**CPSC 6109:** [**Advanced**](https://colstate.view.usg.edu/d2l/lp/ouHome/home.d2l?ou=1218642) **Algorithms**

**Spring 2018**

**Assignment #5**

**Student: Lu Lin**

**Due: 11:59 PM Tuesday, March 13**

Do the following exercises/problems. Each problem is worth 50 points with a total of 100 points.

1. Show all possible legal B-trees of minimum degree 2 that represent the set

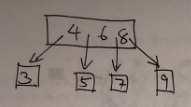
{3, 4, 5, 6, 7, 8, 9}.

Solution:

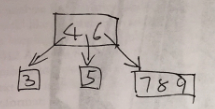
We know that every node except the root must have at least t – 1 = 1 keys, and at most 2t – 1 = 3 keys. Also the leaves stay in the same depth. We have a 2-3-4 tree, every internal node then has either 2, 3, or 4 children.

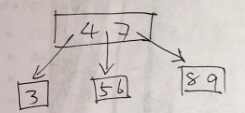
There are 9 possible B-trees for given problems:

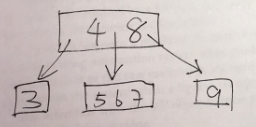
when number of keys = 3 we have one B-tree:

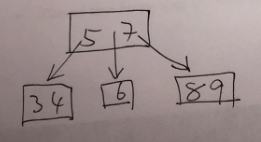


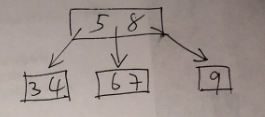
when number of keys = 2 there are six B-trees:

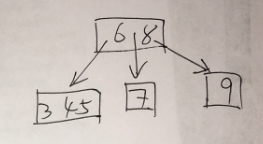




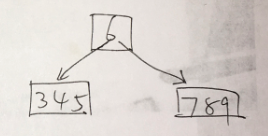


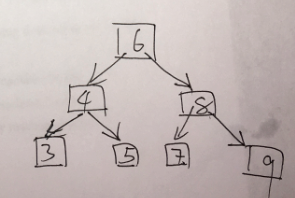






when number of keys = 1 there are two B-trees:





1. Problem **18.2-3** on page 497. Explain how to find the minimum key stored in a B-tree and how to find the predecessor of a given key stored in a B-tree.

Solution:

1. Finding the minimum key in a B-tree is to start from the root, traverse down the left most child until reach a leaf of that root, return the first key as the minimum key.

B-TREE-FIND-MIN(x)

// x is a node

IF (x == NIL)

RETURN NIL

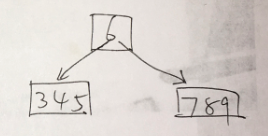
ELSEIF (x.leaf) // x is a leaf

RETURN x.key1

ELSE

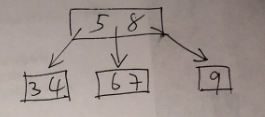
DISK-READ(x.c1)

RETURN B-TREE-FIND-MIN(x.c1)

1. There are a few situations when finding a given key k’s predecessor:
2. k was in a leaf node and is NOT the leftmost key in the node, then return the left key of k in the node simply. Take problem 1 as example, 

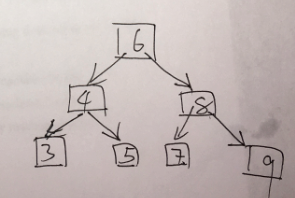
if k = 5 then predecessor is the left key of 5, which is 4.

1. k was in a leaf node and is the leftmost key in the node, go up the tree till find a key in the parent node at the left part of link/pointer, then return this key as result. If you reach the root so the left part key can’t be found, it means that now k is the minimum key, then return null as result. For example,



if k = 6 then find a key in the parent node at the left part of pointer, which is 5; if k = 3 and I to up the tree till reach the root and no keys found at the left part of pointer, which means k is the minimum key and its predecessor is null.

1. k was in an inner node. Traverse by the left child node of x (k is key of x), if reach a leaf then return the rightmost key in the leaf. If reach a non-leaf node then traverse down the rest of the tree by the right child till reach a leaf node, and then return the rightmost key as result. For example,



if k = 4, traverse by left child node, which is a leaf node, then return its rightmost key 3; if k = 6, traverse by the left child node to x = 4, which is a non-leaf node; traverse down by the right child node of x = 4 till reach a leaf node, then return its rightmost key as result, which is 5.

B-TREE-FIND-PREDECESSOR(x, i)

// x is a node, i is index of keys, r is root, T is B-tree

IF (x.leaf AND i > 1)

RETURN x.keyi-1 // return the left key

ELSEIF (x.leaf AND i = 1)

r = x

WHILE (1)

IF r.p == NIL

RETURN NIL // now r.key1 is the minimum key

j = 1

DISK-READ(r.p.c1)

WHILE (r.p.cj != x)

j = j + 1

DISK-READ(r.p.cj)

IF j == 1

r = r.p

ELSE

RETURN r.p.keyj-1

ELSEIF (!x.leaf)

DISK-READ(x.ci)

RETURN B-TREE-FIND-MAX(x.ci)

B-TREE-FIND-MAX(x)

IF x == NIL

RETURN NIL

ELSEIF (x.leaf)

RETURN (x, x.n)

ELSE

DISK-READ(x.cx.n+1)

RETURN B-TREE-FIND-MAX(x.cx.n+1)