Project Final Report

To calculate data:

Used tablesize (global variable) and given input size (the amount of books to be read) to calculate load factor.

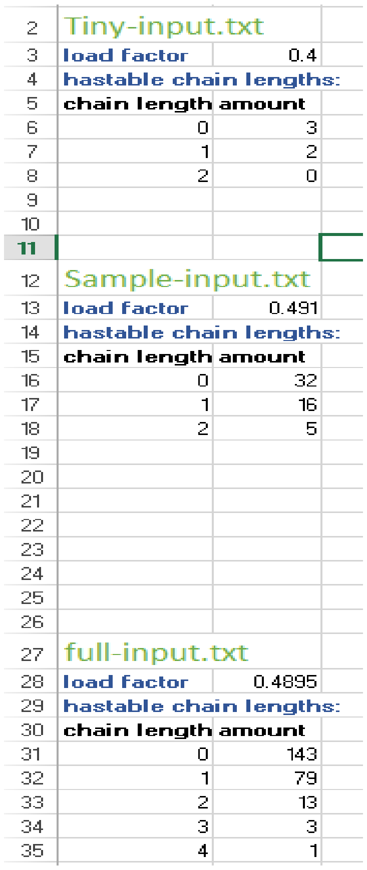
Printed the size of every chain in hashtable and manually collected data (how many of them were 0,1,2,3)

Altered the BST struct to include a number (int right_tree, int left_tree) in which every time a book was added to right/left subtree, this value was incremented by one. Each time a book was deleted from the tree, these were decreased by one. Printed total (right + left + 1), left and right amounts for each tree in my genre array at the end of the program.

Qestions:

(a) Compute the length of each "chain" in your hash table. Report the number of chains of each length.

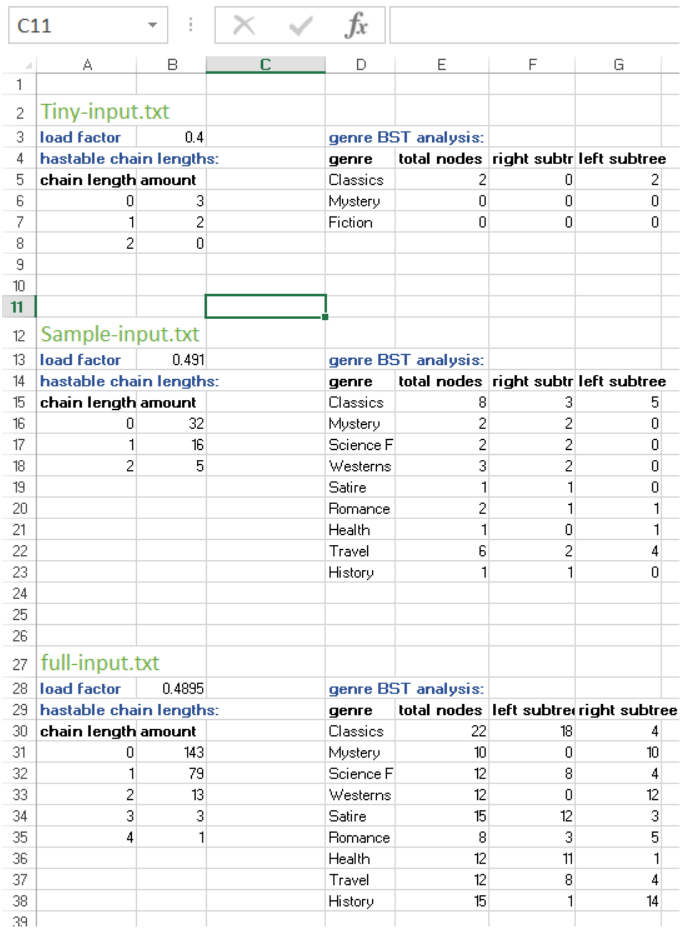
(b) Compute the load factor of your hash table.



Tiny-input.txt	
load factor	0.4
hastable chain lengths:	
chain length	amount
0	3
1	2
2	0
Sample-input.txt	
load factor	0.491
hastable chain lengths:	
chain length	amount
0	32
1	16
2	5
full-input.txt	
load factor	0.4895
hastable chain lengths:	
chain length	amount
0	143
1	79
2	13
3	3
4	1

(c) For each BST corresponding to a genre, compute the total number of nodes in the tree, the

number of nodes in the left and right subtrees, and the number of leaves at each height in the tree.



C11

	A	B	C	D	E	F	G
1							
2	Tiny-input.txt						
3	load factor	0.4		genre BST analysis:			
4	hastable chain lengths:			genre	total nodes	right subtr	left subtree
5	chain length	amount		Classics	2	0	2
6	0	3		Mystery	0	0	0
7	1	2		Fiction	0	0	0
8	2	0					
9							
10							
11							
12	Sample-input.txt						
13	load factor	0.491		genre BST analysis:			
14	hastable chain lengths:			genre	total nodes	right subtr	left subtree
15	chain length	amount		Classics	8	3	5
16	0	32		Mystery	2	2	0
17	1	16		Science F	2	2	0
18	2	5		Westerns	3	2	0
19				Satire	1	1	0
20				Romance	2	1	1
21				Health	1	0	1
22				Travel	6	2	4
23				History	1	1	0
24							
25							
26							
27	full-input.txt						
28	load factor	0.4895		genre BST analysis:			
29	hastable chain lengths:			genre	total nodes	left subtre	right subtree
30	chain length	amount		Classics	22	18	4
31	0	143		Mystery	10	0	10
32	1	79		Science F	12	8	4
33	2	13		Westerns	12	0	12
34	3	3		Satire	15	12	3
35	4	1		Romance	8	3	5
36				Health	12	11	1
37				Travel	12	8	4
38				History	15	1	14
39							

//ALL OF THESE QUESTIONS ARE ANSWERED ON THE EXCEL SHEET PROVIDED AS WELL

// WAS UNABLE TO COMPUTE NUMBER OF NODES AT EACH LEVEL☹

Based on your computations, answer the following questions:

(a) Is your hash table large enough to support your store inventory?

Yes, hashtable if more than enough size to support the inventory. As seen, most spaces in hashtable are left empty (with a lot of 0 sized chains). Also, the load factor represent this as well (<.5 for all)

(b) Is the hash function doing a good job at uniformly distributing titles into the table?

Really good job. Most chain sizes are 1 and 0, very few collisions happen. Max number of collisions is 4, and happened once or twice for full input.

(c) How well balanced are your BSTs in terms of their height?

BST's are not very well balanced, as seen in the data provided. Some are completely left or right sided trees (with no elements in right and left subtrees, respectively). I was not able to calculate number of nodes at each height.

(d) Would you alter the size and/or structure of the data structures used for processing queries or updates?

I would consider reducing the size of the hashtable a little bit in order to have better memory-size complexity in my program. Maybe 1.5*m instead of 2*m; would probably still work. However, hashtable does its job really well.

Since the BST's are quite unbalanced, it might be better to implement and AVL/2-3 tree for processing this kind of data. That was, the BST will be balanced. However, this unbalanced tree (since we have our has table) does not really affect the runtime of our queries, except probably for the algorithm of deleting a BST node if we turn out of stock for that book.