tr>
<tr>
<td>10</td>
<td>g</td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>f</td>
</tr>
</tbody>
</table>
Solution 1: hash table

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>e</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>h</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>d</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>g</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>f</td>
<td>3</td>
</tr>
</tbody>
</table>
**Solution 1: hash table**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>e</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>h</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>d</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>g</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>f</td>
<td>3</td>
</tr>
</tbody>
</table>
Solution 1:(hash table

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>e</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>h</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>d</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>g</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>f</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
### Solution 1: hash table

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>e</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>h</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>d</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>1</td>
<td>1</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>g</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>f</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The hash table structure shows how items are stored and retrieved using the hash function `hash()`. The diagram illustrates the mapping of keys to their respective values or pointers to other values in the table.
Limitations of hash tables
Limitations of hash tables

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>e</td>
<td>2</td>
<td>2</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>h</td>
<td>1</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>2</td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>d</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>g</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>f</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a-c?
## Limitations of hash tables

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>e</td>
<td>2</td>
<td>2</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>h</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>2</td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>d</td>
<td>3</td>
<td>1</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>g</td>
<td>2</td>
<td>3</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>f</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**a-c?**
Limitations of hash tables
Limitations of hash tables

No support for range queries
Limitations of hash tables

No support for range queries

Hash function not perfect in real life
Limitations of hash tables

No support for range queries

Hash function not perfect in real life

Space requirements for collision avoidance
Binary tree
Binary tree

at most 2 children per node
Binary tree

weight-balanced
Binary tree

weight-balanced

weight 1

weight 0
Binary tree

imbalanced
Binary tree

weight-balanced
Binary tree

Rebalancing
Binary search tree
Binary search tree
Binary search tree
Binary search tree

Postings lists
Other angle
Binary search tree

Postings lists
Binary search tree (postings lists only at leaves)

Postings lists
Design choice 2: []-intervals

Postings lists
Design choice 2: \([\cdot])\)-intervals

Postings lists
Binary search tree example
Binary tree example

should?
Binary tree example

should?
Binary tree example

should?
Binary tree example

should?
Binary tree example

should?

come
  be
    almost
carefully
fair
is
Laertes
my
most
possess
take
thine
thy

upon
  time
    you

this
should
that

to

hour

it

merely

be

Complexity
Binary tree example

Complexity: logarithmic
B+-tree

```
      o
    /   \
   j     s
  /     / \  
be  not  or  to
```
B+-tree

We could allow more children
B+-tree
B+-tree

Disks block access
B+-tree example
B+-tree example

All leaves at same depth
All non-leaf nodes have between 3 and 5 children
B+-tree example

But it's fine if the root has less.
B+-tree example

Actual terms (and posting lists) only at the leaves
B-tree example

2-4 B-tree
Children per node
B+-trees often have extra leaf pointers
B+-trees often have extra leaf pointers
B+-tree example

General case: #children between d+1 and 2d+1
B+-tree example

General case: #children between \(d+1\) and \(2d+1\)

1-1 B+-tree
2-3 B+-tree
3-5 B+-tree
4-7 B+-tree
Warning: intervals!
Warning: intervals!

n trees

n-1 intervals
Warning: intervals!

2 keys

3 keys

4 keys

3 children

4 children

5 children
B+-tree example

General case: #children between $d+1$ and $2d+1$

1-1 B+-tree
2-3 B+-tree
3-5 B+-tree
4-7 B+-tree

(means #keys between $d$ and $2d$)
B+-tree example

General case: \#children between \(d+1\) and \(2d+1\)

1-1 B+-tree
2-3 B+-tree
3-5 B+-tree
4-7 B+-tree

(means \#keys between \(d\) and \(2d\))
Insertion
Insertion
Insertion
Insertion
Insertion

More than 2d keys!
Insertion
Insertion

```
        4
       /|
      / |\n     1 2 3
     /    |
    /     4
   /      /|
  4      5 6
```
Insertion
Insertion
Insertion
Insertion
Insertion
Insertion
Insertion
Insertion
Insertion
Insertion
Insertion
Insertion
Deletion
Deletion
Deletion
Deletion
Deletion
Deletion
Storing a B+-tree

+ postings lists
Wildcard queries
Wildcard
Wildcard

* Matches any sequence of characters
Wildcard

Matches any sequence of characters

ETH
Switzerland
Zürich
Mathematics
Wildcard

Z*rich

multiple spellings
Wildcard

\[ Z^* \text{rich} \]

multiple spellings

Zurich
Zuerich
Zürich
Wildcard

informat*

word families
Wildcard

informat*  
word families  
informatics  
information  
informative
Wildcard

foo*eth*bar

multiple wildcards
Wildcard

foo*eth*bar

multiple wildcards

fooethbbar
fooethethzuerichbar
Leading wildcard queries

*ics

economics

physics
Trailing wildcard queries

Math* Mathematically Mathematician Mathematics Mathematical
Which is the easiest?

*ics

or

Math*
Which is the easiest?

*ics

or

Math*
B+-tree
B+-tree

ba*
B+-tree

ba*
B+-tree

**ba***

```
   a   aa   ab
  /       \
ac  bad  cd
  |      |    |
  |      |    |
ac  ba  bac
  |    |    |
  |    |
bad  bb  c
  |    |
  |    |
ce  cf  d
```
B+-tree

ba*
B+-tree

ba*

Posting lists
B+-tree

*ac
B+-tree

*ac
B+-tree

*ac
B+-tree

*ac

Build (reverse) B-tree
B+-tree

*ac
B+-tree

*ac
B+-tree

ca...?   *ac

ba c d

a aa ab ba bb bad c ca cab d ec fc
B+-tree

ca...?  *ac

Posting lists for terms ac and bac
Single-wildcard query

Zu*ch
Single-wildcard query

Zu*ch

Zu* AND *ch
Single-wildcard query

Zu*ch

Zu* AND *ch

B+-tree

Reverse B+-tree
Single-wildcard query

Zu*ch

Rewrite

Zu* AND *ch

B+-tree

Reverse B+-tree

Intersection
Single-wildcard query

stati*tics
Single-wildcard query

\[ \text{stati}^{*}\text{tics} \]

Rewrite

\[ \text{stati}^{*}\text{AND} \text{ *tics} \]
Single-wildcard query

\[
\text{stati}^*\text{tics} \\
\text{stati}*\text{AND} \ *\text{tics}
\]

Rewrite

False positives

\text{statics} \ \text{statics}
Single-wildcard query

statist*ics

Rewrite

stati*AND *tics

Post-filtering
Permuterm index
Permuterm index

plant
Permuterm index
Permuterm index
Permuterm index

plant → $plant
   t$plan
   nt$pla
   ant$pl
   lant$p
   plant$
Permuterm index

plant

Rotations

$plant
t$plan
nt$pla
ant$pl
lant$p
plant$

t$pl*
pl*t
Permuterm index

Rotations

plant

$plant
t$plan
nt$pla
ant$pl
lant$p
plant$
Permuterm index
Permuterm index

plant

$plant$
t$plan$
nt$pla$
ant$pl$
lant$p$
plant$
Permuterm index
Permuterm index

$plant$

ant$pl$

lant$p$

nt$p$la$

plant$

plant$

t$plan$

$eth$

eth$

h$set$

th$e$

eth
Permuterm index

$\text{eth}$
$\text{plant}$
ant$\text{pl}$
eth$
h$et$
lant$p$
nt$pla$
plant$
eth$
t$plan
Permuterm index

Use a B+-tree!
Permuterm index (as a B+-tree)
k-gram

*mpu*
k-gram

computer
k-gram

1-grams: $, c, o, m, p, u, t, e, r, $
k-gram

1-grams  $, c, o, m, p, u, t, e, r, $

2-grams  $c, co, om, mp, pu, ut, te, er, r$

computer
**k-gram**

1-grams  $, c, o, m, p, u, t, e, r, $

2-grams  $c, co, om, mp, pu, ut, te, er, r$

3-grams  $co, com, omp, mpu, put, ute, ter, er$

computer
# k-gram

<table>
<thead>
<tr>
<th>1-grams</th>
<th>$, c, o, m, p, u, t, e, r, $</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-grams</td>
<td>$c, co, om, mp, pu, ut, te, er, r$</td>
</tr>
<tr>
<td>3-grams</td>
<td>$co, com, omp, mpu, put, ute, ter, er$</td>
</tr>
<tr>
<td>4-grams</td>
<td>$com, comp, ompu, mput, pute, uter, ter$</td>
</tr>
<tr>
<td>5-grams</td>
<td>$comp, compu, omput, mpute, puter, uter$</td>
</tr>
<tr>
<td>6-grams</td>
<td>$compu, comput, ompute, mputer, puter$</td>
</tr>
<tr>
<td>7-grams</td>
<td>$comput, compute, omputer, mputer$</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
# k-gram

<table>
<thead>
<tr>
<th>1-grams</th>
<th>$, c, o, m, p, u, t, e, r, $</th>
<th>Not very useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-grams</td>
<td>$c, co, om, mp, pu, ut, te, er, r$</td>
<td></td>
</tr>
<tr>
<td>3-grams</td>
<td>$co, com, omp, mpu, put, ute, ter, er$</td>
<td></td>
</tr>
<tr>
<td>4-grams</td>
<td>$com, comp, ompu, mput, pute, uter, ter$</td>
<td></td>
</tr>
<tr>
<td>5-grams</td>
<td>$comp, compu, omput, mpute, puter, uter$</td>
<td></td>
</tr>
<tr>
<td>6-grams</td>
<td>$compu, comput, ompute, mputer, puter$</td>
<td></td>
</tr>
<tr>
<td>7-grams</td>
<td>$comput, compute, omputer, mputer$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### k-gram

<table>
<thead>
<tr>
<th>k-grams</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-grams</td>
<td>$, c, o, m, p, u, t, e, r, $</td>
<td>Not very useful</td>
</tr>
<tr>
<td>2-grams</td>
<td>$c, co, om, mp, pu, ut, te, er, r$</td>
<td></td>
</tr>
<tr>
<td>3-grams</td>
<td>$co, com, omp, mpu, put, ute, ter, er$</td>
<td></td>
</tr>
<tr>
<td>4-grams</td>
<td>$com, comp, ompu, mput, pute, uter, ter$</td>
<td></td>
</tr>
<tr>
<td>5-grams</td>
<td>$comp, compu, omput, mpute, puter, uter$</td>
<td>Not space efficient</td>
</tr>
<tr>
<td>6-grams</td>
<td>$compu, comput, ompute, mputer, puter$</td>
<td></td>
</tr>
<tr>
<td>7-grams</td>
<td>$comput, compute, omputer, mputer$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**k-gram**

1-grams  \( $, c, o, m, p, u, t, e, r, $ \)  
Not very useful

2-grams  \( $c, co, om, mp, pu, ut, te, er, r$ \)

3-grams  \( $co, com, omp, mpu, put, ute, ter, er$ \)

4-grams  \( $com, comp, ompu, mput, pute, uter, ter$ \)

5-grams  \( $comp, compu, omput, mpute, puter, uter$ \)

6-grams  \( $compu, comput, ompute, mputer, puter$ \)

7-grams  \( $comput, compute, omputer, mputer$ \)

...
k-gram

computer  $co  com  omp  mpu  put  ute  ter  er*
terran    $te  ter  err  rra  ran  an*
k-gram index
k-gram index
k-gram index
k-gram index

*com*
k-gram index

*com*
k-gram index

coster

```
  mpu
 /    \   
|     |    
com    ran  ter
 /    /    /   
|    |    |    
$co$ $te$ an$   com er$ err
 /    /    /  
|    |    |    
computer terran terran
```

```
  mpu  omp  put
 /    /    / 
|    |    | 
ran  rra  ute
 /    /    / 
|    |    | 
computer computer computer
```

```
  ter
 / 
|  
ute
```
k-gram index

coste

\[ \text{co} \quad \text{com} \quad \text{mpu} \quad \text{ran} \quad \text{ter} \quad \text{ute} \]

\[ \text{computer} \quad \text{com} \quad \text{computer} \quad \text{computer} \quad \text{terran} \quad \text{terran} \]

\[ \text{co} \quad \text{ste} \quad \text{an} \quad \text{com} \quad \text{er} \quad \text{err} \quad \text{mpu} \quad \text{omp} \quad \text{put} \quad \text{ran} \quad \text{rra} \]

\[ \text{computer} \quad \text{com} \quad \text{computer} \quad \text{computer} \quad \text{terran} \quad \text{terran} \]
k-gram index

coster
Spelling correction
Approaches to spelling errors
Approaches to spelling errors

Always query also for corrected terms
Approaches to spelling errors

Only query for corrected terms if not in the dictionary
Approaches to spelling errors

Only query for corrected terms if not in the dictionary and not enough results
Approaches to spelling errors

Making a spelling suggestion for corrected terms if not in the dictionary and not enough results
Minimum Edit Distance

or

tor

tor
Minimum Edit Distance

or

to
Search Space
Search Space

- or
- to
- + o
- - r
- s / r / o /
Search Space

- **or**
  - **t**
    - **+t**
    - **-r**
    - **s/r/t/**

- **o**
  - **t**
    - **+o**
    - **-o**
    - **s/o/o/**

- **ot**
  - **+t**
  - **-o**
  - **s/o/t/**
Search Space
Search Space
Search Space
Search Space

or -> tor -> tr -> tor -> to
Depth-First-Search
Breadth-First-Search
Search Space
Search Space
Search Space
Search Space
# Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>o</th>
<th>#</th>
<th>t</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>#</td>
<td>t</td>
<td>o</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>o</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>#</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>t</td>
<td>o</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>o</th>
<th>#</th>
<th>#</th>
<th>t</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table represents dynamic programming problems with given costs or values for different inputs and operations.
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Can we directly build this?
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>+1 (substitute a for c)</td>
<td>+1 (add c)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>+1 (add a)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+1 (substitute a for c)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+1</td>
<td>min(2, 1, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>0 (do nothing)</td>
<td></td>
<td>+1 (add c)</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+1 (add a)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>0 (do nothing)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+1 (add a)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+1 (add c)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min(3, 1, 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The diagram shows the transitions between characters, with arrows indicating the dynamic programming approach to find the path from # to "ate".
# Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The sequence `ate` is the longest common subsequence.
## Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The diagram shows the dynamic programming approach to finding the longest common substring between two strings. The arrows indicate the transition from one character to the next, with 'at' and 'ate' being the longest common substring.
# Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>ate</td>
</tr>
</tbody>
</table>

### Diagram:

The diagram illustrates the dynamic programming approach to solving the problem. Each cell represents the accumulated score, and the arrows show the transitions between states.
# Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>a</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

![Diagram of dynamic programming](attachment:image.png)
Problem with "raw" edit distance

cmputer
Problem with "raw" edit distance

500,000 terms
Problem with "raw" edit distance

This is not feasible in practice!
Problem with "raw" edit distance

just a few terms

Solution: pre-select a few terms
Spelling and k-grams

Assumption:

Two terms

within a small edit distance

have many k-grams in common
Spelling and k-grams

Assumption:

Two terms within a small edit distance have many k-grams in common
Spelling and k-grams

Assumption:

Two terms

within a small edit distance

have many k-grams in common
Spelling and k-grams
Spelling and k-grams

$co$
$com$

$cm$
$cmp$

computer ↔ cmputer
Spelling and k-grams
Spelling and k-grams

computer

route

$co$
com
er$
ter

mpu
put
ute

$ro$
rou
out
te$
Spelling and k-grams
Spelling and k-grams

cmputer
Spelling and k-grams

cmputer $\rightarrow$ cm, cmp, mpu, put, ute, ter, er$
Spelling and k-grams

computer → $cm, cmp, mpu, put, ute, ter, er$

Diagram:

- **computer**
- **terran**
- **terran**

- **com**
  - **$co**
  - **$te**
  - **an$**

- **er$$**

- **err**

- **mpu**
  - **computer**
  - **computer**
  - **terran**

- **ran**
  - **terran**
  - **terran**

- **ute**
  - **computer**
  - **computer**
  - **terran**

239
Spelling and k-grams

cmputer → $cm, cmp, mpu, put, ute, ter, er$

computer: five 3-grams in common

terran: one 3-gram in common
Spelling and k-grams

cmputer $cm, cmp, mpu, put, ute, ter, er$

computer: five 3-grams in common

terran: one 3-gram in common

Threshold, e.g., 3
Issue: false "positives"
Jaccard coefficient

\begin{align*}
\text{or} & \quad \text{d} \quad \text{b} \\
\text{bo} & \quad \text{rd} \\
\text{\cap} & \quad \text{\cup}
\end{align*}
Jaccard coefficient

\[ \frac{\text{or} \cap \text{board}}{\text{or} \cup \text{board}} = \frac{3}{12} = 0.25 \]
Jaccard coefficient
Jaccard coefficient

\[
\text{Jaccard coefficient} = \frac{\text{intersection}}{\text{union}} = \frac{|A \cap B|}{|A \cup B|} = \frac{1}{11} = 0.09
\]
Jaccard coefficient
Jaccard coefficient

$\frac{\text{co} \cap \text{mpu}}{\text{co} \cup \text{cmp}} = \frac{5}{10} = 0.5$
Updated method

Get k-grams from the query term
Updated method

Get k-grams from the query term

Look them up in the k-gram index
Updated method

Get k-grams from the query term

Look them up in the k-gram index

Compute Jaccard coefficients
Updated method

Get k-grams from the query term

Look them up in the k-gram index

Compute Jaccard coefficients

Keep terms within small Jaccard coefficients
Resource issue

We would need to extract k-grams from every term found in the k-gram index, compute the intersection and union, and then the Jaccard coefficients.
Shortcut

We *already know* the k-grams that the query term and the found term have in common.
Shortcut

The denominator can be calculated with just the **found term's length**
Shortcut

#query term's k-grams + #found term's k-grams - #intersection

The denominator can be calculated with just the **found term's number of k-grams**
Spelling and k-grams

cmputer $\rightarrow$ cm, cmp, mpu, put, ute, ter, er$

\begin{align*}
\text{com} & \quad \text{mpu} \\
\text{com} & \quad \text{ran} \\
\text{com} & \quad \text{ter} \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
\text{cmp} & \\
\text{mpu} & \\
Shortcut

$J(\text{computer, cmputer})$
Shortcut

\[ \frac{5}{7 + 8 - 5} \]
Shortcut

\[
\frac{5}{10}
\]
Final method

Get k-grams from the query term

Look them up in the k-gram index

Compute edit distances

Keep terms within small edit distances
Context-sensitive spelling correction

graduate form ETH
Context-sensitive spelling correction

graduate form ETH

✓  ✓  ✓
Context-sensitive spelling correction

graduation  ETA
graduated  ETC
graduate  ETH
form
from
foam
Context-sensitive spelling correction

graduate form ETH

Look up biwords from other users

graduate from from ETH
Phonetic correction
Soundex algorithm

Every word

4 characters fingerprint
Examples

Computer

Cmputer

Zurich

Information

Letter

Retrieval
## Soundex algorithm

<table>
<thead>
<tr>
<th>Change...</th>
<th>To...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A E H I O U W Y</td>
<td>0</td>
</tr>
<tr>
<td>B F P V</td>
<td>1</td>
</tr>
<tr>
<td>C G J K Q S X Z</td>
<td>2</td>
</tr>
<tr>
<td>D T</td>
<td>3</td>
</tr>
<tr>
<td>L</td>
<td>4</td>
</tr>
<tr>
<td>M N</td>
<td>5</td>
</tr>
<tr>
<td>R</td>
<td>6</td>
</tr>
</tbody>
</table>
Examples

Computer

Cmputer

Zurich

Information

Letter

Retrieval
Examples

Computer → C0510306
Cmputer → C510306
Zurich → Z06020
Information → I510650305
Letter → L03306
Retrieval → R0360104
(Repeatedly) remove "duplicates"

Computer  ➔ C0510306  ➔ C0510306

Cmputer  ➔ C510306  ➔ C510306

Zurich  ➔ Z06020  ➔ Z06020

Information  ➔ I510650305  ➔ I510650305

Letter  ➔ L03306  ➔ L0306

Retrieval  ➔ R0360104  ➔ R0360104
Remove zeros

Computer  ➔  C0510306  ➔  C0510306  ➔  C5136

Cmputer  ➔  C510306  ➔  C510306  ➔  C5136

Zurich  ➔  Z06020  ➔  Z06020  ➔  Z62

Information  ➔  I510650305  ➔  I510650305  ➔  I516535

Letter  ➔  L03306  ➔  L0306  ➔  L36

Retrieval  ➔  R0360104  ➔  R0360104  ➔  R3614
## Pad and trim

<table>
<thead>
<tr>
<th>Category</th>
<th>Code</th>
<th>Code 2</th>
<th>Code 3</th>
<th>Code 4</th>
<th>Code 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>C0510306</td>
<td>C0510306</td>
<td>C5136</td>
<td>C513</td>
<td></td>
</tr>
<tr>
<td>Cmputer</td>
<td>C510306</td>
<td>C510306</td>
<td>C5136</td>
<td>C513</td>
<td></td>
</tr>
<tr>
<td>Zurich</td>
<td>Z06020</td>
<td>Z06020</td>
<td>Z62</td>
<td>Z620</td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>I510650305</td>
<td>I510650305</td>
<td>I516535</td>
<td>I516</td>
<td></td>
</tr>
<tr>
<td>Letter</td>
<td>L03306</td>
<td>L0306</td>
<td>L36</td>
<td>L360</td>
<td></td>
</tr>
<tr>
<td>Retrieval</td>
<td>R0360104</td>
<td>R0360104</td>
<td>R3614</td>
<td>R361</td>
<td></td>
</tr>
</tbody>
</table>
Spelling and k-grams