Data Structures and Programming

Spring 2017, Midterm Exam. Solutions

April 25, 2017

---

1. (20 pts) True or False? (Score = max $\{0, \text{Right} - \frac{1}{2}\text{Wrong}\}$). No explanations are needed. (Note: the $\log n$ function is of base 2.)

(1) $n! = O(n^n)$
Ans. ◯

(2) $n^n = O(3^n)$
Ans. $\times$

(3) $\frac{n^2}{\log n} = O(n^{1.5})$
Ans. $\times$

(4) $33n^3 + 4n^2 = \Omega(n^2 \log^2 n)$
Ans. ◯

(5) $\sqrt{n} + \log n = \Theta(\log n)$
Ans. $\times$

(6) If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.
Ans. ◯

(7) If $f(n) = \Omega(g(n))$, then $f(n) = \omega(g(n))$.
Ans. $\times$

(8) Consider a sorted circular doubly-linked list where the head element points to the smallest element in the list. The asymptotic complexity of finding the median element in the list is $O(n \log n)$.
Ans. $\times$ (The correct answer is $O(n)$, although you will receive full credit regardless of what your answer is.)

(9) A stack is based on a FIFO (first-in-first-out) rule.
Ans. $\times$

(10) Each of the common operations of a stack can be implemented using an array in $O(1)$ time.
Ans. ◯

(11) There are five different binary trees with three nodes.
Ans. ◯

(12) Inserting n keys into an initially empty AVL tree takes $O(n \log n)$ time in the worst case.
Ans. ◯

(13) Inserting a key into an AVL tree of n nodes may require $O(\log n)$ rotations in the worst case.
Ans. $\times$

(14) Deleting the minimum key of an AVL tree of n nodes may require $O(\log n)$ rotations in the worst case.
Ans. ◯

(15) Binary search is as efficient on linked lists as on arrays, provided the list is doubly linked.
Ans. $\times$

(16) The queue ADT is useful in evaluating a postfix expression.
Ans. $\times$

(17) $x \oplus y \oplus x = y$, where $\oplus$ denotes the exclusive-or operator.
Ans. ◯

(18) The sentinel node of a singly linked list is an extra node located at the end of the list to make the implementation of list operations simpler.
Ans. $\times$

(19) In a threaded binary tree, if the right-child pointer of a node is a thread, it points to the immediate successor of the node in the preorder sequence.
Ans. $\times$

(20) An AVL tree is an abstract data type supporting insertion, deletion, and search operations.
Ans. ×

2. (10 pts) Consider the following pseudo-code program.

Procedure $P(a,n)$
  If $n=0$ Return 1
  If $n=1$ Return a
  If n is even
    Return $P(a\times a, \frac{n}{2})$
  else
    Return $a\times P(a\times a, \frac{(n-1)}{2})$

(1) (4 pts) What does the program compute?
Ans. a^n

(2) (6 pts) What is the running time of the program (in Θ notation)? Why?
Ans. $\Theta(\log n)$

3. (10 pts) Complete the following table to show the progress of converting the infix expression $2+1-(5-3*1)*6$ to its postfix expression. Write "*a b c*" to denote that the stack contains three symbols *a*, *b*, and *c*, and the top-of-the-stack symbol is *c*.

Input	2	+	1	-	(	5	-	3	*	1	)	*	6
Stack	empty	+	+	-	-(	-(	-(-	-(-	-(-*	-(-*	-	-*	empty
Output	2	2	21	21+	21+	21+5	21+5	21+53	21+53	21+531	21+531*-	21+531*-	21+531*-6*-

4. (15 pts) Show how to implement a queue Q using two stacks S_1 and S_2. When you use the stack for implementing a queue, you are only allowed to use the stack commands, namely *makenew(S), top(S), pop(S), push(x, S)* and *isempty(S)*, where $S\in\{S_1,S_2\}$. Note that *top(S)* returns the top element of stack S.

(1) (10 pts) Explain how you would implement all the following queue operations: *makenew(Q), front(Q), enqueue(x, Q), dequeue(Q)* and *isempty(Q)*. Note that *makenew(Q)* creates an empty queue; *front(Q)* returns the front element of Q ... etc. Fill in blanks in the following table with pseudo-codes.

Queue operation	*makenew(Q)*	*front(Q)*	*enqueue(x, Q)*	*dequeue(Q)*	*isempty(Q)*
Implementation using two stacks					

Ans. Consider the following implementation. Let the top of S_1 be the front of Q and the top of S_2 be the rear of Q.

- *makenew(Q)*:
  Ans. *makenew(S_1)*; *makenew(S_2)*
- *front(Q)*:
  Ans.
  If ($\neg$*isempty(S_1)*) return *top(S_1)*
    else
      while ($\neg$*isempty(S_2)*)
          x:=pop(S_2)
          push(x, S_1)
      return *top(S_1)*
- *enqueue(x, Q)*:
  Ans. *push(x, S_2)*
- *dequeue(Q)*
  Ans.

If ($\neg$*isempty*(S_1)) *pop*(S_1)
else
while ($\neg$*isempty*(S_2))
x:=*pop*(S_2)
push(x, S_1)
return *pop*(S_1)

- *isempty*(Q)
  Ans. *isempty*(S_1) $\wedge$ *isempty*(S_2)

(2) (5 pts) What is the worst case running time of executing a sequence of n queue operations using the above implementation? Why?
Ans. $O(n)$. In the life time of an item x, it can only be pushed into S_2, popped from S_2, pushed into S_1 and popped from S_1.

5. (15 pts) Recall that in a binary tree, a node may have 0, 1, or 2 children. In the following questions about binary trees, the height of a tree is the length (number of edges) of the longest path. A tree consisting of just one node has height 0.

(a) What is the maximum number of nodes in a binary tree of height d?
Ans. $2^{d+1} - 1$

(b) What is the minimum number of nodes in a binary tree of height d?
Ans. $d + 1$

(c) What is the maximum height of a binary tree containing n nodes?
Ans. $n - 1$

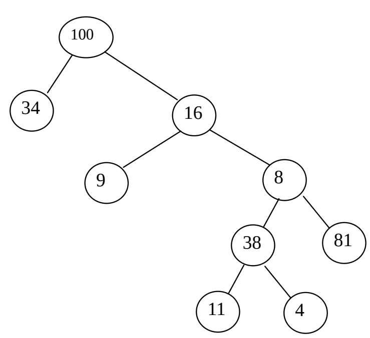
(d) What is the minimum height of a binary tree containing n nodes?
Ans. $\lfloor logn \rfloor$

(e) What is the maximum number of leaf nodes in a binary tree of height d?
Ans. 2^d

6. (15 pts) Suppose we know the preorder and postorder traversal sequences of a binary tree T.

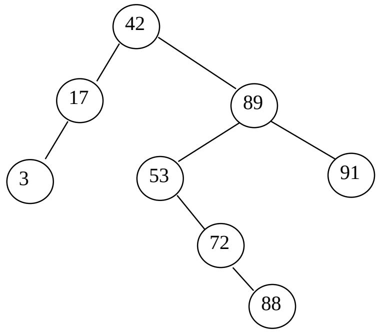
(1) (5 pts) Can we uniquely determine the binary tree? Answer with a short justification.
Ans. No. Consider tree T: root 1 with left child 2, and tree T': root 1 with right child 2. Both T and T' have preorder 1 2 and post order 2 1.

(2) (10 pts) Let the preorder traversal sequence of T be 100, 34, 16, 9, 8, 38, 11, 4, 81 and postorder traversal sequence be 34, 9, 11, 4, 38, 81, 8, 16, 100. If all the non-leaf nodes of T have two children, identify (i.e., draw) T. Show your steps in sufficient detail.
Ans.

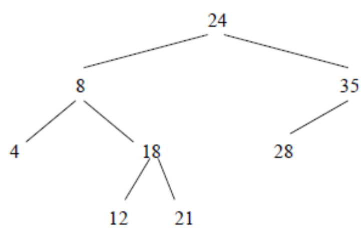


7. (5 pts) Draw the standard binary search tree that results from inserting the following data values in the order given: 42 17 89 53 72 91 3 88.

Ans.

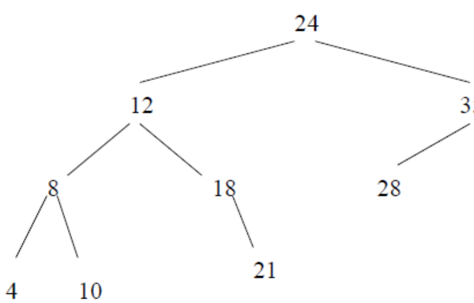


8. (10 pts) Consider the following AVL tree. Show the resulting trees after inserting 10, and then again after deleting 28.



Ans.

AVL Tree
After inserting 10



After deleting 28