CS340 – Advanced Data Structures and Algorithm Design – Fall 2020
Assignment 2 – September 18, 2020

Dr. Sandra Zilles, Department of Computer Science, University of Regina

due September 28, 2020, 10.00am, submitted as a single pdf file in UR Courses

Problem 0 (1 mark). Write down a joke or draw a silly cartoon to make the TAs laugh.

Problem 1 (6 marks). Design an algorithm (in pseudocode) that lists out the nodes of a binary tree in **level-order (breadth-first)**: it first lists the root, then the nodes at depth 1, then the nodes at depth 2, and so on. Assume that every node of a tree is represented in a structure that contains the data element stored at the node, a pointer to the left child of the node, and a pointer to the right child of the node.

It is required that your algorithm has a worst case running time of $O(N)$, where N is the number of nodes in the tree. Explain why your algorithm fulfills this requirement.

Hint: The ADT Queue might be helpful.

Problem 2 (6+2 marks).

(a) Show the result of accessing the keys 3, 9, 1, 5 in order in the splay tree in textbook figure 4.76 (page 185). Additionally, show the result of each splaying step (i.e., show intermediate trees that illustrate the process). (This is an extension of textbook problem 4.27.)

(b) Show the result of deleting the element with key 6 in the resulting splay tree for Problem 2(a). (This is textbook problem 4.28.)

Problem 3 (3+4 marks).

(a) What is the worst case, average case, and best case running time of insertion into slot `a[0]` of an (unsorted) array `a` containing N integers, representing a stack? Give Θ bounds and explain your answer.

(b) What is the amortized running time of insertion of N integers into an initially empty array `a`, representing a stack, if every insertion takes place at `a[0]`? Give a tight O bound and explain your answer.

Problem 4 (2+2+4 marks). Consider the problem of implementing a `k`-bit binary counter that counts upward from 0. We use an array `a[0..k-1]` of `k` bits. A binary number stored in the array has its lowest-order bit in `a[0]` and its highest-order bit in `a[k-1]`. For example, if `k` equals 8 then the number 12 would be represented as

a[7]	a[6]	a[5]	a[4]	a[3]	a[2]	a[1]	a[0]
0	0	0	0	1	1	0	0

To add 1 (modulo $2^{\mathtt{k}}$) to the counter value, we use the following procedure (given in pseudocode).

```
IncrCounter:
i=0;
while i < k and a[i] == 1
  {
     a[i] = 0;
     i++;
  }
if i < k
  {
     a[i] = 1;
  }
```

(a) What is the worst case running time of a single execution of `IncrCounter`? Give a Θ bound in the variable `k` and explain your answer.

(b) Write down the sequence of states of the array `a[0..5]` for 16 executions of `IncrCounter`, starting from the state where all entries of `a` are 0. Mark which bits have been flipped.

(c) What is the amortized running time of M executions of `IncrCounter` starting from an array `a` in which all entries are 0? Give an O bound and explain your answer. **Hints:** (i) Observe which bits are flipped how often in (b). (ii) In the very last step of your analysis, it may be useful to know that

$$1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{k-1}} = \sum_{i=0}^{k-1} \frac{1}{2^i} < \sum_{i=0}^{\infty} \frac{1}{2^i} = 2\,.$$