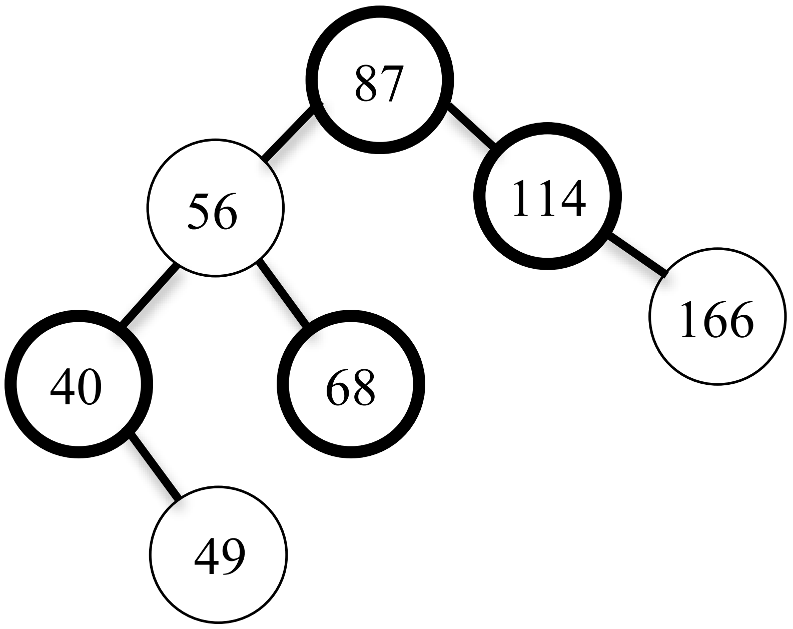
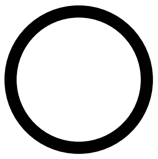
CMSC 341 Data Structures Midterm II Review

These questions will help test your understanding of the material discussed in class and in the text. These questions are only a study guide. Questions found here may be on your exam, although perhaps in a different format. Questions NOT found here may also be on your exam.

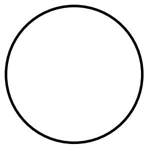
Red-Black Tree Review

1. Define ***Red-Black tree***. List all Red-Black tree properties
2. Define the ***black height*** of a node, *x*.
3. What is the "Big-Oh" performance (in terms of the number of nodes in the tree) for the operations **find, insert** and **remove** for a red-black tree in the best, worst and averages cases?
4. What property of red-black trees is most significant in explaining the "Big-Oh" performance for the operations **find, insert** and **remove**?
5. Prove that in any red-black tree with root x, there are at least $n = 2^{bh(x)} - 1$ internal nodes where $bh(x)$ is the black-height of x.
6. Prove that in any red-black tree, at least half the nodes on any path from the root to a leaf must be black.
7. Prove that in any red-black tree, no path from any node, N, to a leaf is more than twice as long as any other path from N to any other leaf.
8. Prove that if a black node has just one child, that child must be red.
9. Show the tree that results from inserting the values **2, 1, 4, 5, 9, 3, 6, 7** into an initially empty red-black tree. Show the tree after each insertion.
10. Given the following Red-Black Tree, show the tree that results after deleting the node with value **68** using bottom-up deletion.





Represents a BLACK node



Represents a RED node

Priority Queue Review

1. Define the following terms
    a. priority
    b. priority queue
    c. min binary heap
    d. partial ordering
    e. null path length in a binary tree
    f. leftist binary tree
    g. leftist heap
2. Insertion and deletion (of the minimum element) in a min binary heap are $O(lg\ n)$ on average. Explain why this is so.
3. Finding the minimum element in a min binary heap is $O(1)$ in the worst case. Explain why this is so.
4. Although a binary heap is conceptually a binary tree, it can be implemented as an array. Explain why this is so.
5. In a min binary heap with N elements, what is the range of indices in which the largest element will be found?
6. Describe, in English, an algorithm to find the largest element in a min binary heap. What is the asymptotic worst-case performance of your algorithm?
7. Assume that the array representing a min binary heap contains the values 2, 8, 3, 10, 16, 7, 18, 13, 15. Show the contents of the array after inserting the value 4.
8. Assume that the array representing a min binary heap contains the values 2, 8, 3, 10, 16, 7, 18, 13, 15. Show the contents of the array after deleting the minimum element.
9. Show the array representing the min binary heap constructed using the initial values 18, 2, 13, 10, 15, 3, 7, 16, 8.
10. Prove that the largest element in a min binary heap is a leaf.
11. Prove that a complete binary tree is a leftist tree.
12. Prove for any leftist tree with N nodes, the number of nodes, R, on the rightmost path to a non-full node is given by $R \leq lg(N + 1)$
13. Given the drawing of a binary tree, determine if the tree is a leftist tree and if it is a leftist heap. Give reasons why or why not.

14. Given the drawings of the two leftist heaps, draw the leftist heap that results from merging them.
15. Describe how to perform the **findMin, insert,** and **deleteMin** operations on a leftist heap.
16. Describe a method for constructing a leftist heap from an initial set of *N* values. Your algorithm must run in *O*(*N*) time.

B-Tree Review

1. Define B-Tree. List all B-Tree properties.

2. What does it mean to say a B-Tree is **order M**?

3. When describing a B-Tree, what does L represent?

4. Give the pseudo-code for finding a particular element in a B-Tree of order M.

5. Given the drawing of a B-Tree, show the new B-Tree after inserting a given element.

6. Given the drawing of a B-Tree, show the new B-Tree after deleting a given element.

7. Draw a valid B-Tree with M = 4 and L = 3 containing the integer values 1 – 25.

8. Show the result of inserting the elements 1, 3, 5, 7, 9, 11, 6 into an initially empty B-Tree with M = 3 and L = 3. Show the tree at the end of each insertion.

9. Given the following characteristics of an external storage problem, design a suitable B-Tree (i.e. calculate appropriate values of M and L).
    a. The number of items to be stored
    b. The size (in bytes) of the key for each item
    c. The size (in bytes) of each item
    d. The size (in bytes) of a disk block

10. What is the minimum and maximum number of leaves in a B-Tree of height h = 2 when M = 3?

11. The average case performance of the dictionary operations insert, find and delete is *O*(*lg N*) for balanced binary search trees like Red-Black trees. In a B-Tree, the average asymptotic performance for the dictionary operations is $O(log_M N)$ where M is the order of the B-Tree. Discuss the following.

    a. When M = 2, do the B-Tree and the RB Tree have equivalent asymptotic performance for the dictionary operations? Are there advantages of one over the other?

b. B-Tree height is proportional to $log_M\ N$ indicating that for a given N, a B-Tree of higher order will be shorter than one of lower order. Is this true? If so, why not always choose a very high value for M since the average asymptotic performance of the dictionary operations is in $O(height)$.

c. B-Trees find their greatest utility when data are stored externally (on disk rather than in memory). Why is this so?

Hashing Review

1. What is a hash function? Name two desirable properties of a hash function.
2. Define **collision** in a hash table.
3. What is the **clustering** problem in hash tables?
4. Describe the **division method** for generating hash values.
5. Describe the **multiplication method** for generating hash values.
6. Define **Fibonacci hashing**.
7. Describe the **separate chaining** collision resolution method.
8. Describe the **open addressing** collision resolution method.
9. Given a hash table of size 13, show the contents of your hash table after inserting the values {8, 2, 7, 18, 15, 19, 23, 15, 20, 16} using open addressing with linear probing ($f(i) = i$) for collision resolution
10. Repeat question 9, using open addressing with quadratic probing ($f(i) = i^2$) for collision resolution.
11. Repeat question 9 using separate chaining for collision resolution.
12. The average time performance of the insertion and searching operations on a hash table is $O(1)$, which is much better than the performance of a binary search tree for the same operations. Given this wonderful performance of hash tables as compared to binary search trees, when would you want to use a binary search tree instead of a hash table?
13. In a hash table using open addressing with linear probing, the average number of probes for successful search, S, and unsuccessful search (or insertion), U, are

$$S \approx 12\ (\ 1 +\ 11-\lambda\ ) \qquad\qquad U \approx 12\ (\ 1+\ 1(1-\lambda)2\ )$$

where λ is the load factor of the table. Suppose you want a hash table that can hold at least 1000 elements and you want successful searches to take no more than 4 probes on average.

   a. What is the maximum load factor you can tolerate in your hash table?
   b. If the table size must be prime, what is the smallest table size you can use?
   c. Based on your answers to (a) and (b), what is the average number of probes to perform an insertion?