UNIVERSITY OF BIRMINGHAM

School of Computer Science

Data Structures & Algorithms

Sample Exam Questions

Data Structures & Algorithms

1. One can, inefficiently, implement a queue using two stacks, without using any further data structures such as arrays, linked lists etc.

   (a) Explain the basic approach to implementing a queue this way. **[5 marks]**

   (b) Write the pseudocode for the enqueue function. **[5 marks]**

   (c) Write the pseudocode for the dequeue function. **[5 marks]**

   (d) Analyse, in terms of $O(g(n))$, the complexity of each operation and explain, briefly, your reasoning. **[5 marks]**

   Topics: 02 - Stacks and Queues, 03 - Efficiency and Complexity

2. The following is the pseudocode for a recursive version of a function to calculate the size of a binary tree.

```
size(t)
{
    if ( isEmpty(t) )
        return 0
    else
        return (1 + size(left(t)) + size(right(t)))
}
```

   Write a non-recursive pseudocode version using a stack to keep track of the work still to be done. **[20 marks]**

   Topics: 04 - Trees

3. Construct a **minimal** AVL tree of height 5 of positive integers with no duplicate values allowed.

   (a) Draw that AVL tree. **[5 marks]**

   (b) Identify a value that on insertion into the tree would not trigger any rotation. **[5 marks]**

   (c) Identify a value that on insertion into the original tree that would trigger precisely one rotation. **[5 marks]**

   (d) Identify an insertion into the original tree that would trigger precisely one double rotation, i.e. a right rotation followed by a left rotation or a left rotation followed by a right rotation. **[5 marks]**

   Topics: 05 - AVL-Trees

4. For each of the following functions to calculate the largest element of an array, work out the formula for the number of operations (assignments, comparisons and arithmetic operations) executed as a function of the size of the array, n, state which complexity class in big O notation the function belongs to and justify your answer with a short explanation. Assume that the sort algorithm used in (c) takes $O(n \log n)$ operations.

(a)
```
int largest1(int[] arr) {
  int n = arr.length;
  int max = 0;
  for (int i=0; i<n; i++) {
    bool largest = true;
    for (int j=0; j<n; j++) {
      if (arr[i] < arr[j])
        largest = false;
    }
    if (largest)
      max = arr[i];
  }
  return max;
}
```

[7 marks]

(b)
```
int largest2(int[] arr) {
  int n = arr.length;
  int max = 0;
  if (arr.length == 0) {
    return 0;
  } else {
    max = arr[0];
    for (int i=0; i<n; i++) {
      if (arr[i] > max)
        max = arr[i];
    }
    return max;
  }
}
```

[7 marks]

(c)
```
int largest3(int[] arr) {
  sort(arr);
  if (arr.length == 0) {
    return 0;
  } else {
    int last = arr[arr.length - 1];
    return last;
  }
}
```

[6 marks]

Topics: : 03 - Efficiency and Complexity

5. Consider a binary search tree used to store integers with methods `isEmpty(t)`, `left(t)`, `right(t)` and `root(t)`, to return whether the tree `t`, is empty, the left child tree, the right child tree and the integer value stored at the root respectively.

   - Write a **recursive** function `sum_rec(tree)` to calculate and return the sum of all integers stored in the tree. **[8 marks]**
   - A `Queue` ADT has, for a queue `q`, methods `q.enqueue(val)` and `q.dequeue()` to enqueue a value val in the queue and to dequeue and return a value from the queue respectively. It also has a constructor which can be invoked with the commmand `new Queue()` to create and return a new empty queue.

     Write the pseudocode for a **non-recursive** function `sum_nonrec(tree)` using the `Queue` ADT, to calculate and return the sum of all integers stored in the tree. **[12 marks]**

   Topics: 02 - Stacks and Queues, 04 - Trees