

Cover page for CSC263 Homework #2

(fill and attach this page to your homework)

SUBMITTED BY:

(1) Family Name: _____ (2) Family Name: _____
Given Name: _____ Given Name: _____
Student Number: _____ Student Number: _____

Graded Homework should be returned in Tutorial: _____(tutorial room number)

FOR HELP WITH YOUR HOMEWORK YOU MAY CONSULT ONLY THE INSTRUCTOR, THE TEACHING ASSISTANTS, YOUR HOMEWORK PARTNER (IF YOU HAVE ONE), YOUR TEXTBOOK AND YOUR CLASS NOTES. **You may not consult any other source.**

By virtue of submitting this homework I/we acknowledge that I am/we are aware of the policy on homework collaboration for this course.

Homework Assignment #2
Due: February 16, 2012, by 5:30 pm
(in the drop box for this course in Bahen 2220)

1. *On the cover page of your assignment, in addition to your name(s), you must write the location of the tutorial where you want the graded homework to be returned.*
2. *For any question, you may use data structures from class without describing how the operations on these data structures are implemented. You may also use any result (e.g., a worst-case time complexity bound) that we covered in class, or is in the course textbook, by referring to it.*
3. *Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision and conciseness of your presentation.*

Question 1. (10 marks) In this question, you must use the insertion and deletion algorithms as described in the “Balanced Search Trees: AVL trees” handout posted on the course web site.

Insert into an initially empty AVL tree each of the following keys, in the order in which they appear in the sequence: 19, 6, 9, 16, 22, 2, 11, 27, 18, 15, 10, 14.

Show the resulting AVL tree T . (Only the final tree should be shown; any intermediate trees shown will be disregarded, and not given partial credit.)

From AVL tree T , delete 16, and show the resulting tree.

In the two trees you must show the key and balance factor of each node.

Question 2. (20 marks)

We want an efficient algorithm for the following “on-line” problem. The algorithm is given an integer $m \geq 2$, and then a (possibly infinite) sequence of distinct keys are input to the algorithm, one at a time. A *query* operation can occur at any point between any two key inputs in the sequence. When a *query* occurs, the algorithm must return, *in sorted order*, the m smallest keys among all the keys that were input before the *query*. Assume that at least m keys are input before the first *query* occurs.

For example, suppose $m = 3$, and the key inputs and query operations occur in the following order:

20, 15, 31, 6, 13, 24, *query*, 10, 17, *query*, 9, 16, 5, 11, *query*, 14, ...

Then the first *query* should return 6, 13, 15; the second *query* should return 6, 10, 13; the third *query* should return 5, 6, 9, etc.

a. (16 marks) Describe a simple algorithm that, for every $m \geq 2$, solves the above problem with the following worst-case time complexity:

- $O(\log m)$ for each key input operation, and
- $O(m)$ for each *query* operation.

Partial credit will be given for an algorithm whose worst-case time complexity is $O(\log m)$ for each key input and $O(m \log m)$ for each *query*. Your algorithm *must* use a data structure that we learned in class.

b. (4 marks) Explain why your algorithm achieves the above worst-case time complexity.

Question 3. (30 marks) A seaport has a number of ships awaiting cargo. Each ship is moored at some pier p and has some remaining loading capacity of c tonnes; note that there may be several ships with the same pier location and/or the same remaining loading capacity. You must design a data structure D that stores every ship in the seaport (including its name, pier location and remaining capacity) and supports efficient implementation of the four operations listed below.

When cargo arrives at the seaport to be loaded on a ship, you will need to decide which ship to load it on. We assume that all ships are sailing to the same destination. We don't want a single shipment to get split up, so a shipment of s tonnes must be loaded completely onto one ship whose remaining capacity is at least s . Of course, there may be several ships with sufficient capacity: you must choose to load on a ship with the smallest possible pier number (since our "better customers" get the lower pier numbers).

Thus your data structure D must support the following four operations:

INSERT(D, x): If x is a pointer to a new ship (with a name, pier location, and remaining capacity) that is docking insert the ship pointed to by x into D .

DELETE(D, x): If x is a pointer to a ship in D that's about to set sail, remove that ship from D .

FIND(D, s): Return a pointer to a ship in D such that: (a) its remaining capacity is at least s **and** (b) its pier number is minimum over all ships in D with a remaining capacity of at least s . If no ship in D has a remaining capacity of at least s tonnes, then return NIL.

LOAD(D, x, s): If x is a pointer to a ship in D , load s tonnes on it. If the capacity remaining on this ship was c , it is thus decreased to $c - s$.

In this question, you must explain how to implement D as an *augmented AVL tree* and explain how each of the above operations is executed. The worst-case run time of each of the above operations should be $O(\log n)$ where n is the number of ships in the data structure D .

Since this implementation of D is based on a data structure and algorithms described in class and in the AVL handout, you should focus on the extensions and modifications needed here (you do not have to reproduce code or details provided in class or in the handout).

a. (6 marks) Give a precise and full description of your data structure. Illustrate this data structure by giving an example of it on some collection of docked ships of your own choice.

b. (24 marks) Explain how to implement each of the above four operations in $O(\log n)$ time and explain why, in each case, your algorithm achieves this time complexity.