

# Introduction to Informa

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

Lecture 4: Dictionaries and retrieval

# This lecture

---

- Dictionary data structures
  - “Tolerant” retrieval
    - Wild-card
    - Spelling co
- Assignment Project Exam Help
- <https://eduassistpro.github.io/>
- Add WeChat edu\_assist\_pro



# Dictionary data structures for inverted indexes

---

- The dictionary data structure stores the term vocabulary, document frequency, pointers to each postings list ... **Assignment Project Exam Help**

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



# A naïve dictionary

---

- An array of struct:

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

char[20]   int                      Postings \*

20 bytes   4/8 bytes           4/8 bytes

- How do we store a dictionary in memory efficiently?
- How do we quickly look up elements at query time?



# Dictionary data structures

---

- Two main choices:
    - Hash table
    - Tree
  - Some IR systems use B-trees
- <https://eduassistpro.github.io/>
- Add WeChat edu\_assist\_pro



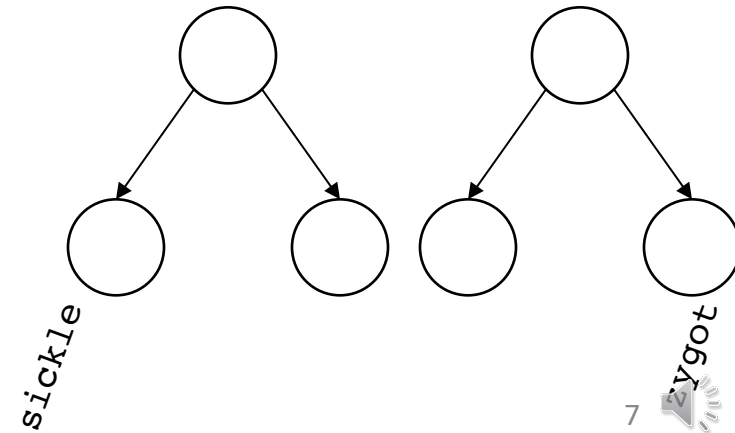
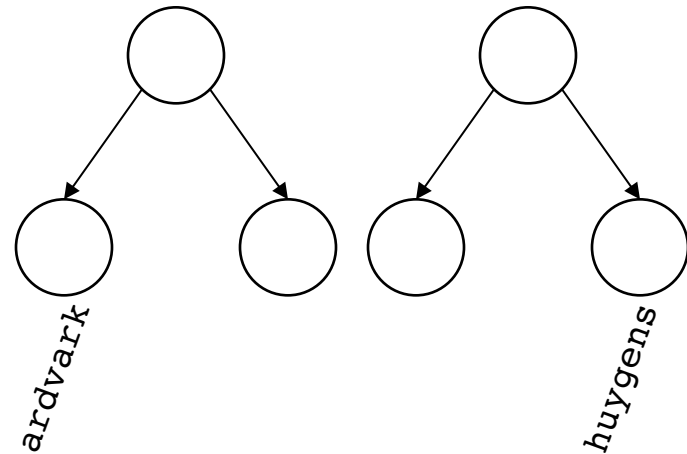
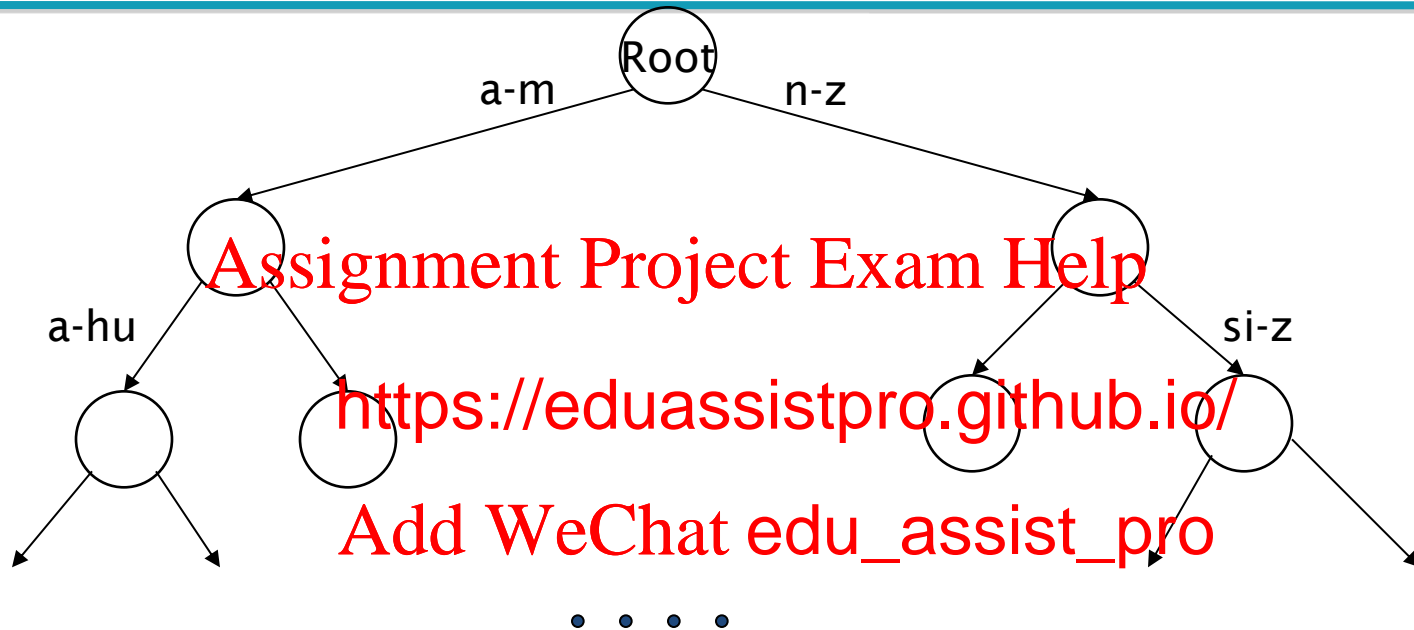
# Hashes

---

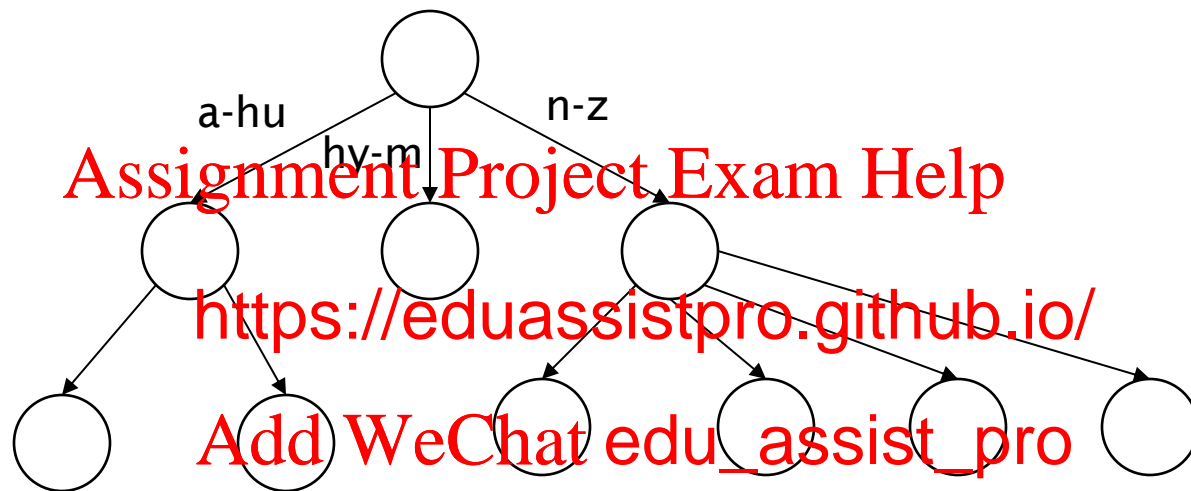
- Each vocabulary term is hashed to an integer
  - (We assume you've seen hashtables before)
- Pros: **Assignment Project Exam Help**
  - Lookup is fast **<https://eduassistpro.github.io/>**
- Cons: **Add WeChat edu\_assist\_pro**
  - No easy way to find minor v
    - judgment/judgement
  - No prefix search **[tolerant retrieval]**
  - If vocabulary keeps growing, need to occasionally do the expensive operation of rehashing *everything*



# Tree: binary tree



# Tree: B-tree



- Definition: Every internal node has a number of children in the interval  $[a, b]$  where  $a, b$  are appropriate natural numbers, e.g.,  $[2, 4]$ .



# Trees

- Simplest: binary tree
- More usual: B-trees
- Trees require a standard ordering of characters and hence strings ... but we
- Pros:
  - Solves the prefix problem (the `edu_assist` with `hyp`)
- Cons:
  - Slower:  $O(\log M)$  [and this requires *balanced* tree]
  - Rebalancing binary trees is expensive
    - But B-trees mitigate the rebalancing problem

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat [edu\\_assist\\_pro](https://eduassistpro.github.io/)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

## WILD-CARD QUERIES



# Wild-card queries: \*

- ***mon\****: find all docs containing any word beginning “mon”.
- Easy with bin words in range ***mon***: retrieve all words in range ***mon***. <https://eduassistpro.github.io/>
- ***\*mon***: find words ending ***mon***. harder
  - Maintain an additional B-tree for terms *backwards*.  
Can retrieve all words in range: ***nom ≤ w < non***.

Exercise: from this, how can we enumerate all terms meeting the wild-card query ***pro\*cent***?



# Query processing

- At this point, we have an enumeration of all terms in the dictionary that match the wild-card query.
- We still have to look up the postings for each enumerated term.  
**Assignment Project Exam Help**  
<https://eduassistpro.github.io/>
- E.g., consider the query:  
**Add WeChat edu\_assist\_pro**  
*se\*ate AND fil\*er*

This may result in the execution of many Boolean *AND* queries.



# B-trees handle \*'s at the end of a query term

- How can we handle \*'s in the middle of query term?
  - *co\*tion*
- We could look up *co\** AND *\*tion* in a B-tree and intersect the results.
  - Expensive
  - Still need *verification* to remove *irrelevant*
- The solution: transform wild-card queries so that the \*'s occur at the end
- This gives rise to the **Permuterm** Index.

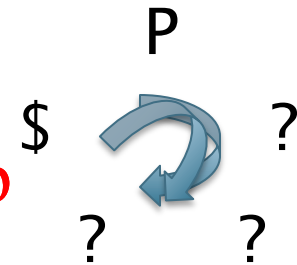


# Permuterm index

- For term **hello**, index under:

- hello\$, ello\$h, llo\$he, lo\$hel, o\$hell**

where \$ is a special symbol.



- Queries:

- P** Exact match **P\$**
  - P\*** Range match **\$P\***
  - \*P** Range match **P\$\***
  - \*P\*** Range match **P\***
  - P\*Q** Range match **Q\$P\***
  - P\*Q\*R** ??? Exercise!

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

hy not **P\*\$\***

Query = **hel\*o**  
**P=hel, Q=o**  
 Lookup **o\$hel\***



# Permuterm query processing

- Rotate query wild-card to the right
- Now use B-tree lookup as before.
- *Permuterm pr* *exicon size*

Assignment Project Exam Help

<https://eduassistpro.github.io/>  
for English.

Add WeChat edu\_assist\_pro



# Bigram ( $k$ -gram) indexes

- Enumerate all  $k$ -grams (sequence of  $k$  chars) occurring in any term
- e.g., from text **Assignment Project Exam Help** *"April is the cruelest month"* we get the 2-grams ( <https://eduassistpro.github.io/>

*\$a,ap,pr,ri,Add,WeChat,edu\_assist,pro,cr,ru,  
ue,el,le,es,st,t\$, \$m,mo,on,nt,h\$*

- \$ is a special word boundary symbol
- Maintain a second inverted index from bigrams to dictionary terms that match each bigram.

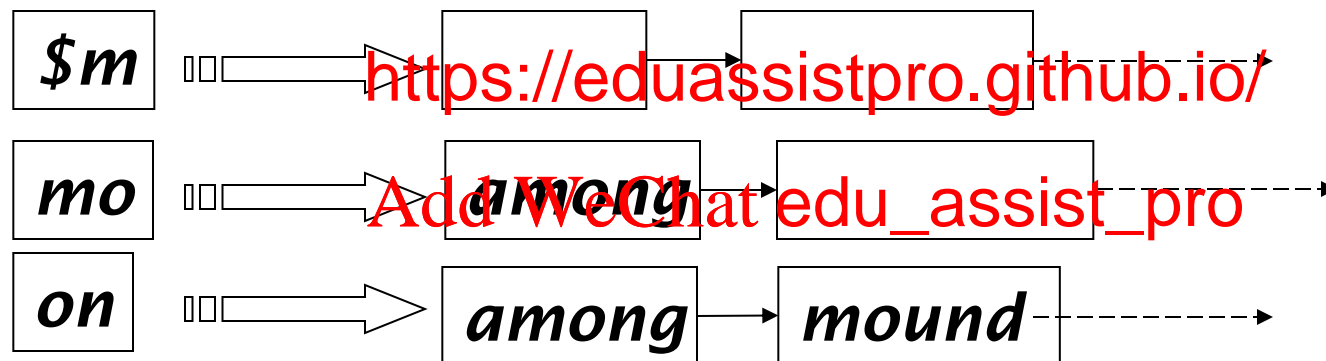




# Bigram index example

- The  $k$ -gram index finds *terms* based on a query consisting of  $k$ -grams (here  $k=2$ ).

Assignment Project Exam Help



# Processing wild-cards

- Query ***mon***\* can now be run as
  - ***\$m AND mo AND on***
- Gets terms that match AND version of our wildcard query.  
<https://eduassistpro.github.io/>
- But we'd enumerate ***moon***.  
Add WeChat edu\_assist\_pro
- Must **verify** these terms against the index.
- Surviving enumerated terms are then looked up in the term-document inverted index.
- Fast, space efficient (compared to permuterm).



# Processing wild-card queries

- As before, we must execute a Boolean query for each enumerated, filtered term.
- Wild-cards can result in expensive query execution (very large dis)
  - `pyth* AND prog*`
- If you encourage “laziness”

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

I respond!

Search

Type your search terms, use '\*' if you need to.  
E.g., Alex\* will match Alexander.



# Resources

- IIR 3, MG 4.2
- Efficient spell retrieval:
  - K. Kukich. Techniques for automatically correcting words in text. ACM Computing Surveys 24(4), Dec 1992.
  - J. Zobel and P. <https://eduassistpro.github.io/> ches in large lexicons. Softw 5(3), March 1995.  
<http://citeseer.ist.psu.edu/zobel95find>
  - Mikael Tillenius: Efficient Generation a pelling Error Corrections. Master's thesis at Sweden's Royal Institute of Technology.  
<http://citeseer.ist.psu.edu/179155.html>
- **Nice, easy reading on spell correction:**
  - Peter Norvig: How to write a spelling corrector  
<http://norvig.com/spell-correct.html>

