REPORT

Advanced Data Structure - Project

NAME: Vikrant Sagar
UFID : 1505 - 1993
EMAIL: coderz@ufl.edu

TOOLS

Compiler: GNU g++
Version: 4.4.3
System: Linux
Github Link: https://github.com/hellcoderz/ADS/tree/master/dict

INSTRUCTIONS

Compiling:
> *make*
Running:
> *./dictionary -r s order*
and
> *./dictionary -u filename*

PROGRAM STRUCTURE

I have created doxygen documentation of my entire source code so i think it will far more informative than simple function prototypes. All documentation is arranged as namespace, classes and files. The file "program_structure.html" will point to namespaces page, but you can go to classes and file list also. Click on any class to view its function and short and long description. Click on any function to view its position in the source code.
Instruction: Double Click on *"program_structure.html"* file.

EXPERIMENTS

Optimal Btree order:

BTree Order	Insert Time(ms)	Search Time(ms)
10	671	232
15	537	205
25	464	202
30	467	209
50	494	257
90	548	335
150	700	507

Since the values in Red are almost equal most of the time. So i will choose ORDER = 30 as an optimal order.

Optimal Hash Size:

Hash Size(s)	AVL Hash (Insert)	AVL Hash (Search)	BTree Hash (Insert)	BTree Hash (search)	RB Hash (Insert)	RB Hash (Insert)
3	529	186	597	273	385	240
11	541	196	622	282	395	259
101	668	291	701	330	517	379

s = 3 seems good choice for the above data structures.

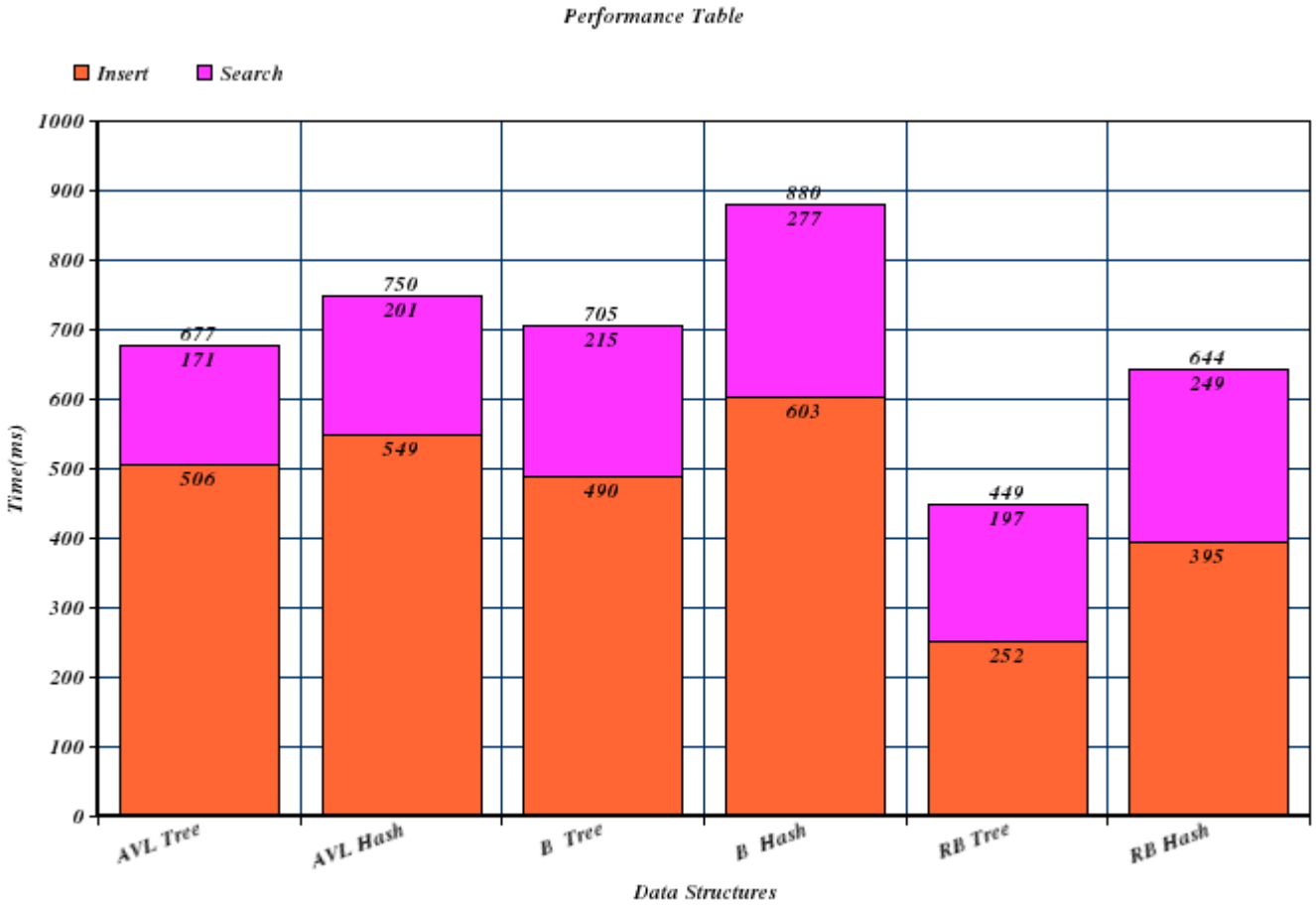
Expected Results for Order = 30 and s = 3

DATA STRUCTURE	INSERT(ms)	SEARCH(ms)
AVL Tree	I1	S1
AVL Hash	I3 (< I1)	S3(< S1)
B+ Tree	I5(<I2)	S5(<S2)
B+ Hash Tree	I6(<I5)	S6(<S5)
RB Tree	I2 (<I1)	S2(<S1)
RB Hash Tree	I4(< I2)	S4(<S2)

Experiment Results for Order = 30 and s = 3

DATA STRUCTURE	INSERT(ms)	SEARCH(ms)
AVL Tree	506	171
AVL Hash	549	201
B+ Tree	490	215
B+ Hash Tree	603	277
RB Tree	252	197
RB Hash Tree	395	249

Graph:



RECOMMENDATIONS:

- If n=1000000 the for the worst case i will use B+ Hash Tree because its total height is less and using hash table it becomes lower so for worst case search it will be good Data Structure.
- For the expected case RB Tree comes out to be a good Data Structure because experimental result shows it takes lower time for both search and insert.
- If we also wanted to support nearest match search apart from normal search then i will go with B+ Tree because all nearest searches may be present in the same node. Apart from this if our requirement is just to find if the element is present or not then we can use Trie data structure to find nearest matches.