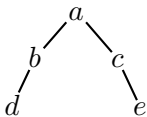


School of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 8

Sample answers

1. Traverse  in preorder, inorder, and postorder.

Answer: Preorder: $a-b-d-c-e$. Inorder: $d-b-a-c-e$. Postorder: $d-b-e-c-a$.

2. A certain binary yields a, b, c, d, e when traversed preorder, and it yields the same sequence when traversed inorder. What does the binary tree look like?

Answer: It is a stick, with nodes $a \cdots e$, each parent having only a right-child.

3. A certain binary tree yields 14, 83, 63, 42, 19, 27, 74, 99, 51, 37 when traversed preorder and 63, 83, 19, 42, 14, 27, 99, 51, 74, 37 when traversed inorder. Which sequence does it yield when traversed postorder?

Answer: Postorder yields 63, 19, 42, 83, 51, 99, 37, 74, 27, 14. To see this, construct the binary tree for the preorder and inorder sequences.

4. The following algorithm was designed to compute the number of leaves in a binary tree T :

```
function LEAFCOUNT(T)
  if  $T = \text{EmptyTree}$  then
    return 0
  else
    return LEAFCOUNT( $T.\text{left}$ ) + LEAFCOUNT( $T.\text{right}$ )
```

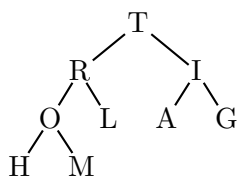
Fix the error in this algorithm.

Answer: We need to check for the case of a single-node tree:

```
function LEAFCOUNT(T)
  if  $T = \text{EmptyTree}$  then
    return 0
  else
    if  $T.\text{left} = \text{EmptyTree}$  and  $T.\text{right} = \text{EmptyTree}$  then
      return 1
    else
      return LEAFCOUNT( $T.\text{left}$ ) + LEAFCOUNT( $T.\text{right}$ )
```

5. Make a max-heap out of the keys A, L, G, O, R, I, T, H, M, using the bottom-up heap creation algorithm.

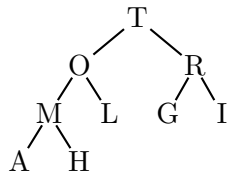
Answer:



Or, as an array: T, R, I, O, L, A, G, H, M.

6. Construct a max-heap from the empty heap by inserting the keys A, L, G, O, R, I, T, H, M, one by one, in that order. Is the result the same as the heap from the previous question?

Answer:



Or, as an array: T, O, R, M, L, G, I, A, H, which is different from the previous question's answer.

7. Give an algorithm for deciding whether an array $A[1..n]$ is a heap.

Answer: Easy:

```

function ISHEAP( $A[1..n]$ )
  for  $i \leftarrow 1$  to  $n/2$  do
    if  $A[i] < A[2 \times i]$  then
      return False
    if  $2 \times i < n$  and  $A[i] < A[2 \times i + 1]$  then
      return False
  return True
  
```