ECE-250 – Algorithms and Data Structures (Winter 2012)

Midterm Examination

2012-02-28, 17:30 – 18:50 (RCH-302 / RCH-306)

INSTRUCTIONS:

You must work on your exam strictly individually.

Questions are in order of the course material, and not in order of difficulty level.

No materials allowed:

- No books, no course notes or printouts of any kind.
- No calculators, no cellphones/smartphones, computers, or electronic devices of any kind.
- You must turn off any electronic devices and store them under your desk — simply having any device (even if turned off) with you during the exam constitutes a violation and will be reported.
- If you need to borrow a pencil, sharpener, eraser, etc., *you must ask a proctor*. You're not allowed to directly ask any of your neighbours.

No questions allowed during the exam:

- 2 marks deduction off the exam's grade *for each* question asked).
- The only acceptable question is "*may I go to the washroom?*"
- ***If a question seems ambiguous to you, or if you think there is a typo/mistake in the question, write down your assumptions and continue answering based on those assumptions.***
- You may report any "major" issues with the exam (question missing, page missing, etc.) through the form at the end of the exam (in such case, you should call one of the proctors and give them the completed form, *maintaining complete and absolute silence*)

Before, during, and at the end of the exam:

- You're not allowed to leave during the examination period.
- Do not stand up or talk until all exams have been picked up.
- You must stop writing when any of the proctors or the instructor announces that the examination is finished. NO EXCEPTIONS. (10 marks deduction off the grade of the exam, plus the incident would be reported as an academic offence)

THIS BLOCK MUST BE COMPLETED USING ALL CAPITAL LETTERS IN PEN

Surname / Last Name								
Legal Given / First Name(s)								
UW Student ID Number	2	0						
UW User ID								

I have read and understood the above
instructions and rules for the examination: _____

Student's signature

For all questions: if the space for the answer is not enough, use the back of the previous page, properly labelling.

1 – (5 pts) Prove that the result of a number modulo 2^n has a binary representation consisting of the n least-significant bits of the binary representation of the number.

See also solution to question 1 from practice problems

$$N = \sum_{k=0}^{m} b_k 2^k = \sum_{k=0}^{n-1} b_k 2^k + \sum_{k=n}^{m} b_k 2^k = \sum_{k=0}^{n-1} b_k 2^k + 2^n \sum_{k=n}^{m} b_k 2^{k-n} = r + 2^n q$$

where r denotes the first sum, and q the second one. Since r is an n-bit number, it is less than 2^n, and so it must be the remainder of the division $N \div 2^n$

2 – (5 pts – 1-2-2) Since logarithms of different bases only differ by a multiplicative constant, they are all Θ of each other. For exponentials, this is not the case — that is, $a^n \neq \Theta(b^n)$ if $a \neq b$.

(a) Explain why this is the case (you have to at least use the limit criterion in your answer as part of your justification)

(b) Indicate which Landau symbols (**o**, **O**, **ω**, or **Ω**) describe the relationship between them. Indicate *all* Landau symbols that apply, considering both cases, $a < b$ and $a > b$, and expressing the relationship as a^n = **??**(b^n), where **??** is the Landau symbol that you should indicate. Justify your answer.

(c) Indicate which Landau symbol *best describes* the relationship (both cases, $a < b$ and $a > b$), expressing the relationship like in part (b). Justify your answer.

(a) $$\lim_{n \to \infty} \frac{a^n}{b^n} = \lim_{n \to \infty} \left(\frac{a}{b} \right)^n$$

If $a > b$, the limit is ∞; if $a < b$, the limit is 0 — neither case matches the definition for Θ

(b) For $a < b$, since the limit is 0, we have that $a^n = o(b^n)$ and also $a^n = O(b^n)$
For $a > b$, since the limit is ∞, we have that $a^n = \omega(b^n)$ and also $a^n = \Omega(b^n)$

(c) The "closest match" is given by little-oh and little-omega — that is, these are the ones that best describe the relationship, in that the big-?? symbols account for two possibilities, whereas the little-?? symbols restrict to just one.

3 – (10 pts) Explain why Θ is an equivalence relation. You should justify each condition using either the formal definition of Θ or the limit criterion.

An equivalence relation ~ is defined by the following three properties: reflexivity (a ~ a), symmetry (a ~ b ⇒ b ~ a), and transitivity (a ~ b and b ~ c ⇒ a ~ c). For Θ, we have:

Reflexive: f(n) = Θ(f(n)) ∀ f(n) (this is trivial — limit is 1)

Symmetric: if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = L$, where L is finite and non-zero, then $\lim_{n \to \infty} \frac{g(n)}{f(n)} = \frac{1}{L}$ (also finite and non-zero)

Transitive: if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = L_1$ and $\lim_{n \to \infty} \frac{g(n)}{h(n)} = L_2$, with both L1, L2 finite and non-zero, then $\lim_{n \to \infty} \frac{f(n)}{h(n)} = \lim_{n \to \infty} \frac{f(n)}{g(n)} \cdot \lim_{n \to \infty} \frac{g(n)}{h(n)} = L_1 L_2$ (also finite and non-zero)

Notice that the manipulation is allowed because g(n) is guaranteed to be asymptotically non-zero.

4 – (10 pts) Determine the run time (expressed as Θ) of the following fragment of code, as a function of *n*. You may assume that *n* is large — in particular, much larger than 5. You may also assume that the expression **`n*n*n`** does not cause an integer overflow. Justify your answer.

```
for (int i = 5; i < sqrt(n*n*n); i *= 2)
{
    sum += i;
}
```

Note: **`sqrt`** is the square root function from the Standard Library. You may assume that that function runs in Θ(1) time.

(see also question 5 of practice problems)

We observe that sqrt(n^3) can simply be rewritten as $n^{3/2}$. So, i starts at 5, and is doubled at every pass — after k passes, its value is $5 \cdot 2^k$.

The loop stops when $i = n^{3/2}$, which is when $5 \cdot 2^k = n^{3/2}$, from which we solve for k:
$k = 3/2 \lg(n) - \lg(5) = \Theta(\log n)$

5 – (15 pts) We have the following modified versions of a binary search (in this case, binary search returns the position where the value is found, or −1 if the value is not found): For an array of n elements (values in ascending order), we have procedures P1 and P2 as described below:

P1: As usual for binary search, at each iteration we narrow down the interval to half the size (since we compare and discard one of the halves of the search interval). In this procedure P1, we narrow down the search interval until it has size 10 or less; at that point, we do a linear search for the value (you may assume that n is always much larger than 10).

P2: Suppose that our binary search procedure is limited to searching in arrays of no more than 1024 elements. So, for an array of n elements, with n larger than 1024, we split the array into chunks of 1024 (plus the leftover, if n is not an exact multiple of 1024), and do binary search on each of those arrays, in order (binary search on the first chunk, then on the second chunk, etc.). If one of those binary searches returns a non-negative value, then we return the appropriate position (i.e., we don't continue doing binary searches on the remaining chunks). Otherwise, we return −1.

(a) Determine the run time (worst-case) of P1 as a function of n (expressed as Θ).
(b) Determine the worst-case and average-case run times of P2 as a function of n (expressed as Θ).
(c) Suggest a modification to P2 to improve its worst-case run time (by "improve", we mean improve by more than just a constant speedup factor)

In all cases, justify your answers. For part (c), you may assume that n is less than one million, if you need to or if it makes it easier.

(a) This is clearly logarithmic (if the binary search continued until narrowing down to 1, we know it is logarithmic — this does the same number of iterations minus a fixed number of them at the end)

Then, the linear search on an interval of fixed (or bounded by a constant, in any case) length is constant time, so it does not change the complexity class.

(b) We execute one binary search (constant time, since it is applied to a fixed-size input) per chunk, and there are n/1024 chunks. Thus, this one is linear time.

(c) A possible improvement is to do binary search on the blocks first — careful: we DO NOT want to copy the start value of each block, because then we'd be stuck in linear time! The binary search is done on the start value of each block in-place — doing some simple arithmetic with the subscripts. This would be logarithmic time to find the right block; then we search within that block, resulting in logarithmic time for this modified version of P2.

6 – (10 pts) Given a circular doubly linked list with sentinel element (assuming data member **d_head** points to the sentinel element), draw a graphical representation of the memory after each step, and determine what the bug(s) is(are) in the following fragment of code, intended to insert a new value (a new node in the list) after a given node. Assume that **insert** is a friend function of class **Node**.

Note: if there is more than one bug, and for example the first bug was something that makes the program instantly crash, you will still need to list all the subsequent bugs — you're not allowed to claim that because the first bug makes the program crash, no other instructions will execute and thus there are no more bugs. In that sense, whenever a bug is identified, continue searching for subsequent bugs as if this first bug had not been present.

```
template <typename Type>
void insert (const Type & element, Node<Type> * node)
{
    Node<Type> * new_node;
    node->next()->d_previous = new_node;
    node->d_next = new_node;
    new_node->d_previous = node;
    new_node->d_next = node->next();
}
```

There are two bugs — the first one (and most blatant) is that new_node is uninitialized. Assuming, for the purpose of analyzing the rest, that it was (new_node = new Nodte<Type>(element);), then the second bug is at the last assignment — or, depending on how we want to look at it, we could say that it is at the third statement; we're assigning node->next and thus losing track of what used to be its next; then, by the last statement, we end up assigning new_node to point to itself!

(you were supposed to draw the diagrams of what's going in — clearly, I'm not going to do that with this annoying word processor!! :-))

7 – (10 pts) Given a hash table of size 10, with the hash function h(x) = x mod 10, and where linear probing is used to handle collisions, we have that at some point, the contents of the underlying array is as follows:

0	1	2	3	4	5	6	7	8	9
89	21	30	11					48	19

(a) Insert the value 13, then the value 18

(b) From the state as shown above, remove 19 (that is, from the state of the table without adding 13 or 18).

Show and explain each of the steps, and show the state of the table after each of the operations. (you may use the empty tables on the right)

(a) 13 hashes to 3, but bin 3 is taken, so we test bin 4 which is empty. 18 hashes to 8, which is taken, so we probe the following bins, cycling back to 0 and going all the way until finding 5 empty:

0	1	2	3	4	5	6	7	8	9
89	21	30	11	**13**	**18**			48	19

(b) Removing 19 leaves a hole, so we search forward for elements that become invalid. 89 is the first one that does (89 hashes to 9, and 9 is not empty, so it would not be found). Move 89 back; then, that gets 30 in trouble, so we move it back, which then gets 11 in trouble, so we move it back:

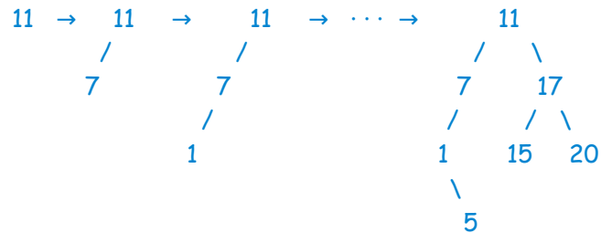
0	1	2	3	4	5	6	7	8	9
30	21	**11**						48	**89**

8 – (10 pts) Using the definition that a perfect binary tree of height $h > 0$ has two sub-trees that are each perfect binary trees of height $h - 1$, use Mathematical induction to prove that a perfect binary tree of height h has $2^{h+1}-1$ nodes.

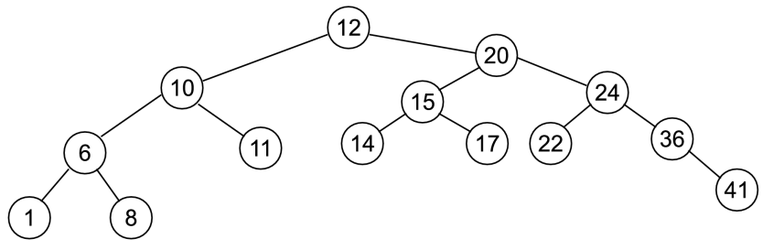
See slides on binary trees (2012-02-06), page 41 and the few that follow.

9 – (10 pts) Insert the following values, in the given order, into an initially empty binary search tree: 11, 7, 1, 5, 17, 15, 20. Explain/justify the steps — in this case, one sentence suffices, provided that it does reasonably explain all the key aspects involved in the various steps.

Brief explanation: to determine the position where a value should go, follow the rule for BSTs: if it is less than a node, it goes at the left, otherwise at the right:



10 – (15 pts – 3-12) Given the following AVL tree:



(a) What is the maximum number of values that can be inserted without increasing the height of the tree, and without causing any imbalance? (explain)

(b) Remove 17, then 22, then 10, doing any balancing as required. Show the steps (including the step-by-step procedures for any rotations), and draw the resulting tree after each removal (you may draw just the relevant sub-tree — for example, if we were to remove 6, promoting 8, you could in that case just draw the resulting sub-tree with root 10, with a ··· sign for the rest)

Slight glitch in this question — the drawing suggests that both sub-trees have the same height, since the ones on the right are more tightly squeezed; needless to say that we accepted both versions of the answers.

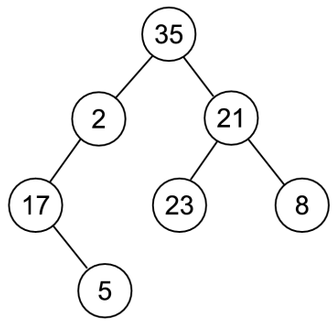
In any case, one detail that was expected in the answer (a) is that the node 11 does not accept any insertions (10 is already in the tree, and anything below 10 would not make it to below 11). Same for 12, and same for 15 on the right of 14.

(b) 17 is removed without causing any imbalance

Removing 22 causes an imbalance at 24, so we rotate left, ending with 36 at the root, 24 at its left, 31 at its right.

Removing 10 requires 11 to be promoted, causing imbalance at 11; it could be seen as an aligned imbalance, 11 – 6 – 1, fixed by rotating right, 6 ending at the root, 11 at its right, 1 at its left — 8, temporarily detached, is reattached at the left of 11, or a "zig-zag" type of imbalance, 10 – 6 – 8, which is fixed by a double rotation — 8 ends up at the root, with 11 at its right, 6 at its left; 1 stays where it was)

11 – (5 pts) Given the following binary tree



(a) Indicate the output of pre-order depth-first, in-order depth-first, and post-order depth-first traversals, justifying your answers. In this case, one sentence explaining each of the traversals suffices, provided that it does reasonably explain all the key aspects involved in the various steps

(b) Using a queue, show the step-by-step procedure to execute a breadth-first traversal (that is, you should indicate the contents of the queue at each step).

(a) Pre-order means root first, then left subtree, then right subtree; post-order means first both subtrees (left, then right), then the root. In-order means left subtree, then root, then right subtree:

Pre-order: 35 – 2 – 17 – 5 – 21 – 23 – 8
Post-order: 5 – 17 – 2 – 23 – 8 – 21 – 35
In-order: 17 – 5 – 2 – 35 – 23 – 21 – 8

(b) Enqueue the root: 35
Dequeue 35 and enqueue its children: 2 – 21
Dequeue 2 and enqueue its children: 21 – 17
Dequeue 21 and enqueue its children: 17 – 23 – 8
Dequeeue 17 and enqueue its children: 23 – 8 – 5
Dequeue those three in sequence to complete the breadth-first traversal.

Bonus Marks – (5 pts)

Prove that insertion of an element in a balanced binary search tree has a worst-case run time $\Omega(\log n)$, where n is the number of nodes in the tree, regardless of whether the tree is height-balanced, weight-balanced, or any type of balance. That is, prove that insertion in a balanced binary search tree can not have a worst-case run time faster than logarithmic time.

Proof: Reduction from sort:

We implement the following sorting algorithm: Take the n input values and insert them in the tree. Then, perform an in-order traversal to output the values in order.

The resulting time is less than the time for n insertions if the tree had n elements at all times: n T(n) + linear time for the traversal. Since this must be at least n log(n), then T(n) must be at least log(n).

Bonus Marks – (2 pts)

In reference to the article *Getting it Wrong: Surprising Tips on How to Learn*:

(a) Why do they claim that in the pre-evaluations, the subjects were bound to fail on almost every question?

(b) How much time did each of the groups (the group that did the pretest vs. the group that just studied the information, without having to first guess the answers) have to learn/memorize the associations? Comment on the significance of these numbers (i.e., don't just write down two numbers as your answer)

I'm not giving you the answers — go read the article !!! :-)

http://www.scientificamerican.com/article.cfm?id=getting-it-wrong