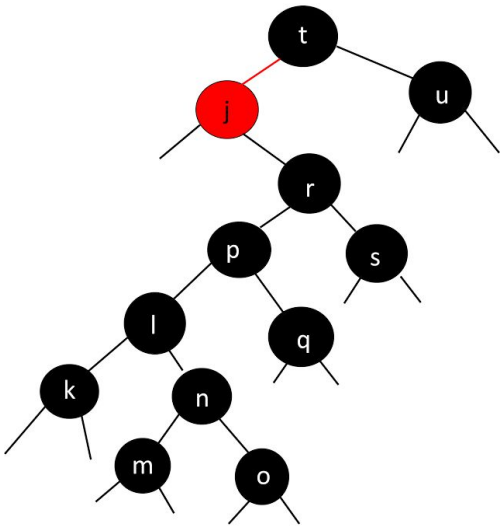
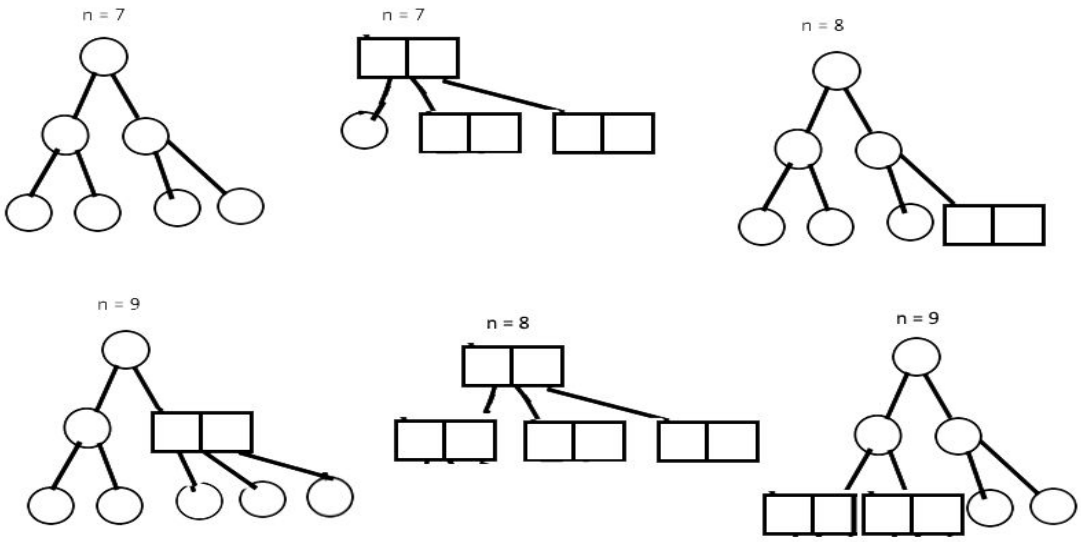
Skúli Arnarsson skulia15
Darri Valgarðsson darriv15

(1) (15%) (Problem 3.3.16) Show the result of inserting the letter n into the red-black BST shown on page 450 (only the search path is shown, and you need to include only these nodes in your answer).

(2)



(15%) (Problem 3.3.5, shortened) Draw all structurally different 2-3 trees for N = 7, 8, and 9. (The trees for N from 1 to 6 are shown on p. 449)



(3) (10%) (Problem 3.4.10) Insert the keys E A S Y Q U T I O N in that order into an initially empty table of size M=16 using linear probing. Use the hash function 11 k % M to transform the k-th letter of the alphabet into a table index (so, 'A' = 0). Redo this exercise for M=16.

M = 16

E = 11 * 4 % 16 = 12
A = 11 * 0 % 16 = 0
S = 11 * 18 % 16 = 6
Y = 11 * 24 % 16 = 8
Q = 11 * 16 % 16 = 0
U = 11 * 20 % 16 = 12
T = 11 * 19 % 16 = 1
I = 11 * 8 % 16 = 8
O = 11 * 14 % 16 = 10
N = 11 * 13 % 16 = 15

[0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16]
[A	Q	T				S		Y	I	O		E	U		N	]

M = 10

E = 11 * 4 % 10 = 4
A = 11 * 0 % 10 = 0
S = 11 * 18 % 10 = 8
Y = 11 * 24 % 10 = 4
Q = 11 * 16 % 10 = 6
U = 11 * 20 % 10 = 0
T = 11 * 19 % 10 = 9
I = 11 * 8 % 10 = 8
O = 11 * 14 % 10 = 4
N = 11 * 13 % 10 = 3

[0	1	2	3	4	5	6	7	8	9	10]
[A	U		N	E	Y	Q	O	I	T	S]

(4) (10%) Suppose that the keys A through G, with the hash keys given below, are inserted in some order into an initially empty table of size 7 using linear probing (M=7, no resizing).
[Corrected 6 Oct]

key	A	B	C	D	E	F	G
hash	2	0	5	4	4	4	2

Which of the following (more than one might apply) could not possibly result from inserting these keys?

(a) B E A G D F C - Works
(b) C F A G D E B - Doesn't work
(c) F B G A E C D - Works
(d) F C B G A D E - Doesn't work

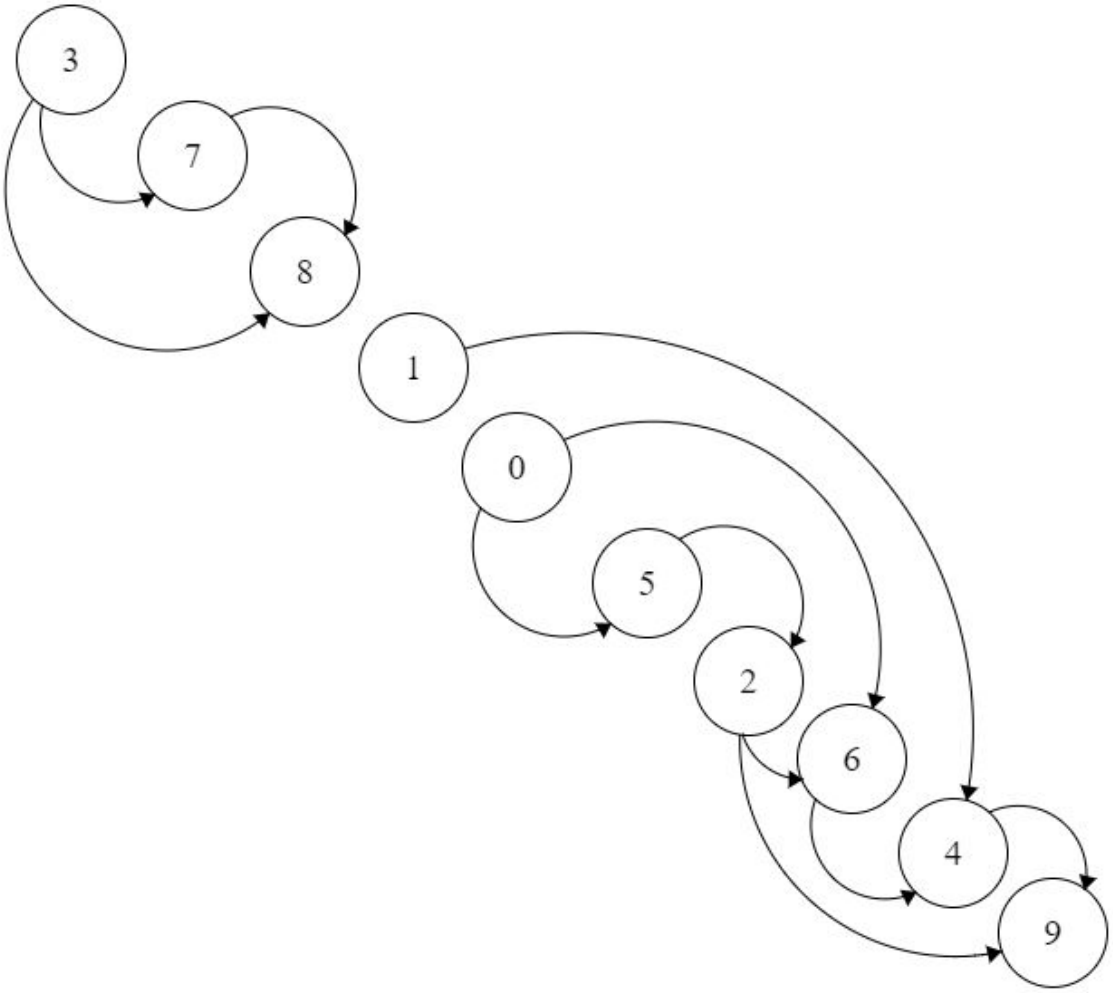
Give the minimum and the maximum number of probes that could be required to build a table of size 7 with these keys, and an insertion order that justifies your answer.

minimum = 14 B A G D E F C -> B	C	A	G	D	E	F
maximum = 14 B A D C G E F -> B	F	A	G	D	C	E

(5) (25%) Modify BreadthFirstPaths.java to compute the number of shortest paths between two given vertices v and w in a given digraph. This replaces the method boolean hasPathTo(inv v) with int nrOfPathsTo(inv v). Hint: Note that if vertex w is of distance k from v, and it is adjacent to vertices a, b and c of distance k −1 from v, then each shortest v − w path runs through one of a, b, and c. [Updated 5 Oct to refer to digraphs]

Submission on Mooshak :)

(6) (10%) [5 Oct] Give a topological sort of the digraph containing the edge list: 3→7 1→4 7→8 0→5 5→2 3→8 2→9 0→6 4→9 2→6 6→4.



(7) (15%) [5 Oct] Run DepthFirstOrder on the digraph t11.txt attached. Give the preorder, postorder, and reverse postorder obtained. (Alternatively, you can answer the original question of running the Kosaraju-Shamir algorithm for strongly connected components on the graph.)

Preorder: 0 5 1 2 4 10 6 3 7 8 9
Postorder: 10 6 4 2 1 5 0 3 9 8 7
Reverse postorder: 7 8 9 3 0 5 1 2 4 6 10