CS435 Algorithms Mid-block exam

September 2007

Name ________________________________ ID______________________________

Answer True or False next to each question. (1½ points each.)

1. ______ A programming solution to a problem can be called an algorithm only if it finishes running in a finite time interval.
2. ______ An algorithm that has a time complexity of O(*n log n*) is also called an exponential algorithm.
3. ______Generally, an algorithm that runs in $O(n^2)$ time will take longer than an algorithm that has O(*n log n*) time complexity for some n_0 when $n > n_0$.
4. ______When deciding between using a linked-list and an array data structure, if the application will be frequently accessing the elements by rank and not inserting elements by rank then it is better to chose the list to store the elements.
5. ______For a Queue, implemented with a circular array, the enqueue and dequeue operations run in O(log *n*) time.
6. ______In a binary search tree, every internal node has two children.
7. ______The height of a tree *T* is equal to the maximum depth of all the external nodes of *T*.
8. ______In-order traversal of a tree means the node is "visited" after the node's parent is "visited".
9. ______A red-black tree that stores 1000 key-object items will have a height between 20 and 30.
10. ______Insertion-sort and selection-sort are best used only on short sequences of less than hundred keys since faster sorting algorithms are available for larger sequences.
11. ______The hash table implementation of an unordered dictionary is very efficient for finding items because a hash table uses binary search to find the key-object pair.
12. ______In Radix-sort, sorting in lexicographic order, the key is divided into components and the Bucket-sort algorithm is run on the input data using first the right-most or least-significant component, followed by Bucket-sorts using each component in order.

Multiple choice. Circle the letter of the statement with the best answer. (2½ points each)

13. An algorithm with O(n^2) average case time complexity that takes 10 seconds to execute for an input size of 1000 elements will take how long to run when the input size is 10,000 elements.
   a) less than 5 seconds
   b) between 5 and 50 seconds
   c) between 50 and 500 seconds
   d) between 500 and 5000 seconds
   e) more than 5,000 seconds

14. What is the worst case time complexity of an insertion into a red-black tree of size n?
   a) O(1).
   b) O(log n) .
   c) O(n).
   d) O(n log n).
   e) O(n^2).

15. What is the primary benefit offered by hash tables? (Pick only one.)
   a) They store the keys in sorted order.
   b) They expand automatically with no extra operations.
   c) They do not require a key to insert and retrieve objects.
   d) They are very fast for insertion and retrieval.

16. Which of these data structures would be best for implementing an ordered dictionary?
   a) heap
   b) red-black tree
   c) hash table
   d) stack
   e) queue

17. Which of these data structures would be best for implementing an unordered dictionary with keys that are strings?
   a) heap
   b) red-black tree
   c) hash table
   d) array
   e) queue

18. Which situation is Bucket-sort the best method to use for sorting?
   a) When the input size of the data elements is less than a million.
   b) When the keys are integers in a range less than the input size.
   c) When the keys are very long and can be sub-divided evenly.
   d) When the keys are short strings less than 32 characters.

19. In a Red-Black tree, the restructuring and recoloring operations...

a) ... keep the balance between red and black nodes so they are always equal in number.
b) ...cause insertions to take about *n* times longer than searching.
c) ...are performed when searching for key-element pairs.
d) ...are designed to maintain the balance between branches so searches are faster.

20. What is the heap-order property for a min-heap?

a) All the external nodes do not store keys or key-element pairs.
b) All the internal nodes on a level are "to the left" of the external nodes on the same level.
c) The last internal node of the tree stores the minimum key.
d) The key stored at a node is greater than or equal to the key stored at the parent.

21. What is the primary advantage for implementing the Priority-Queue ADT using the min-heap data structure?

a) Heaps are more space-efficient than arrays or lists.
b) The heap provides random access to any key stored in the heap.
c) Inserting items in the heap always leaves them at the end of the list.
d) Finding and removing the element with the minimum key is very fast.

22. A total order relation for keys is necessary to make _____________.

a) key-object pairs.
b) proper comparisons.
c) even-sized components.
d) bin assignments.

Short answer questions.

23. (6 points) Examine following pseudo-code and answer the following questions:

```
Algorithm doSomething(A)
    Input: An array A of n positive integers
    Output: (see the question)
    Q ← empty queue
    t ← 0
    for i ← 0 to n − 1 do
            if A[i] is an odd number then
                Q.enqueue(A[i]).
            else
                while Q is not empty do
                        t ← t + Q.dequeue()
    while Q is not empty do
            t ← t + Q.dequeue().
    return t.
```

a) What is the output of this algorithm for the array A = {1, 9, 11, 14, 5, 3, 7, 13, 3, 12, 5}?

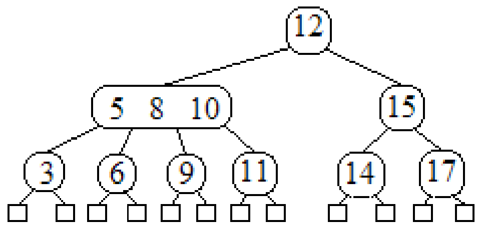
b) Describe in one sentence what this algorithm computes and returns.

c) What is the running time of this algorithm using big-O notation?

24. (5 points) Draw the hash table that results when inserting the following key-object pairs (only the keys are shown). Use linear-probing for collision handling. Let N=13. The hash function is $h(k) = (a_0 + a_1z) + a_2z^2)$ where z =33.

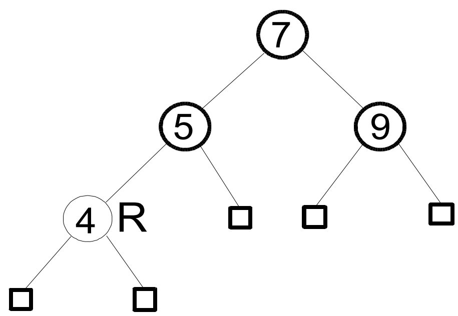
k	h(k)	h(k) mod N
aba	108964	11
bca	108998	6
bbc	111143	6
cab	110022	3
bbb	110054	9
cac	110022	7
acb	108997	5
bab	110021	2
aca	108997	5

25. (5 points) Let *T* be a (2, 4) tree shown below, which stores items with integer keys. Insert an item with key 7 into *T* and redraw the new tree. (2 bonus points: delete 17)



26. (5 points) Draw an AVL tree that stores items with integer keys. Insert these keys in the tree, following the rules of insertion and rebalancing: 12, 5, 6, 15, 9, 3, 17, 11, 16.
(2 bonus points: delete 5)

27. (5 points) Let T be a red-black tree shown below, which stores items with integer keys. Insert an item with key 3 in T and redraw the tree below. [Black nodes are darker. Label your red nodes with a letter for the color.] (2 bonus points: delete 9)



28. (5 points) Write the pseudo-code for the *remove(p)* operation of the List ADT, returning the element at *p*. Assume that the List is implemented using a doubly-linked list with sentinels that are regular nodes.

29. (12 points) Given an array A containing n unique integers in the range [0, n] in sorted order (smallest to largest). Describe an O(log n) time algorithm for finding the integer in the range [0, n] that is not in A. Write pseudo-code with comments to describe your algorithm. For example, if n=12, and A is {0,1,2,3,4,5,6,7,8,9,11,12 }your algorithm should return 10.

30. (8 points) Suppose you wish to sort a sequence of n=1,000,000 numbers, each of which is an RSA encryption key with 128 binary digits. How many rounds, d, would be needed to sort them using radix sort, if each round involves bucket sorting with N=1024 buckets? Would radix sorting be a better choice for this application than a general-purpose comparison sorting routine? Why or why not?

31. (5 points) Choose **one** of the following statements:

a) Describe a principle from the Science of Creative Intelligence that is expressed in the algorithm design of Merge-sort or Quick-sort.

b) Describe how the principle of "do less and accomplish more" is expressed in computer algorithm research.

c) Describe how some qualities of Creative Intelligence are expressed in the operations of the ordered dictionary abstract data type.

Cheat sheet for Enhanced Search Trees

AVL tree properties:

Heights of children can differ by at most 1.

AVL tree insertion:

After insertion, check for imbalance by looking up the tree from the new node.
Label node with first imbalance z. Label child of z with larger height, y.
Label child of y with larger height x.
Look at x,y,z and label them a,b,c according to in-order traversal.
Replace subtree rooted at z with the subtree rooted at b, rotating as needed.

AVL tree removal:

After removal, check for imbalance by looking up the tree from the parent of the node removed.
Label the node with the first imbalance z. Label child of z with larger height, y. Label child of y with larger height x.
Look at x,y,z and label them a,b,c according to in-order traversal.
Restructure by replacing subtree rooted at z with the subtree rooted at b, rotating as needed.

(2,4) tree properties:

Every node has at most four children.
All external nodes have the same depth.

(2,4) tree insertion:

To remedy overflow, perform a split creating 3-node with k_1 & k_2, move k_3 to parent and create 2-node for k_4.

(2,4) tree removal:

Node to remove might need to move to a node with external children using rule:

Swap node to be removed with in-order successor until it is in a node with external children.

Underflow can occur:

Case 1: Adjacent sibling is 2-node - Fusion needed: merge v with sibling and move item from parent to v. May cause underflow in parent.
Case 2: Adjacent sibling is 3- or 4-node - Transfer needed: 1) move child of sibling to v 2) move item from parent to v 3) move item from sibling to parent.

Red-black tree properties:

Root is black.
Every external node is black.
The children of a red node are black.
All the external nodes have the same black depth.

Red-black tree insertion:

To remedy double red -

Case 1: Sibling w of v is black - restructure.
Case 2: Sibling w of v is red - recolor.

Red-black tree removal:

To remedy double black -

Case 1: Sibling y of r is black and has a red child z - restructure.
Case 2: Sibling y of r is black and both children of y are black - recolor.
Case 3: Sibling y of r is red - do adjustment, then apply Case 1 or 2.