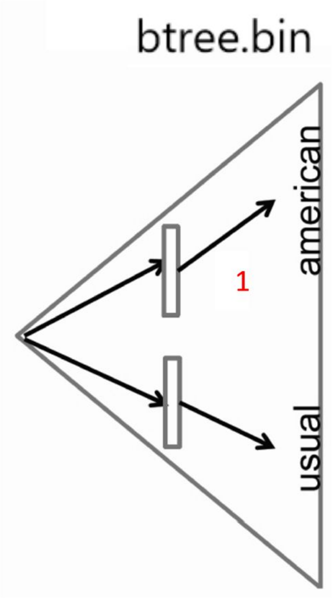
Extending the Functionality of a B+ Tree

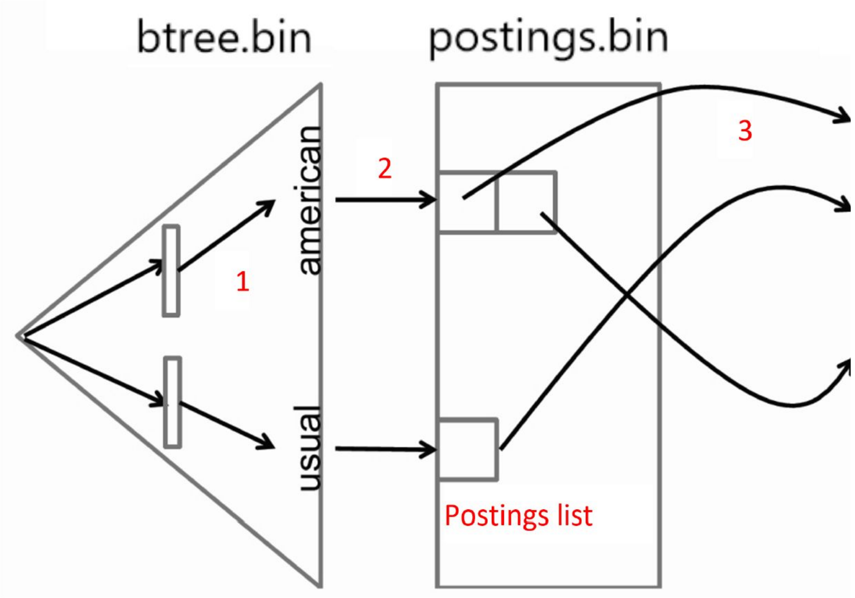
Database Applications - Recitation 11
Zeinab Khalifa
April 2nd, 2020

What are we trying to solve?

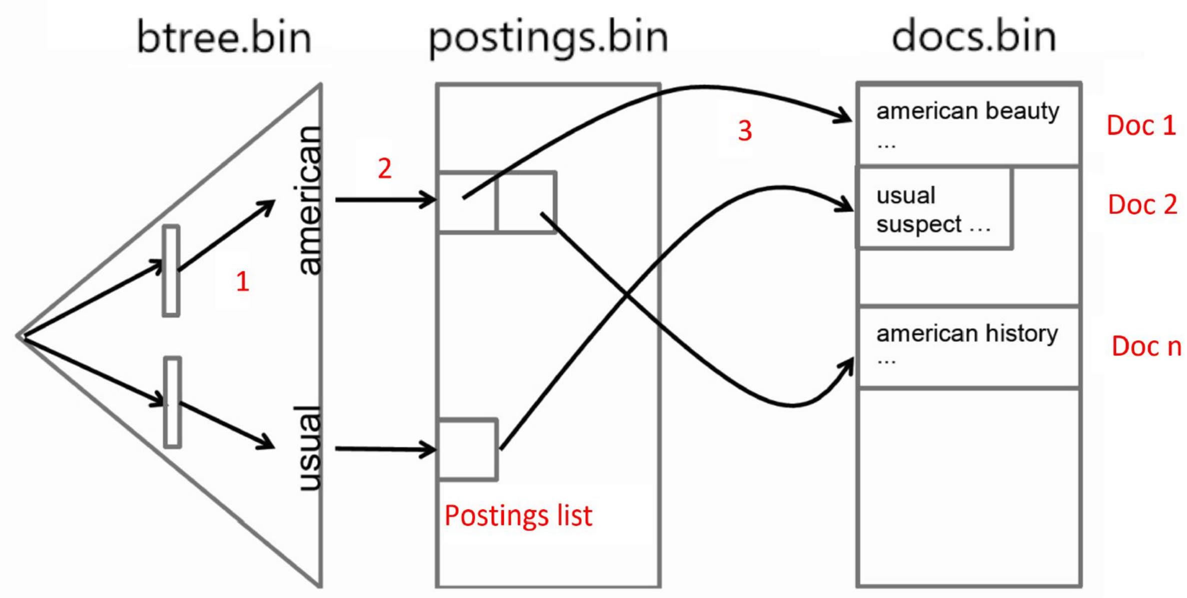
High-Level Overview



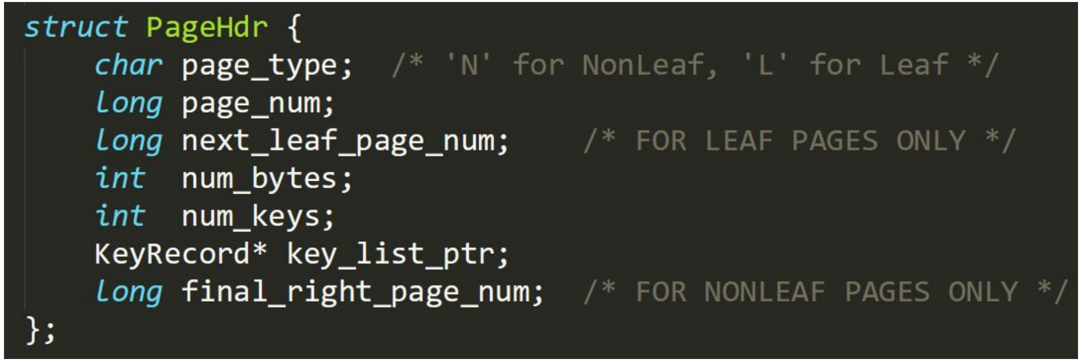
High-Level Overview



High-Level Overview

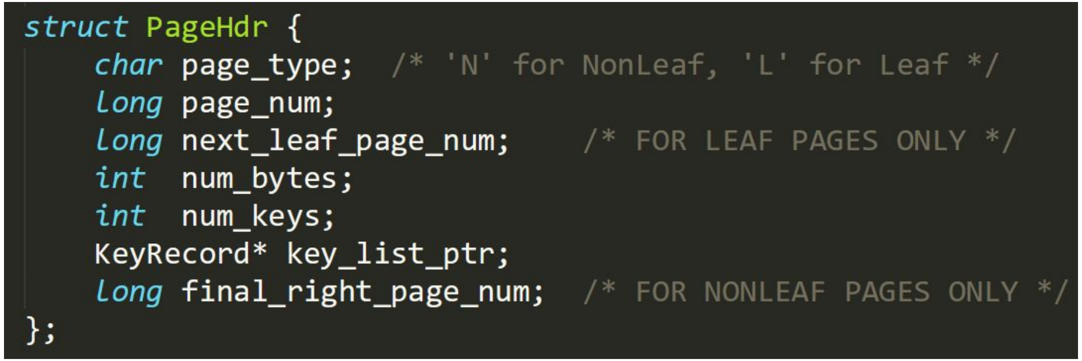


Understanding the Tree & KeyRecord Structures

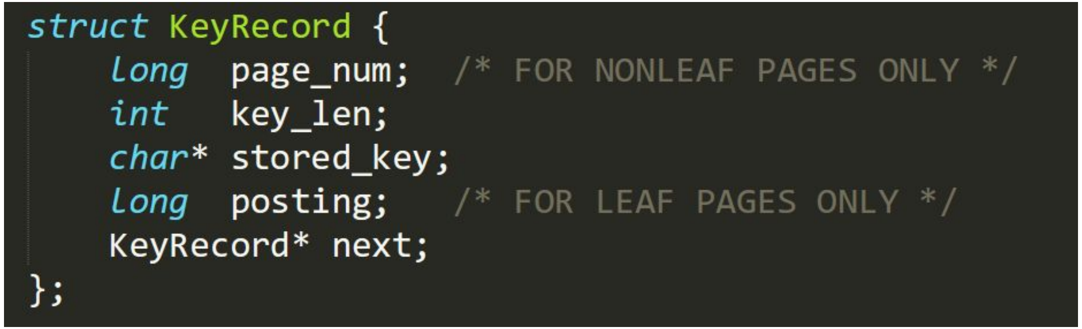


```
struct PageHdr {
    char page_type;  /* 'N' for NonLeaf, 'L' for Leaf */
    long page_num;
    long next_leaf_page_num;    /* FOR LEAF PAGES ONLY */
    int  num_bytes;
    int  num_keys;
    KeyRecord* key_list_ptr;
    long final_right_page_num;  /* FOR NONLEAF PAGES ONLY */
};
```

Understanding the Tree & KeyRecord Structures

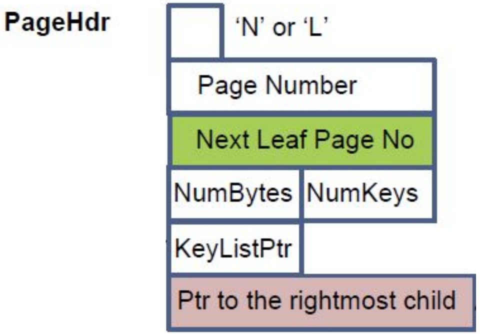


```
struct PageHdr {
    char page_type;  /* 'N' for NonLeaf, 'L' for Leaf */
    long page_num;
    long next_leaf_page_num;    /* FOR LEAF PAGES ONLY */
    int  num_bytes;
    int  num_keys;
    KeyRecord* key_list_ptr;
    long final_right_page_num;  /* FOR NONLEAF PAGES ONLY */
};
```



```
struct KeyRecord {
    long  page_num;  /* FOR NONLEAF PAGES ONLY */
    int   key_len;
    char* stored_key;
    long  posting;   /* FOR LEAF PAGES ONLY */
    KeyRecord* next;
};
```

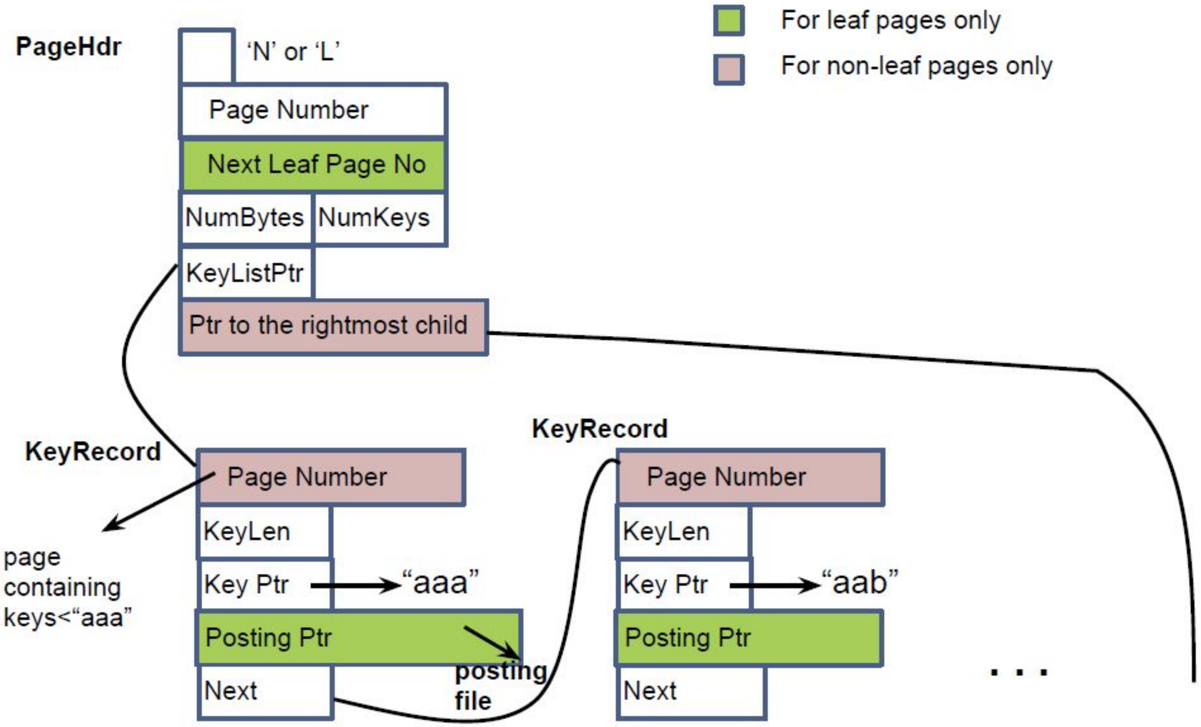
Bree & KeyRecord Example



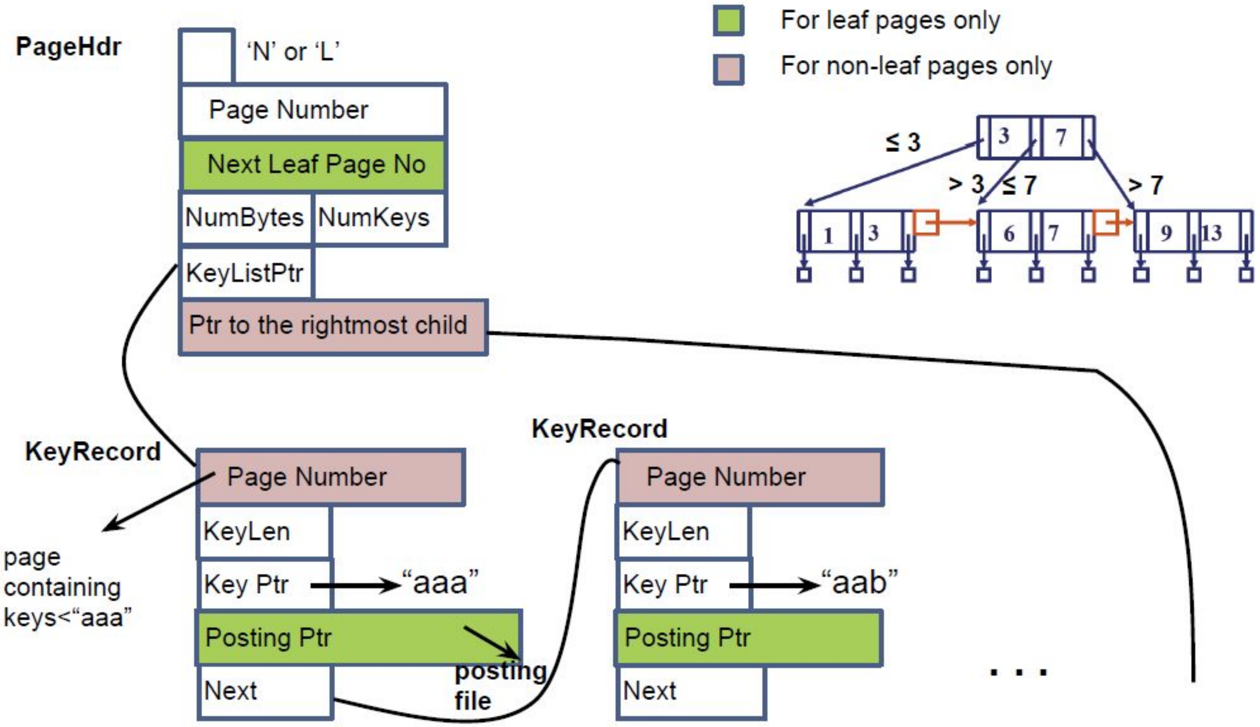
🟩 For leaf pages only

🟫 For non-leaf pages only

Bree & KeyRecord Example



Bree & KeyRecord Example



🟩 For leaf pages only

🟥 For non-leaf pages only

PageHdr

- 'N' or 'L'
- Page Number
- Next Leaf Page No
- NumBytes
- NumKeys
- KeyListPtr
- Ptr to the rightmost child

KeyRecord

- Page Number
- KeyLen
- Key Ptr → "aaa"
- Posting Ptr
- Next

page containing keys<"aaa"

posting file

KeyRecord

- Page Number
- KeyLen
- Key Ptr → "aab"
- Posting Ptr
- Next

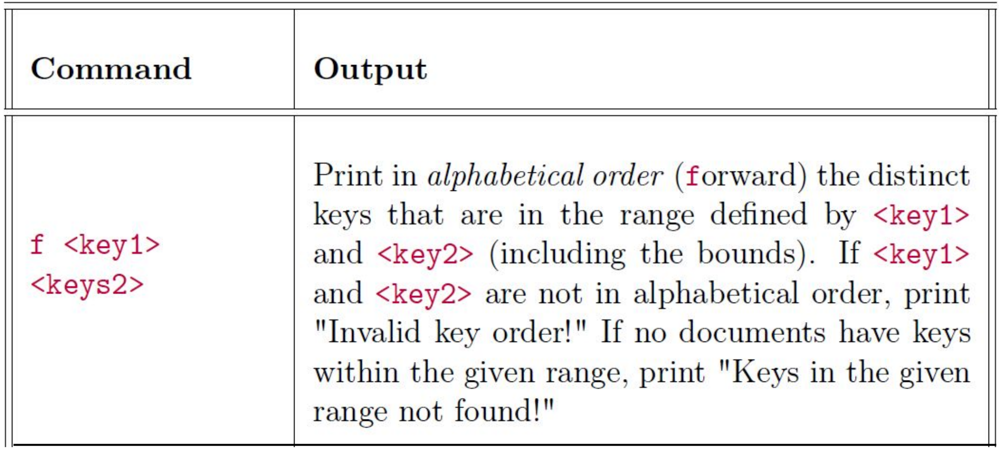
. . .

Existing Functionality

- C
- i <doc>
- p <num>
- s <key>
- S <key>
- T
- q

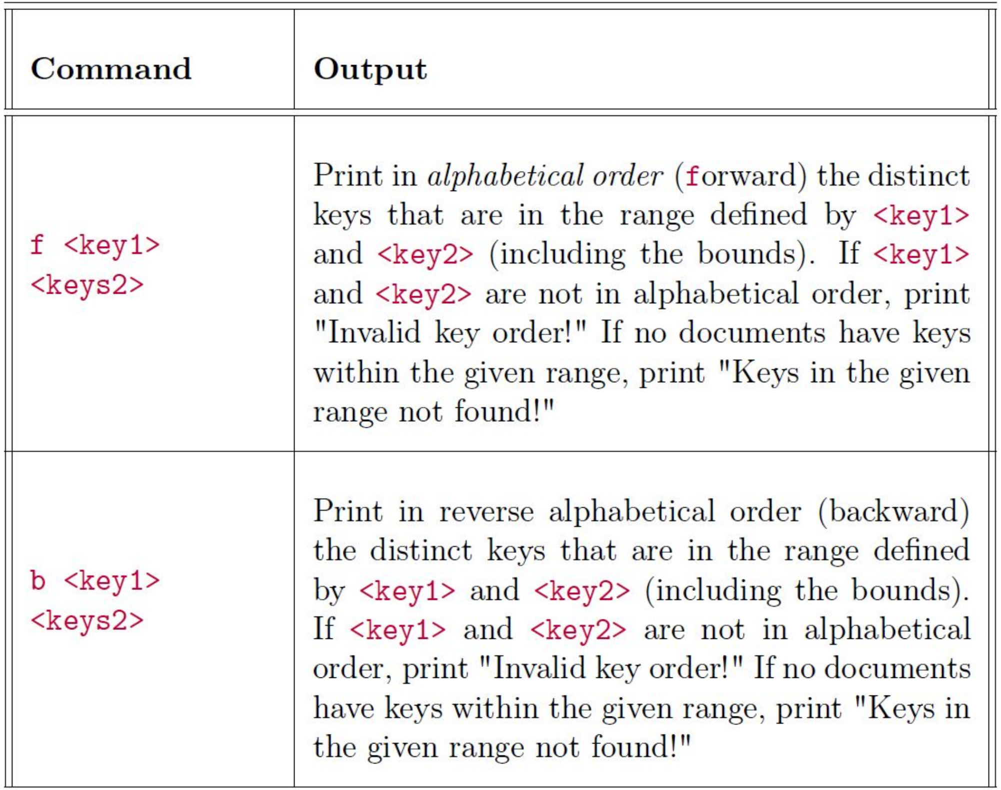
Project structure & Demo

What do you need to implement?



Command	Output
`f <key1> <keys2>`	Print in *alphabetical order* (`f`orward) the distinct keys that are in the range defined by `<key1>` and `<key2>` (including the bounds). If `<key1>` and `<key2>` are not in alphabetical order, print "Invalid key order!" If no documents have keys within the given range, print "Keys in the given range not found!"

What do you need to implement?



Command	Output
f <key1> <keys2>	Print in *alphabetical order* (forward) the distinct keys that are in the range defined by <key1> and <key2> (including the bounds). If <key1> and <key2> are not in alphabetical order, print "Invalid key order!" If no documents have keys within the given range, print "Keys in the given range not found!"
b <key1> <keys2>	Print in reverse alphabetical order (backward) the distinct keys that are in the range defined by <key1> and <key2> (including the bounds). If <key1> and <key2> are not in alphabetical order, print "Invalid key order!" If no documents have keys within the given range, print "Keys in the given range not found!"

How to start?

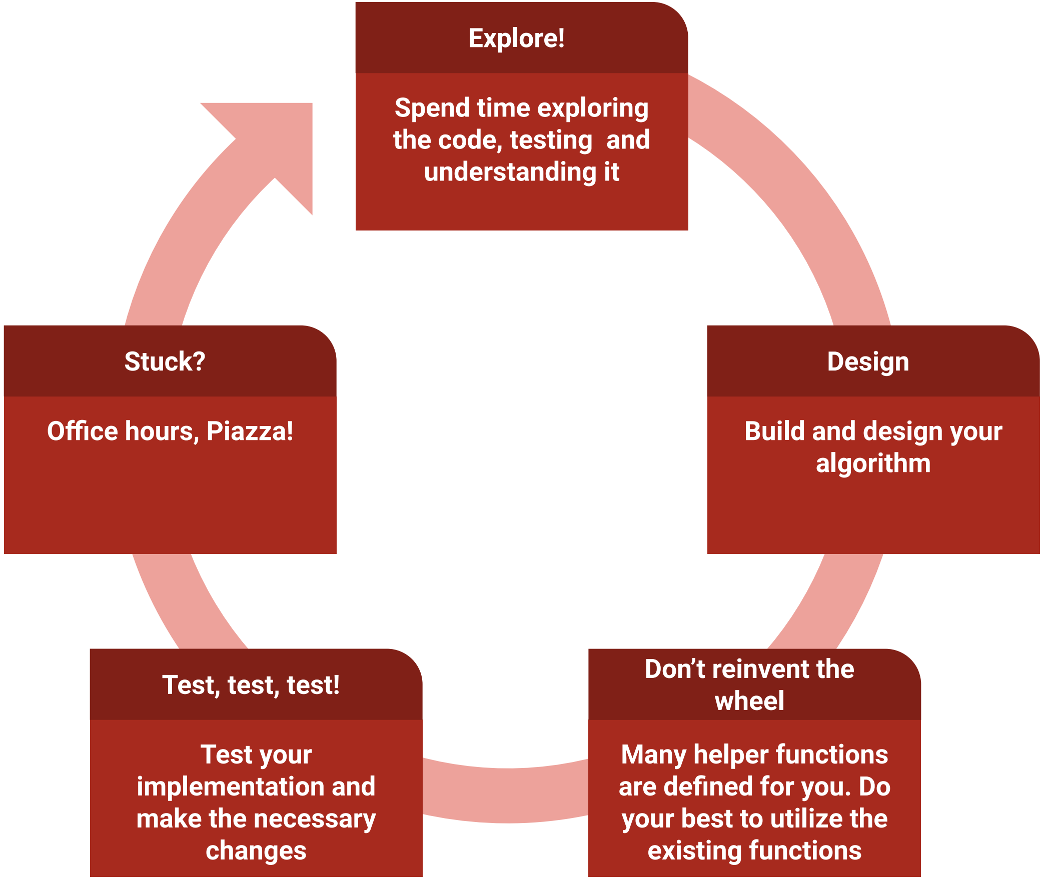
Start Early!

Due on April 18th, 2020

How to start?

Start Early!

Due on April 18th, 2020



Explore!

Spend time exploring the code, testing and understanding it

Design

Build and design your algorithm

Don’t reinvent the wheel

Many helper functions are defined for you. Do your best to utilize the existing functions

Test, test, test!

Test your implementation and make the necessary changes

Stuck?

Office hours, Piazza!