CMU SCS

Carnegie Mellon University
15-415/615 Database Applications
Spring 2014,
C. Faloutsos & A. Pavlo

HW3: Indexing

TAs: Alex Beutel; Vagelis Papalexakis
Shen Wang; Ming Zhong

CMU SCS

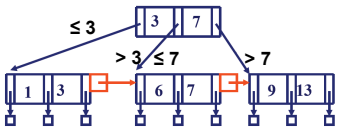
Overview

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

- Today:
  - Brief intro to B+ trees
  - Overview of the code
  - Homework description

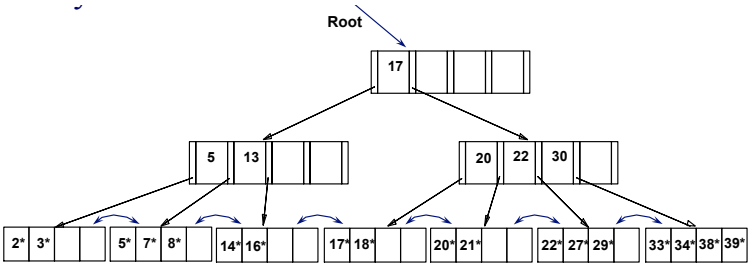
Brief intro to B+ Trees

- N-ary search trees
- Highly used in Databases
- Similar to B-trees
  - Each non-leaf node contains only keys
  - Leaf nodes contain key-value pairs
  - Leaf nodes form a linked list

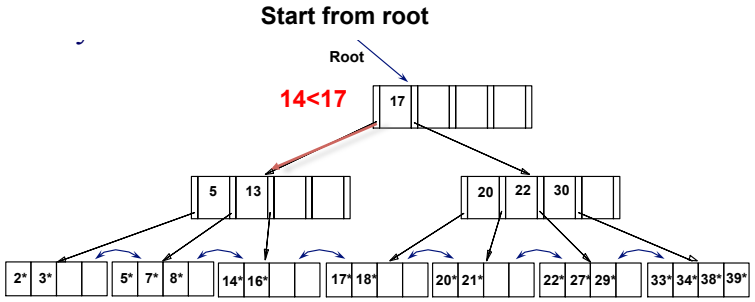


Basic search in B+ Trees

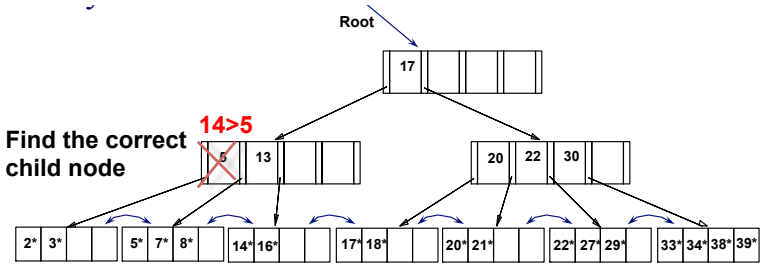
Search for key '14'



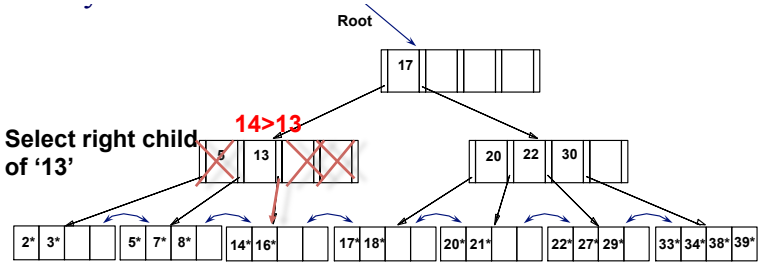
Basic search in B+ Trees



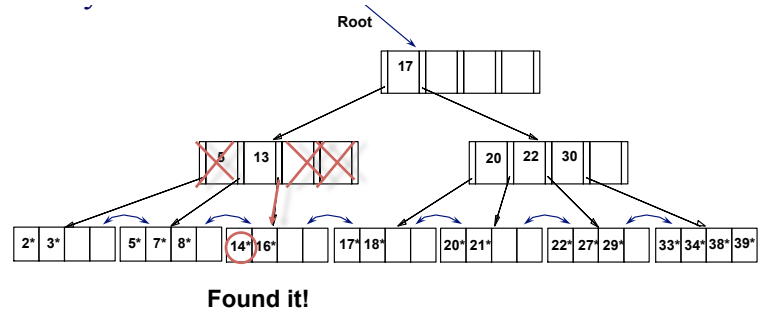
Basic search in B+ Trees



Basic search in B+ Trees



Basic search in B+ Trees



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

- File: **parms**

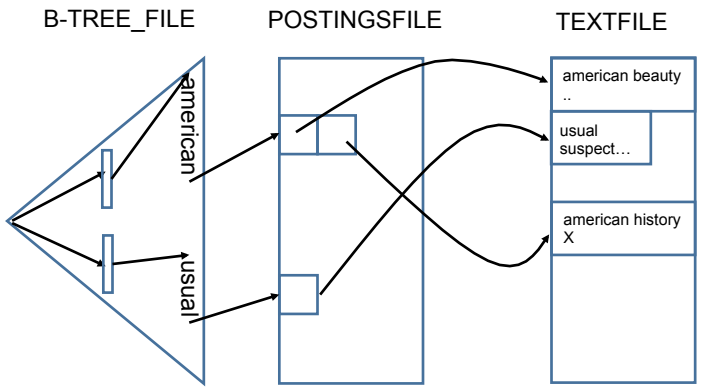
  128 // page size in bytes – leave it as is

  1.618 // ignore it : (expansion rate for postings list)
- Folders
  - **DOC**: documentation
  - **SRC**: source code
  - **Datafiles** : sample documents data
  - **Tests**: test files

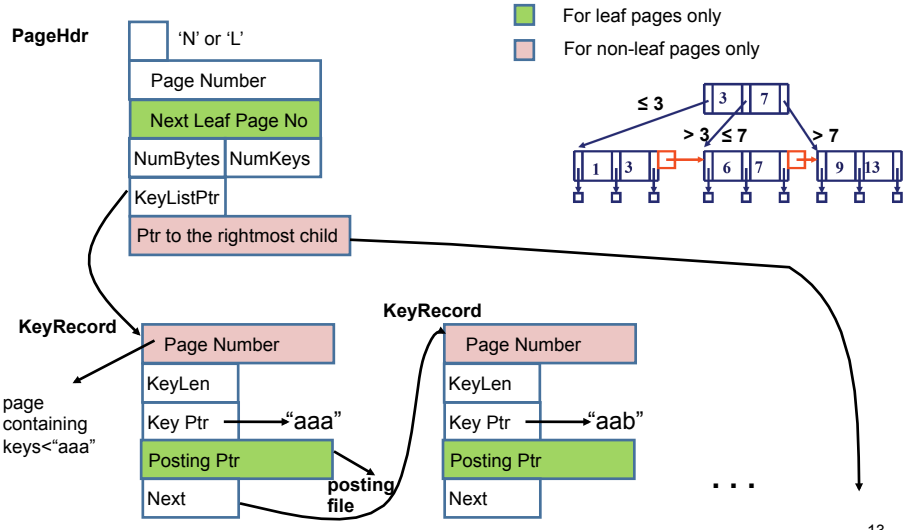
B+ Tree Package

- B-TREE_FILE, POSTINGSFILE, TEXTFILE, 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

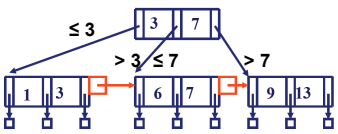
- search.c

```
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

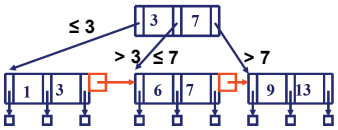
```
12  POSTINGSPTR treesearch( PageNo, key)
13  PAGENO PageNo;
14  char *key;
15  {
16      POSTINGSPTR result;
17      PAGENO          ChildPage;
18      struct KeyRecord *KeyListTraverser;  /* Pointer to list of keys */
19      struct PageHdr   *PagePtr;
20      PAGENO  FindPageNumOfChild();
21      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  PAGENO 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  }
```



In this homework

- implement 2 different types of string search
  - Prefix search:

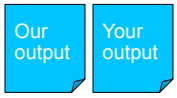
    P organ // matches ‘organism’, ‘organs’, …
  - Substring search (‘M’iddle of word)

    M organ // matches ‘morgana’, ‘organs’,
- Also report
  - count #pages read

Testing Mechanism

Correctness

- Run your code against the given test files:
  - `diff` should be empty



- Make sure you also test on your **own** datasets:
  - Empty B-tree
  - Only one key value in B-tree
  - Root just split
  - ….

Hand-in

- **Hard copy** (in class):
  A. Print answers to the questions
  B. ONLY new/changed code (save the trees ☺)
- **Electronic copy** (Blackboard):
  A. btree_andrewId.tar file with only the necessary files & the makefile.

     % tar xvf; make # should run your code on

     # sample scripts

Hand-in

FYI, we will use scripts to grade, thus please follow **carefully** the instructions for the output format!

Questions?

- Come to office hours (4 TAs + 2 instructors)
- Post your questions on blackboard.