



15-415/615

Database Applications

Spring 2013, Faloutsos

HW3: Indexing

TAs: Danai Koutra
Ekaterina Taralova
Carnegie Mellon University



Overview

- You are given a basic B+ Tree implementation
- Task: extend the B+ tree implementation for new operations



Basic B+ Tree Implementation

- Creates an “inverted index” in the form of a B+ tree
 - key: word, value: document name
- Supports: insert, scan, search, print
- No duplicate keys are allowed
- No support for deletion
- The tree is stored on disk

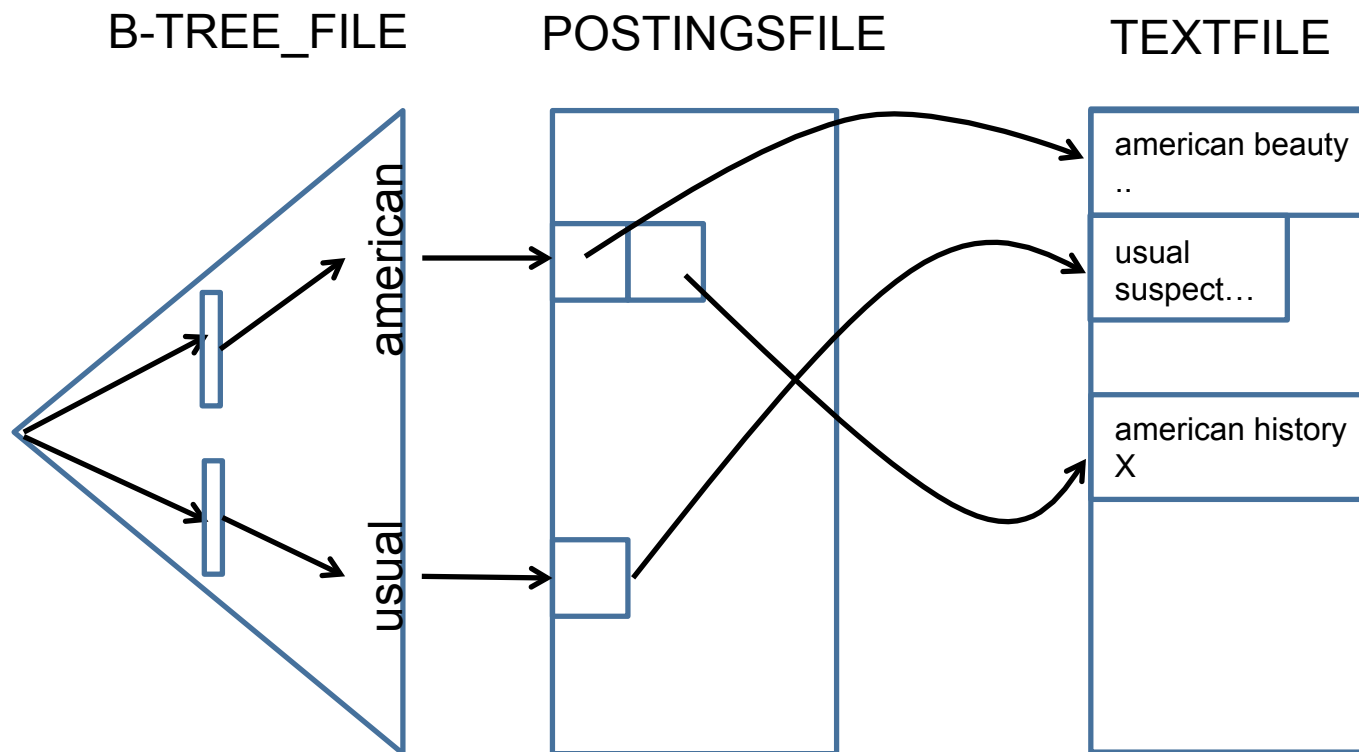


B+ Tree Package

- Folders
 - **DOC**: documentation
 - **SRC**: source code
 - **Datafiles** : sample documents data
 - **Tests**: test files
- B-TREE_FILE, POSTINGSFILE, TEXTFILE, parms are created by the b+ tree.
 - Want a new tree? Delete them

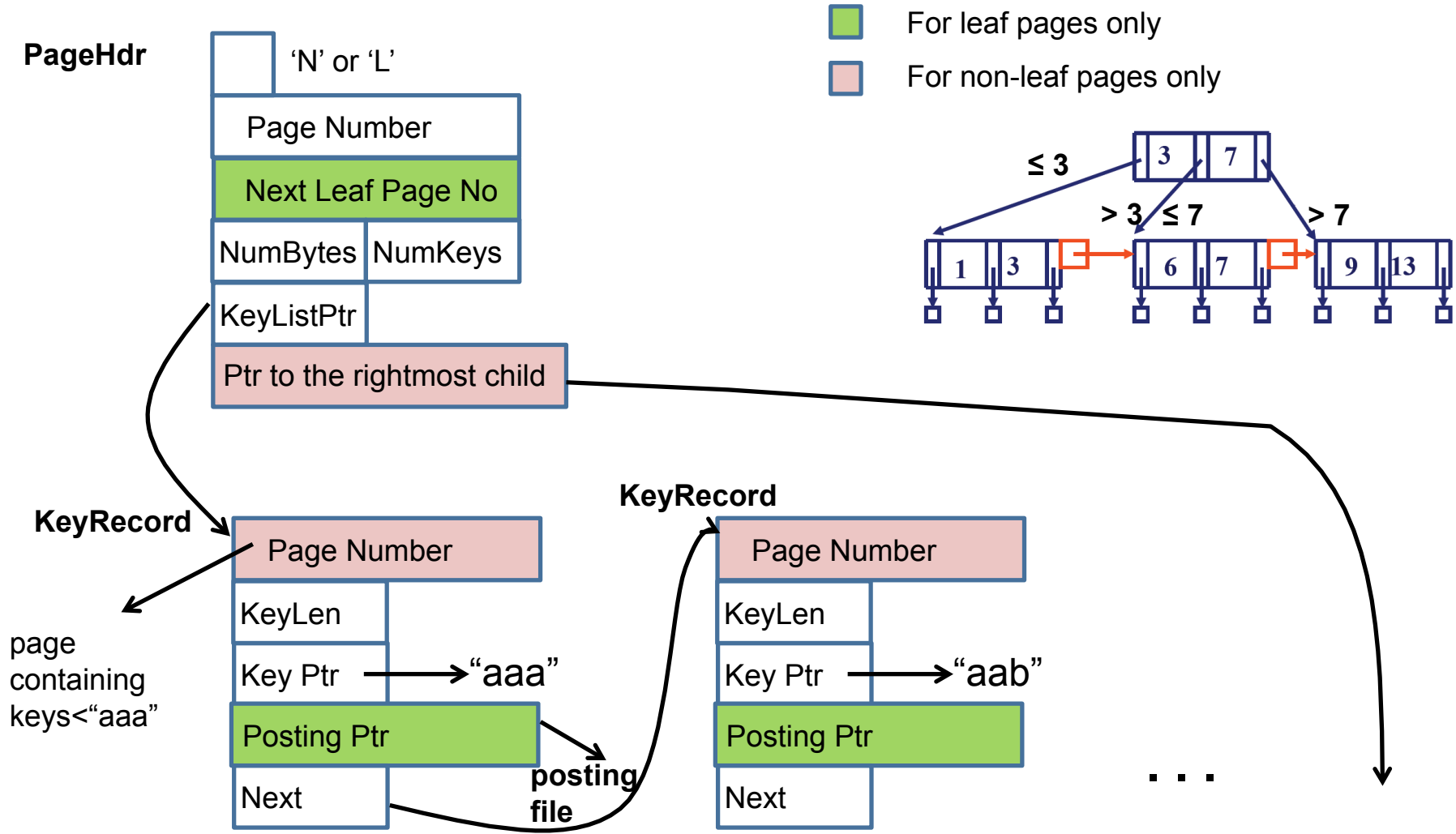


B+ Tree Structure





Structure of a Page (def.h)





Existing Functions

- **C** : print all the keys
- **i** <document_name> : insert the document
 - key: word, value: document_name
- **p** <page_no> : print the info on the page
- **s** <key> : search the key
- **S** <key> : search the key, and print the documents
- **T** : print the tree

Demo



Example code : search

- search.c

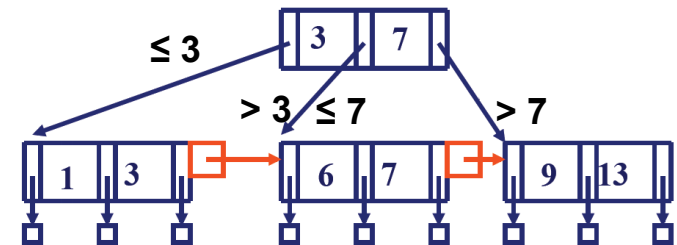
```
16 search(key, flag)
17 char *key;
18 int flag;
19 {
20
21     POSTINGSPTR treesearch();
22     POSTINGSPTR pptr;
23
24     /* Print an error message if strlen(key) > MAXWORDSIZE */
25     if (strlen(key) > MAXWORDSIZE) {
26         printf ("ERROR in \"search\": Length of key Exceeds Maximum Allowed\n");
27         printf (" and key May Be Truncated\n");
28     }
29     if( iscommon(key) ) {
30         printf("\"%s\" is a common word - no searching is done\n",key);
31         return;
32     }
33     if( check_word(key) == FALSE ) {
34         return;
35     }
36     /* turn to lower case, for uniformity */
37     strtolow(key);
38
39     pptr = treesearch(ROOT,key);
40     if( pptr == NONEXISTENT) {
41         printf("key \"%s\": not found\n", key);
42         uqCount++;
43     } else {
44         if(flag) {
45             getpostings(pptr);
46             sqCount++;
47         } else {
48             printf("Found the key!\n");
49         }
50     }
51 }
52 }
```



Example code : search

- **treesearch.c**

```
12 POSTINGSPTR treesearch( pageNo, key)
13 PAGENO pageNo;
14 char *key;
15 {
16     POSTINGSPTR result;
17     PAGENO      ChildPage;
18     struct KeyRecord *KeyListTraverser; /* Pointer to list of keys */
19     struct PageHdr   *PagePtr;
20     PAGENO FindPageNumOfChild();
21     struct PageHdr   *FetchPage();
22
23     PagePtr = FetchPage(pageNo);
24
25     if (IsLeaf(PagePtr)) {
26         result = searchLeaf(PagePtr, key);
27     }
28 }
29 /* The root page contains zero keys */
30 else if ((IsNonLeaf(PagePtr)) && (PagePtr->NumKeys == 0)) {
31     /* keys, if any, will be stored in Page# 2
32     THESE PIECE OF CODE SHOULD GO soon! */
33     result = treesearch(FIRSTLEAFPG,key);
34 } else if ((IsNonLeaf(PagePtr)) && (PagePtr->NumKeys > 0)) {
35     KeyListTraverser = PagePtr->KeyListPtr;
36     ChildPage = FindPageNumOfChild(PagePtr,KeyListTraverser,
37                                   key,PagePtr->NumKeys);
38     result = treesearch(ChildPage,key);
39 }
40 /* -christos-: free the space of PagePtr - DONE! */
41 FreePage(PagePtr);
42 return ( result);
43 }
```

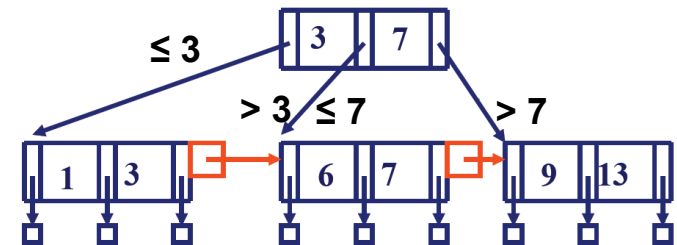




Example code : search

- FindPageNumOfChild.c

```
18 PAGEN0 FindPageNumOfChild(PagePtr,KeyListTraverser,Key,NumKeys)
19 struct PageHdr *PagePtr;
20 struct KeyRecord *KeyListTraverser; /* A pointer to the list of keys */
21 NUMKEYS NumKeys;
22 char *Key; /* Possible new key */
23 {
24     /* Auxiliary Definitions */
25     int Result;
26     char *Word; /* Key stored in B-Tree */
27     int CompareKeys();
28
29
30     /* Compare the possible new key with key stored in B-Tree */
31     Word = KeyListTraverser->StoredKey;
32     (*(Word + KeyListTraverser->KeyLen)) = '\0';
33     Result = CompareKeys(Key, Word);
34
35     NumKeys = NumKeys - 1;
36
37     if (NumKeys > 0) {
38         if (Result == 2) { /* New key > stored key: keep searching */
39             KeyListTraverser = KeyListTraverser->Next;
40             return(FindPageNumOfChild(PagePtr,KeyListTraverser,Key,NumKeys));
41         } else /* New key <= stored key */
42             return(KeyListTraverser->PgNum); /* return left child */
43     } else /* This is the last key in this page */
44     {
45         if ((Result == 1) || (Result == 0)) /* New key <= stored key */
46             return(KeyListTraverser->PgNum); /* return left child */
47         else /* New key > stored key */
48             return(PagePtr->PtrToFinalRtgPg); /* return rightmost child */
49     }
50 }
```





To be implemented

- **f** : print all the distinct keys within the range *d* defined by [word1, word2] in alphabetical order
- **b** : print all the distinct keys within the range *d* defined by [word1, word2] in *reverse* alphabetical order



Hint

- Example
 - make demo
 - Initialize the tree
 - Insert sample document.
 - Perform the 'S american' command.

Demo



Hint

- Sample Tests
 - make test_range
 - Initialize the tree
 - Insert sample document.
 - Perform the ‘f’ command.
 - Store the output to a file.
 - Compare the output to the correct solution (“diff” command)

Demo



Hint

- Sample Tests
 - make test_rangeRev
 - Initialize the tree
 - Insert sample document.
 - Perform the ‘b’ command.
 - Store the output to a file.
 - Compare the output to the correct solution (“diff” command)

Demo



Testing Mechanism

- Correctness
 - output the correct list of words (don't forget to check all the corner cases!)
- Format
 - Make sure the output follows the same format as the sample test solutions.
- **NOTE:** *you are not allowed to store the words (in an array) in alphabetical order, and just print them backwards for command 'b'. Use the structure of the B+ tree.*



Hand-in

- Create a **tar file** of your source code, as well as the makefile.
- **Hard-copy** of a document with the functions that you modified/added.
- Please make sure that the “make” command compiles all the source code without any errors
- Submit **your code** on blackboard.



Questions?

- Come to office hours (5 TAs + instructor)
- Please do not send personal emails.
- Post your questions on blackboard.