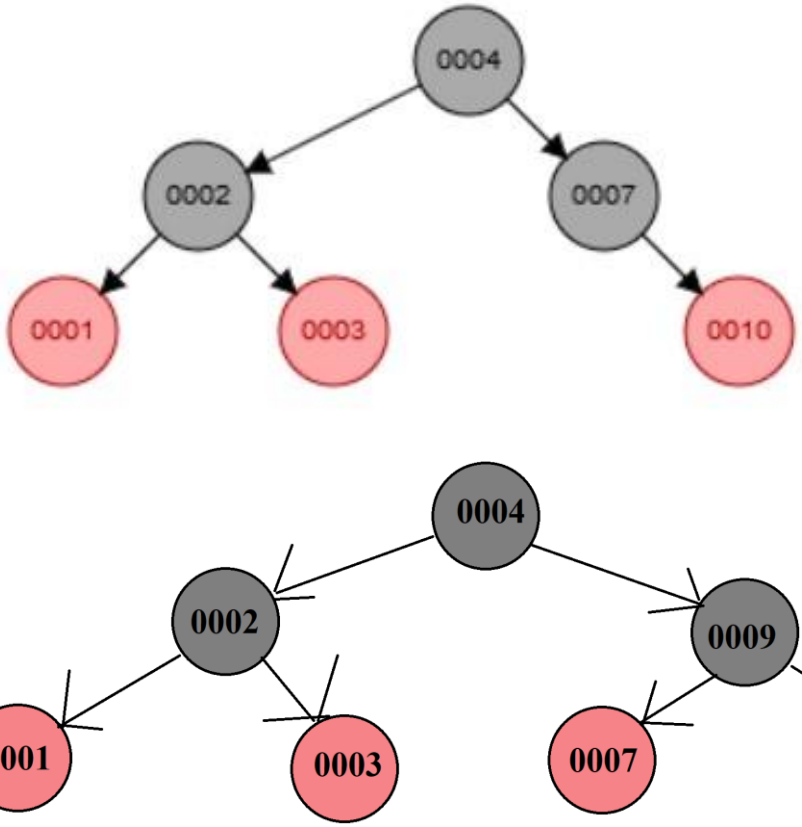
CptS 223 – Advanced Data Structures in C++

Written Homework Assignment 4: Red-black Trees, B+ Trees

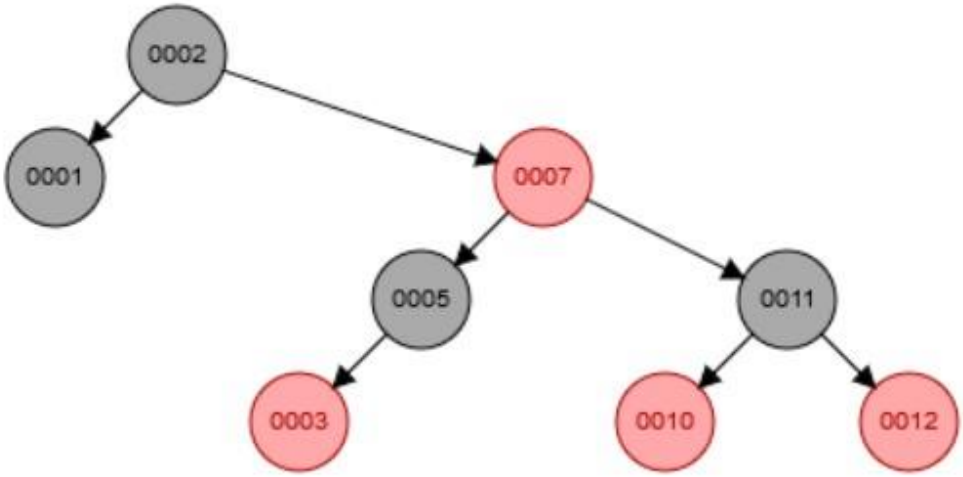
I. Problem Set:

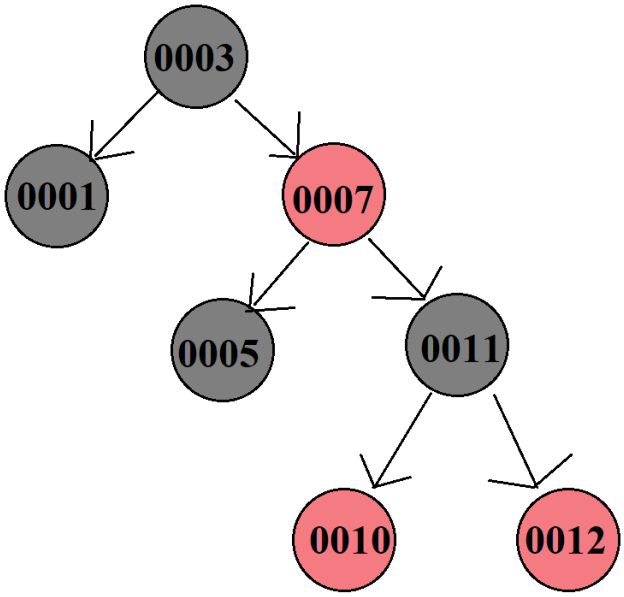
1. **(10 pts)** Insert the value "9" into the following Red-Black tree; draw the result. Use Double-circle to denote red nodes and single circle to denote black nodes.



(done in paint)

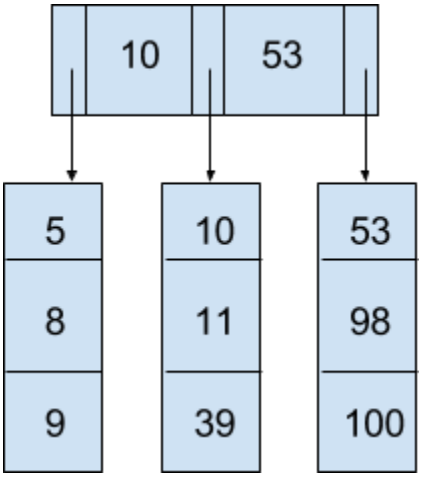
2. **(10 pts)** Delete the value "2" from the following Red-Black tree; draw the result. Use Double-circle to denote red nodes and single circle to denote black nodes.



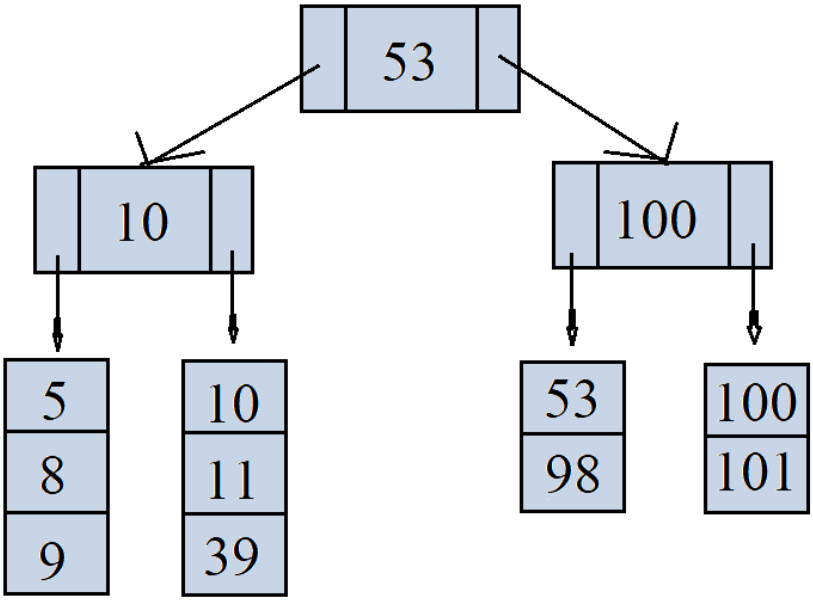


(done in paint)

3. **(30** Given the following B+ tree (M = 3, L = 3):

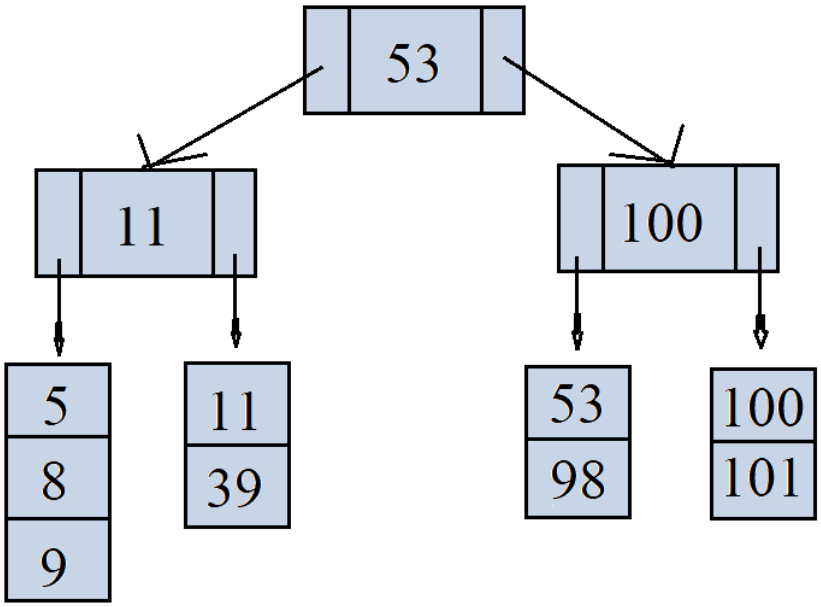


a. **(15 pts)** Insert data item 101 into the tree and draw the resulting B+ Tree:



(done in paint)

b. **(15 pts)** Based on the tree resulting from part (a), now remove 10 and draw the new tree:



(done in paint)

4. **(50 pts)** We are going to design our B+ Tree to be as optimal as possible for our hard drives. We want to keep the tree as short as we can, and pack each disk block in the filesystem as tightly as possible. We also want to access our data in sorted order for printing out reports, so each leaf node will have a pointer to the next one. See figure #1 on next page for a visualization of our tree.

   CPU architecture: Intel Xeon with 64 bit cores (pointer size = 64/8 = 8 bytes)
   Filesystem: Ext4 with 8KB (8096 byte) blocks
   The customer records are keyed by a random UUID of 64 bits Customer's Data record definition from the header file:

```
#include
<uuid>
struct
CustomerDat
a
{
     uuid_t uuid;         // Customer 64 bit key
     char[32] name;       // Customer name (char is 1 byte
each)
     uint32_t ytd_sales; // Customer year to date sales
};
```

a. **(15 pts)** Calculate the max number of children in each internal node (M) for our B-tree:

   M = floor[(B+K)/(4+K)]
   B = 8192bits and K = 8bits

   So plugging these in we get:
   M = floor[(8192+8)/(4+8)]
   M = floor(683.3333…)
   M = 683 bits

b. **(15 pts)** Calculate the max number of data records in each B-tree leaf node (L) to keep the list of leaf nodes:

L = floor[B/D]
B = 8192 bits
D = 32bits + (64bits/8bits) + 4bits
(Name) (pointer) (unsigned int)
D = 44 bits

So we get:
L = floor[8192/44]
L = floor[186.18181…]
L = 186

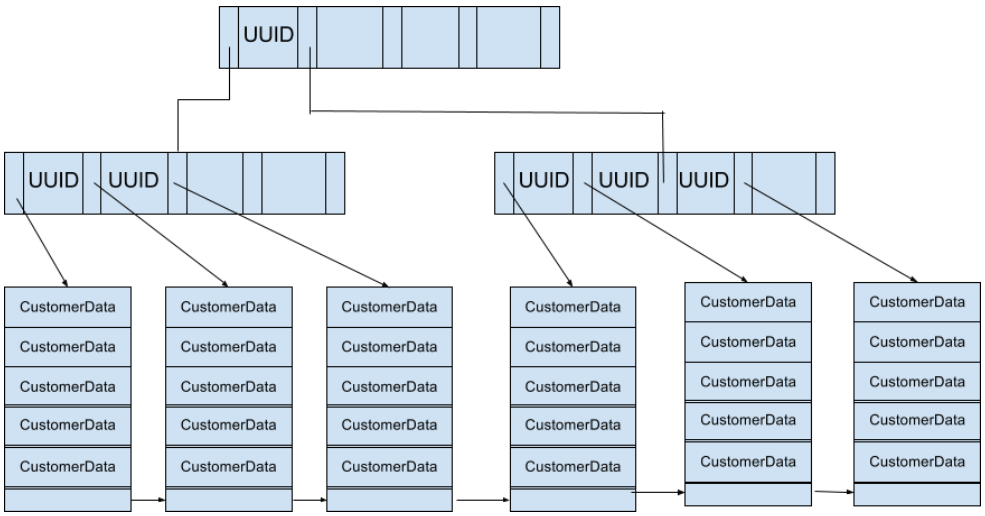


Figure #1: Visualization of our B+ Tree of height 2, customer data records, and pointers between the leaf nodes. M and L shown in this tree are not the real M and L in this question.

c. **(10 pts)** How tall (on average) will our tree be (in terms of M) with N customer records?

Height = $Log_M X$ where X = ceil(N/L)
M = 683
L = 186

Height = Log_{686}ceil(N/186)

d. **(5 pts)** If we insert 30,000 CustomerData records, how tall will be tree be? (You don't have to calculate the result of log)

Height = Log_{686}ceil(30,000/186)
Height = Log_{686}162

e. **(5 pts)** If we insert 2,500,000 customers how tall will the tree be? (You don't have to calculate the result of log)

Height = Log_{686}ceil(2,500,000/186)
Height = Log_{686}13,441

II. Submitting Written Homework Assignments:

1. On your local file system, create a new directory called HW4. Move your HW4.pdf file into the directory. In your local Git repo, create a new branch called HW4. Add your HW4 directory to the branch, commit, and push to your private GitHub repo created in PA1.
2. Do not push new commits to the branch after you submit your link to Canvas otherwise it might be considered as late submission.
3. Submission: You must submit a URL link of the branch of your private GitHub repository to Canvas.

III. Grading Guidelines:

This assignment is worth 100 points. We will grade according to the following criteria:

- See above problems for individual point totals.