CS 331 Midterm Exam 2

Friday, December 4th, 2015

Please bubble your answers in on the provided answer sheet. Also be sure to write and bubble in your student ID number (without the leading 'A') on the answer sheet.

Part 1: Time Complexity

1. Which of the following specific run-time estimates would reduce to the highest run-time complexity?

   (a) $1000N^3$

   (b) $100000\log N + N^2$

   (c) $\frac{3^N}{1000} - N$

   (d) $10N^2 + 500N + 999\frac{N}{2}$

2. What is the time complexity for appending an element to the end of an array-backed list of N elements?

   (a) $O(1)$

   (b) $O(\log N)$

   (c) $O(N)$

   (d) $O(N\log N)$

3. What is the time complexity for removing an arbitrary element from an array-backed list of N elements?

   (a) $O(1)$

   (b) $O(\log N)$

   (c) $O(N)$

   (d) $O(N\log N)$

4. What is the time complexity for searching for an element in an unsorted array-backed list of N elements?

   (a) $O(1)$

   (b) $O(\log N)$

   (c) $O(N)$

(d) $O(N \log N)$

5. What is the time complexity for retrieving the element in the middle of a doubly-linked list of N elements?

   (a) $O(1)$

   (b) $O(\log N)$

   (c) $O(N)$

   (d) $O(N \log N)$

6. What is the time complexity for removing the last element from a doubly-linked list of N elements?

   (a) $O(1)$

   (b) $O(\log N)$

   (c) $O(N)$

   (d) $O(N \log N)$

7. What is the time complexity for inserting an item into an AVL tree?

   (a) $O(1)$

   (b) $O(\log N)$

   (c) $O(N)$

   (d) $O(N \log N)$

8. What is the run-time complexity of the following function?

```
def c1(N):
    ret = 1
    for i in range(N//2, N):
        ret *= i
    return ret
```

   (a) $O(\log N)$

   (b) $O(N)$

   (c) $O(N \log N)$

   (d) $O(N^2)$

9. What is the run-time complexity of the following function?

```
def c2(N):
    count = 0
    for i in range(1, N+1):
        for j in range(1, i):
            if i % j == 0:
                count += 1
    return count
```

(a) $O(\log N)$

(b) $O(N)$

(c) $O(N \log N)$

(d) $O(N^2)$

10. What is the run-time complexity of the following function?

```
def c3(N):
    x = 1
    q = N // 10
    while q > 0:
        x += 1
        q = q // 10
    return x
```

(a) $O(\log N)$

(b) $O(N)$

(c) $O(N \log N)$

(d) $O(N^2)$

Part 2: Searching & Sorting

11. The following is the binary search implementation we came up with in class:

```
def binary_search(lst, to_find):
    def binary_search_rec(bot, top):
        if bot > top:
            return None
        mid = (bot + top) // 2
        if to_find == lst[mid]:
            return lst[mid]
        elif to_find < lst[mid]:
            return binary_search_rec(bot, mid-1)
        else:
            return binary_search_rec(mid+1, top)
    return binary_search_rec(0, len(lst)-1)
```

Given the call `binary_search([2, 5, 7, 9, 13, 22], 22)`, which values (in order) in `lst` are compared to `to_find` before returning from the call?

(a) `5 9 13 22`

(b) `9 22`

(c) `7 13 22`

(d) `7 9 22`

12. The following is the insertion sort implementation we came up with in class:

```
def insertion_sort(vals):
    for j in range(1, len(vals)):
        to_insert = vals[j]
        i = j - 1
        while i >= 0 and vals[i] > to_insert:
            vals[i+1] = vals[i]
            i -= 1
        vals[i+1] = to_insert
```

When called with the array `[6, 4, 3, 1]`, what are the contents of `vals` at the end of each outer `for` loop?

(a)
```
[1, 6, 4, 3]
[1, 3, 6, 4]
[1, 3, 4, 6]
```

(b)
```
[4, 6, 3, 1]
[3, 4, 6, 1]
[1, 3, 4, 6]
```

(c)
```
[4, 3, 1, 6]
[3, 4, 1, 6]
[1, 3, 4, 6]
```

(d)
```
[6, 4, 3, 1]
[3, 1, 6, 4]
[1, 3, 4, 6]
```

Part 3: Array-Backed List, Stack and Queue

13. Which of the following correctly "removes" the element at position `idx` from an array-backed list?

(a)
```
for i in range(len(self.data)-1, idx, -1):
    self.data[i-1] = self.data[i]
```

(b)
```
for i in range(idx):
    self.data[i] = self.data[i+1]
```

(c)
```
for i in range(0, idx-1, 1):
    self.data[i+1] = self.data[i]
```

(d)
```
for i in range(idx, len(self.data)-1):
   self.data[i] = self.data[i+1]
```

14. In a circular, array-backed queue implementation, which of the following appends x to the tail of the queue and correctly advances the `tail` index? Assume the queue is not full.

(a)
```
self.data[self.tail] = x
self.tail = (self.tail % len(self.data)) + 1
```

(b)
```
self.data[self.tail] = x
self.tail = len(self.data) + 1 - self.tail
```

(c)
```
self.data[self.tail] = x
self.tail = (self.tail + 1) % len(self.data)
```

(d)
```
self.data[self.tail] = x
self.tail = (len(self.data) + 1)) % len(self.data)
```

15. In a circular, array-backed queue implementation, which of the following can be used as the body of a generator-based `__iter__` method? Assume that `self.count` denotes the number of elements in the queue, and `self.head` is the index of the head element.

(a)
```
for i in range(self.tail):
    yield self.data[i]
```

(b)
```
for i in range(self.count):
    yield self.data[(self.head + i) % self.tail]
```

(c)
```
for i in range(self.count+1):
    yield self.data[self.head + i]
```

(d)
```
for i in range(self.count):
   yield self.data[(self.head + i) % len(self.data)]
```

16. In a dual-stack (referenced by `outbox` and `inbox`) backed queue implementation, which of the following deals with the scenario when the `outbox` is empty when attempting to dequeue?

(a)
```
return self.inbox.pop()
```

(b)
```
self.outbox.push(self.inbox.pop())
return self.outbox.pop()
```

(c)
```
while self.inbox:
    self.outbox.push(self.inbox.pop())
return self.outbox.pop()
```

(d)
```
while self.inbox:
    self.outbox.push(self.inbox.pop())
while self.outbox:
    self.inbox.push(self.outbox.pop())
return self.inbox.pop()
```

Part 4: Linked List

17. Given that `to_rem` refers to a node in a singly-linked (i.e., nodes contain only `next`, and not `prior`, references) list, and `self.head` refers to the sentinel head, which of the following removes to_rem from the list?

(a)
```
p = self.head
while p.next.next is not to_rem:
    p = p.next
p.next = p.next.next
```

(b)
```
to_rem.next = to_rem.next.next
self.head = to_rem
```

(c)
```
p = self.head.next
while p is not to_rem:
    p = p.next
p = p.next
```

(d)
```
p = self.head
while p.next is not to_rem:
    p = p.next
p.next = p.next.next
```

18. Given that `self.head` refers to the sentinel head link of a circular, doubly-linked list implementation, which of the following prepends x to the beginning of the list?

(a)
```
l = LinkedList.Link(x, prior=self.head, next=self.head.next)
self.head.next.prior = l
self.head.next = l
```

(b)
```
l = LinkedList.Link(x, prior=self.head.prior, next=self.head)
self.head.prior = l
self.head.next = l
```

(c)
```
self.head.next = LinkedList.Link(x, prior=self.head,
next=self.head)
```

(d)
```
l = LinkedList.Link(x, prior=self.head, next=self.head)
self.head.prior.next = l
self.head.next.prior = l
```

19. Given that `self.head` refers to the sentinel head link of a circular, doubly-linked list implementation, which of the following sets `count` to the number of times x appears in the list?

(a)
```
count = 0
n = self.head.next
while n.next is not self.head:
    if x == n.val:
        count += 1
    n = n.next
```

(b)
```
count = 0
n = self.head.next
while n is not self.head:
    if x == n.val:
        count += 1
    n = n.next
```

(c)
```
count = 0
n = self.head.next
while n is not self.head and n.val != x:
    count += 1
    n = n.next
```

(d)
```
count = 0
n = self.head
while n.next is not self.head:
    if x == n.val:
        count += 1
    else:
        n = n.next
```

20. Given that n1 and n2 refer to successive nodes (other than the sentinel) within a circular, doubly-linked list implementation, which of the following swaps their positions?

(a)
```
n1.prior = n2
n2.next = n1
n2.prior = n1.prior
n1.next = n2.next
n1.prior.next = n2
n2.next.prior = n1
```

(b)
```
n1.prior.next.prior = n2
n2.next.prior.next = n1
n1.prior = n2
n2.next = n1
```

(c)
```
n2.prior = n1.prior
n1.prior.next = n2
```

```
n1.next = n2.next
n2.next.prior = n1
n1.prior = n2
n2.next = n1
```

(d)
```
n1.prior.next = n2
n2.next.prior = n1
n2.prior = n1.prior
n1.next = n2.next
n1.prior = n2
n2.next = n1
```

Part 5: Binary Search Tree

21. What is the maximum number of values that can be stored in a binary search tree with a height of *H*?

    (a) $H \log H$

    (b) $H^2 - 1$

    (c) $H^2 - 2H$

    (d) $2^H - 1$

22. Which of the following returns the number of occurrences of element x in the binary search tree rooted at node n (assuming that duplicate elements are possible)?

    (a)
    ```
    def count_of(x, n):
        if not n:
            return 0
        elif x >= n.val:
            return 1
        else:
            return count_of(x, n.left)
    ```

    (b)
    ```
    def count_of(x, n):
        if not n:
            return 0
        elif x == n.val:
            return 1 + count_of(x, n.right)
        elif x < n.val:
            return count_of(x, n.left)
        else:
            return count_of(x, n.right)
    ```

    (c)
    ```
    def count_of(x, n):
        if not n:
            return 0
        elif x < n.val:
    ```

```
            return count_of(x, n.left)
        elif x > n.val:
            return count_of(x, n.right)
        else:
            return 1
```

(d)
```
def count_of(x, n):
        if not n:
            return 0
        elif x == n.val:
            return count_of(x, n.left) + count_of(x, n.right)
        elif x < n.val:
            return -1
        else:
            return 1
```

For the remaining problems in this part of the exam, consider the non-balanced binary search tree constructed from the following (ordered) sequence of values:

```
10 15 8 5 6 20
```

23. What is the value stored in the root of the tree?

    (a) `5`

    (b) `8`

    (c) `10`

    (d) `20`

24. In what order are the values visited in a *pre-order* traversal of the tree?

    (a) `5 6 8 10 15 20`

    (b) `8 10 5 15 6 20`

    (c) `10 8 5 6 15 20`

    (d) `20 15 10 8 6 5`

25. In what order are the values visited in a *post-order* traversal of the tree?

    (a) `5 6 8 15 20 10`

    (b) `6 5 8 20 15 10`

    (c) `8 5 6 15 20 10`

    (d) `20 15 10 8 6 5`

26. What is the height of the tree?

(a) 4

(b) 3

(c) 2

(d) 1

27. If inserting the value 9 into the tree, where would it go?

    (a) Above node (8)

    (b) As the right child of node (6)

    (c) As the right child of node (8)

    (d) As the left child of node (10)

28. If deleting the value `10`, how should we best go about updating the tree?

    (a) Set the right child of node (8) to node (15), and make (8) the root of the tree

    (b) Set the value of node (10) to 20, and set the right child of (15) to None

    (c) Set the left child of node (20) to node (8), and make (20) the root of the tree

    (d) Set the value of node (10) to 6, and set the right child of (5) to None

Part 6: Balanced Binary Search Tree (AVL Tree)

29. Consider the following (ordered) sequence of values used to construct a binary search tree:

    ```
    20 15 5
    ```

    What sequence of rotations (if any) would be required to balanced the resulting tree?

    (a) left rotation about node (5)

    (b) right rotation about node (20)

    (c) left rotation about node (15), followed by a right rotation about (20)

    (d) no rotations are needed

30. Consider the following (ordered) sequence of values used to construct a binary search tree:

    ```
    5 12 8
    ```

    What sequence of rotations (if any) would be required to balanced the resulting tree?

    (a) left rotation about node (5)

    (b) left rotation about node (12)

(c) right rotation about node (12), followed by a left rotation about node (5)

(d) no rotations are needed

31. Which of the following implements a right rotation around the root of a binary search tree referred to by `t`?

(a)
```
r = t.root.right
l = t.root.left
t.root = BSTree.Node(l.val, left=r, right=l)
```

(b)
```
r = t.root
l = t.root.left
t.root = BSTree.Node(l.val, left=l.left, right=r)
r.left = l.right
```

(c)
```
r = t.root
l = t.root.left
t.root = BSTree.Node(l.val, left=l, right=r)
r.right = l.left
```

(d)
```
r = t.root.right
l = t.root
t.root = BSTree.Node(r.val, left=r, right=r.right)
```

32. Which of the following conditions returns true for a “RR" imbalance at node `n` in a binary search tree?

(a)
```
(BSTree.height(n.left)+1 < BSTree.height(n.right) and
 BSTree.height(n.right.left) < BSTree.height(n.right.right))
```

(b)
```
(BSTree.height(n.left) < BSTree.height(n.right)+1 and
 BSTree.height(n.right.left) < BSTree.height(n.right.right)+1)
```

(c)
```
(BSTree.height(n.left) >= BSTree.height(n.right) and
 BSTree.height(n.right.left) <= BSTree.height(n.right.right))
```

(d)
```
(BSTree.height(n.left)+1 < BSTree.height(n.right) and
 BSTree.height(n.right.left) > BSTree.height(n.right.right)+1)
```

For the remaining problems in this part of the exam, consider the *balanced* AVL tree constructed from the following (ordered) sequence of values:

```
1 2 3 4 5 6
```

33. In what order are the values visited in a *pre-order* traversal of the tree?

(a) `6 5 3 4 2 1`

(b) `4 2 1 3 5 6`

(c) `2 3 1 4 5 6`

(d) `1 2 3 4 5 6`

34. Which of the following values, if added to the tree, would require additional rebalancing (through one more more rotations)?

    (a) 0

    (b) 3.5

    (c) 4.5

    (d) 5.5

35. How many rotations were needed over the course of adding all the values to keep the tree balanced?

    (a) 2

    (b) 3

    (c) 4

    (d) 6

36. What types of rotations were performed?

    (a) 2 right rotations

    (b) 3 left rotations

    (c) 2 left rotations, 2 right rotations

    (d) 3 right rotations, 3 left rotations