ECE 250 *Algorithms and Data Structures*
Sections 001 and 002
FINAL EXAMINATION

Douglas Wilhelm Harder dwharder@uwaterloo.ca EIT 4018 x37023

2014-04-16T09:00P2H30M

Rooms: PAC 7, 8

If you are writing a **supplemental examination** (you failed the course previously and are writing this examination to clear that failure), please check here: ____

Initial each instruction:

_____ There are 79 marks.
_____ No aides.
_____ Turn off all electronic media and store them under your desk.
_____ If there is insufficient room, use the back of the previous page.
_____ You may ask only one question during the examination:
"May I go to the washroom?"
_____ Asking **any** other question **will** result in a deduction of 5 marks from the exam grade.
_____ If you think a question is ambiguous, write down your assumptions and continue.
_____ **Do not leave during first hour or after there are only 15 minutes left.**
_____ Do not stand up until all exams have been picked up.
_____ If a question asks for an answer, you do not have to show your work to get full marks; however, if your answer is wrong and no rough work is presented to show your steps, no part marks will be awarded.

Attention:
The questions are in the order of the course material.

THIS BLOCK MUST BE COMPLETED USING ALL CAPITAL LETTERS (1 mark for getting it right)

Last name as appearing in Quest (e.g., HARDER)																			
H	A	R	D	E	R														
Legal Given names as appearing in Quest (e.g., DOUGLAS WILHELM)																			
D	O	U	G	L	A	S		W	I	L	H	E	L	M					
Common or nick name (entirely optional; e.g., DOUG)																			
D	O	U	G																

UW Student ID Number	**2**	**0**	1	2	3	4	5	6	
UW User ID									@uwaterloo.ca

I have read the above instructions:

Signature: ____

Asking Any Question other than those Questions noted above.

-5

A. Relations, asymptotic analysis and algorithm analysis

A.1 (3) Every linear relationship on a finite number of elements also defines a hierarchy where the first item is the root node and subsequent items are children of the previous entry. For each pair, enter the most appropriate word "Every", "Not Every", and "No".

_____ weak ordering defines a linear ordering.

_____ hierarchical ordering defines an adjacency relation.

_____ weak ordering defines an equivalence relation.

_____ hierarchical relation defines an equivalence relation.

_____ linear order defines an equivalence relation.

_____ linear ordering defines a weak ordering.

A.2 (4) What is the run-time of the following four loops?

```
for ( int i = 0; i < n*n*n; ++i ) {
    for ( int j = 0; j < n; ++j ) {
        ++sum;
    }
}

for ( int i = 0; i < n*n*n; ++i ) {
    for ( int j = 0; j < i; ++j ) {
        ++sum;
    }
}

for ( int i = 0; i < n; ++i ) {
    for ( int j = 0; j < n*n*n; ++j ) {
        ++sum;
    }
}

for ( int i = 0; i < n; ++i ) {
    for ( int j = 0; j < i*i*i; ++j ) {
        ++sum;
    }
}
```

B. Linear abstract data types, arrays and linked list

B.1 (5) Create a function that goes through a doubly linked list and erases the first instance of the largest element stored in the linked list. Return `true` if the largest element is unique and was erased, and `false` otherwise. You may call `pop_front()` and `pop_back()`, if necessary. Other member functions you may require are `head()`, `size()`, `tail()` and the list node class has `element`, `next_node` and `previous_node`, returned by `retrieve()`, `next()` and `previous()`, respectively.

```
template<typename Type>
bool Double_list<Type>::erase_largest() {
```

C. Hierarchies and trees

C.1 (3) Insert the words in the poem "she sells seashells by the sea shore" into an initially empty trie by drawing the existing nodes and their links. Be sure to identify the terminal nodes.

D. Ordered trees (perfect, complete and balanced trees)

D.1 (3) Prove by induction that every perfect ternary tree (three children) has

$$\frac{3^{h+1}-1}{2}$$

nodes.

E. Sorted abstract data types and search trees

E.1 (7) Implement the erase function for a binary search tree that returns true if the object is erased and false otherwise. Indicate using a star (*) where additional code would have to be added in order to AVL balance the tree in the case the erase causes an AVL imbalance. You can use the member functions `retrieve()`, `left()`, `right()` and `is_leaf()` and; and the member variables `element`, `left_tree` and `right_tree`. The first three member functions return the given member variables, respectively, and the `is_leaf()` member function returns true if the current node is a leaf node and false otherwise. Recall that the second argument is either root_node (if called from the `Binary_tree` class) or `left_tree` or `right_tree` from the parent.

```
bool Binary_search_node<Type>::erase( Type const &obj, Binary_search_node<Type> *&ptr_to_this ) {
```

E.2 (3) Insert the following 10 objects into an initially empty B+ tree with $L = 3$ and $M = 3$.

34, 15, 65, 59, 69, 42, 40, 80, 50, 65

You may wish to show reasonable intermediate steps to get part marks just in case you make a mistake.

E.3 (3) Explain in your own words the rational for considering data structures such as a B+ tree as opposed to AVL trees. In your discussion, be sure to compare and contrast the asymptotic run times of both data structures with respect to various operations as well as other considerations.

F. Priority queues and binary heaps

F.1 (3) Consider the merging process used in merge sort. Suppose we have two binary min-heaps stored as complete trees with an underlying array representation, as discussed in class. Could we merge these two binary min-heaps into a single binary min-heap using the same merging process? If yes, provide an argument or proof, if no, provide a counter-example and explain how the merging process could be modified so as to allow the merging of two binary min-heaps.

F.2 (1) We determined that top has a run time of $\Theta(1)$ while, on average for the insertion of randomly distributed values, the run time of a push operation has an amortized $\Theta(1)$ run time. Provide an argument as to why popping off a min heap will, however, be $O(\ln(n))$ regardless of other considerations.

H. Hash tables

H.1 (3) Using linear probing with the least-significant digit as the initial hash function, the nine numbers were inserted into the hash table shown in Table 1.

34, 15, 65, 59, 79, 42, 40, 80, 50

Table 1. A hash table using linear probing.

0	1	2	3	4	5	6	7	8	9
79	40	42	80	34	15	65	50		59

Erase the entries 59, 79, 42 and 50, in that order, from the hash table and put your solution in Table 2.

Table 2. Your solution.

0	1	2	3	4	5	6	7	8	9

G. Sorting algorithms

G.1 (2) If the number of inversions in a list of size n is o(n), then the run-time of insertion sort is o(n). Is this true or false, and why?

G.2 (3) Convert the array in Table 3 into a max-heap using the heapification algorithm we discussed during heap sort. Insert your answer into Table 4.

Table 3. An array of randomly generated values.

23	46	57	3	29	22	99	73	44	2	48	88	98

Table 4. The array in Table 3 converted into a max-heap.

G.3 (5) Devise a strategy for implementing the quicksort algorithm in C++ as discussed in class. You do not have to actually implement any of the functions, but you should give signatures for the various functions, a brief description of what each of the functions does, and describe how they would be called from the main top-level, the signature of which will be:

```
void quicksort( double *array, int a, int b );
```

This top-level function sorts the entries of the array from index a to index b. To sort an entire array, you would call `quicksort( array, 0, n - 1 );`. Marks will be assigned on the assumption your description is being given to an intelligent programmer who knows nothing about the quicksort algorithm. You should make intelligent choices about what should be called in appropriate sub-functions called from your main function. Indicating that everything should happen at the top level will result in a maximum grade of 1/5.

G.4 (3) During our discussion of radix sort for binary numbers, we described how we could use a *dual queue* to store the values. Show the result of the queue after the first loop of enqueuing and "dequeuing" these digits based on the least-significant bit so that the digits are linearly ordered for the next iteration of the algorithm. Put your answer in Table 5.

111, 110, 001, 011, 101, 110, 011, 001, 100, 010, 000, 000, 010.

Table 5. An array used for the radix sorting of the given 13 binary digits.

I. Graph algorithms

I.1 (3) What relationship must hold between courses and their prerequisites within the University of Waterloo. What data structure should be used to store the courses and the associated relationships? Use this to prove that not every course can have prerequisites. You may assume that the only prerequisites of a course taught at the University of Waterloo are other courses taught at the University of Waterloo.

I.2 (2) Given a directed acyclic graph with $|V|$ vertices, describe the arrangement of the minimum number of edges necessary to ensure that there is a single unique topological sort of the vertices.

I.3 (2) If we apply Prim's algorithm to the graph in Figure 1, we only get a minimum spanning tree on one connected component of the graph.

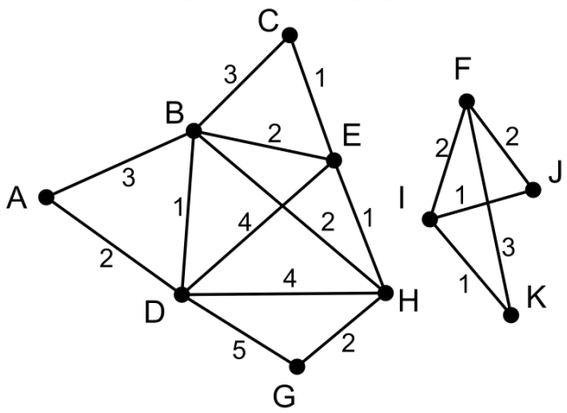

Figure 1. A weighted graph.

Assuming we have the two arrays `bool visited[num_vertices]` and `double distance[num_vertices]` as described in class, what would you do so that we continue to find a *minimum spanning forest*; that is, find minimum spanning trees on all connected subgraphs?

I.3 (4) Kruskal's algorithm for finding a minimum spanning tree in a weighted graph with $|V|$ vertices and $|E|$ edges is as follows:

1. Sort the edges from shortest to longest.
2. Start with a graph of only the vertices, and in succession, attempt to add each edge in order of length as long as inserting that edge does not cause a loop.

Suppose that checking if inserting an edge causes a loop takes $\Theta(\alpha(n))$ time where $1 = o(\alpha(n))$ and $\alpha(n) = o(\ln(n))$. What is the run-time of Kruskal's algorithm if we use quicksort? Describe under which conditions the run time could be reduced to $O(|E|\ \alpha(|E|))$. What sorting algorithm would be used to achieve such a run time?

J. Algorithm design

J.1 (2) You are given a sequence of n *x*'s and n *o*'s and are asked to connect each *x* to one *o* using the minimum length of wire. Assume the distance between adjacent *x*'s and *o*'s is one. Devise a greedy algorithm to solve this problem.

x o o x x x o x o x o o

J.2 (2) Matrix-vector multiplication can rewritten using divide-and-conquer to four matrix-vector multiplications of size $n/2$ where the resulting vectors must then be added together. Like the original operation, the run time is still $\Theta(n^2)$. What did we do in the case of the fast Fourier transform to reduce the run time of the divide-and-conquer algorithm to $\Theta(n \ln(n))$?

J.3 (4) You have just come up with an integer multiplication algorithm that satisfies

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ 5T\left(\frac{n}{4}\right) + \Theta(n) & n > 1 \end{cases}$$

Show how you can take the expression

$$T(n) = T\left(b^m\right) = a^m \sum_{\ell=0}^{m} \left(\frac{b^k}{a}\right)^{\ell}$$

to deduce the run time of your integer multiplication problem. Is your algorithm faster or slower than the Karatsuba algorithm we saw in class, which ran in $\Theta(n^{\lg(3)})$?

J.4 (2) In discussing dynamic programming, we also discussed bottom-up and top-down design techniques. The merge sort algorithm, as discussed in class, is an example of a top-down design. How would you implement merge sort using a bottom-up design?

J.5 (4) Devise an algorithm that approximates the value of a function on a given subset of the real line. You will pick *n* random numbers on the interval [*a*, *b*] and for each of those numbers, *x*, if valid(*x*) returns true, we will use that value to estimate the average value. You will use the valid numbers to estimate the average as described in class. If none of the points are in the interval, `return nan()`. You may wish to recall quickly the means of finding a double on the interval [*a*, *b*].

```
      double x = a + drand48()*(b - a);

double average( double f( double ), bool valid( double ),
                int n, double a, double b ) {
```

K. Theory of computation

K.1 (3) The set of all NP Complete problems have been shown to be polynomially reducible to each other. Explain how finding a polynomial-time algorithm to one NP Complete problem will allow us to find polynomial-time algorithms to all other NP Complete problems.

L Bonus Questions (If you attempt both, cross out the one we should not mark, otherwise, we will mark L.1.)

L.1*a* (2) What sets are represented by this disjoint set?

0	1	2	3	4	5	6	7	8	9	10	11
6	3	9	3	11	1	6	3	7	9	0	11

L.1*b* (2) Describe how you quickly test whether or not two entries in a disjoint set belong to the same set.

L.2*a* [1] Is a splay tree always going to be balanced, in the sense that the height is always $\Theta(\ln(n))$? (Yes or No)

L.2*b* [3] Demonstrate how the splay tree in Figure 2 would change if a search is made for the number 13.

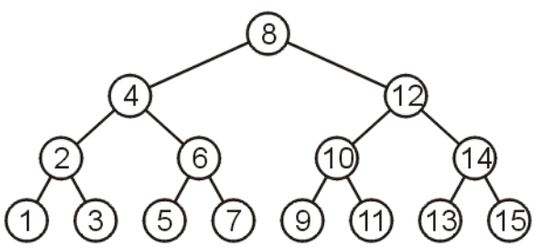

Figure 2. A splay tree.