Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

CSI2110/CSI2510

Data Structures and Algorithms

Final Examination

Length of Examination: 3 hours **December 13th, 2011**

Professor: P. Flocchini, J. Sabourne

Last name: ____________________

First name: ____________________

Student number: ____________________

Signature: ____________________

Closed Books.

Please answer in the space provided (in this questionnaire).

No calculators or other electronic devices are allowed.

At the end of the exam, when time is up:

- Stop working and turn your exam upside down.
- Remain silent.
- Do not move or speak until *all* exams have been picked up, and a TA or a Professor gives the go-ahead.

Page	Marks of each page
PAGE 2	out of 5.5
PAGE 3	out of 6
PAGE 4	out of 5
PAGE 5	out of 2.5
PAGE 6	out of 5
PAGE 7	out of 3.5
PAGE 8	out of 3.5
PAGE 9	out of 1.5
PAGE 10	out of 3
PAGE 11	out of 3.5
PAGE 12	out of 2
PAGE 13	out of 4
TOTAL	out of 45

Question 1 [2 points] What is the running time complexity of the algorithms in the following pseudo-code fragments (in big-Oh notation) ? Note. in the following n is considered big (say $n > 100$).

Algorithm Hello(A)
Let A be an array of size n.
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 10$ **to** n^2 **do**
    $A[i] \leftarrow j$

a) $O(\log n)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3)$ e) $O(n^4)$

Algorithm GoodLuck(n)
$i \leftarrow 1$; $j \leftarrow 1$;
while $i \leq n$ **do**
  { $j \leftarrow j + 3$;
  $i \leftarrow i * 2$;}

a) $O(\log n)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3)$ e) $O(n^4)$

Question 2 [2 points] What is the worst case big-Oh complexity of deleting a key k in the following data structures (as a function of the size n of the data structure):

1. MIN-HEAP: *a)* $O(1)$ *b)* $O(\log n)$ *c)* $O(n)$ *d)* $O(n \log n)$
2. AVL TREE: *a)* $O(1)$ *b)* $O(\log n)$ *c)* $O(n)$ *d)* $O(n^2)$
3. HASH TABLE: *a)* $O(1)$ *b)* $O(\log n)$ *c)* $O(n)$ *d)* $O(n^2)$
4. SORTED SEQUENCE (ARRAY IMPLEMENTATION):
   a) $O(\log n)$ *b)* $O(n)$ *c)* $O(n \log n)$ *d)* $O(n^2)$

Question 3 [1.5 point] You would like to design an algorithm for transforming any AVL tree of size $n > 3$ into a Min-heap. For each of the following procedures indicate whether it is correct or not and, if correct, write its big-Oh worst case complexity.

1. perform a post-order traversal of the AVL tree and insert the keys obtained in this order directly into an array that represent a heap, starting from index 1.
   CORRECT NOT CORRECT Complexity:
2. read the keys from the AVL tree in an arbitrary order and insert them, one by one, into an initially empty Heap by restructuring the heap at each step.
   CORRECT NOT CORRECT Complexity: O(n log n)
3. read the keys from the AVL in pre-order and perform bottom-up heap construction.
   CORRECT NOT CORRECT Complexity: O (n)

Question 4 [1.5 points] Suppose that Quicksort in-place is used to sort the following array where the pivot is always chosen to be the last number. Before recursively calling Quicksort in-place, the keys must be partitioned around the pivot. Write the content of the array after each swap, using the PARTITION algorithm seen in class.

note: I am only showing the changes

- - This is the initial array (with 18 as pivot):

1	2	3	4	10	15	25	20	1	30	5	18

- - Step 1:

						5				25	18

- - Step 2:

							1	20			18

- - Step 3:

								18			20

Question 5 [1.5 points] You are given the following sequence of values $\{1, 3, 5, 7, 2, 4, 6, 8, 9, 10\}$ stored in an array A (with $A[1]$ containing 1, $A[2]$ containing 3, etc). Write the content of the array after performing a **bottom-up heap construction** to construct a **Max-heap**.

index	1	2	3	4	5	6	7	8	9	10
value	10	9	6	8	3	4	5	1	7	2

Question 6 [1.5points] A hash-map of size 13 has been constructed with DOUBLE-HASHING by applying $h_j(k_i) = [h(k_i) + jd(k_i)] \bmod 13$. The primary hashing function is $h(k_i) = k_i \bmod 13$ and the secondary hashing function is $d_i(k_i) = k_i$ div 13 where *div* is integer division. Perform `Insert(28)` and mark in the hash-map below the cells which will be probed.

i =	0	1	2	3	4	5	6	7	8	9	10	11	12
		27	15	14	17	available	6			35	18	25	

Question 7 [1.5points] A hash-map of size 11 has been constructed with quadratic hashing by applying $h(k_i) = (3k_i - 2) \bmod 11$. Perform `Find(23)` and mark in the hash-map below the cells which will be probed.

i =	0	1	2	3	4	5	6	7	8	9	10
		1	5	9	2	available	10		7		

Question 8 [3 points] Consider the Dijkstra algorithm for finding the shortest path spanning tree of a graph. Execute the algorithm for the undirected graph represented by the following adjacency list, starting from node A. (The numbers in parenthesis are the weights of the corresponding edges).

$A \to B(4), C(5)$
$B \to A(4), D(3)$
$C \to A(5)D(1), E(3)$
$D \to B(3), C(1), E(1)$
$E \to C(3), D(1)$

Draw the graph. Fill the chart below to keep track of the changes of the distance labels after including each new node to the cloud. The first line of the chart is already filled with the initial distance labels.

new vertex	B	C	D	E	new edge
A	4	5	∞	∞	—
B			7		(A,B)
C			6	8	(A,C)
D				7	(C,D)
E					(D,E)

Note: only showing updates.

Question 9 [2 points] The Boyer-Moore algorithm is used to find the pattern P in the string T. Indicate the next 6 comparisons performed by the algorithm in the table below. The first comparison is already filled out.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T =	T	h	e		r	i	n	g		i	s		s	h	i	n	i	n	g
P =	s	h	i	n	i	n	g												

Comparison #	i	j	$T[i]$	$P[j]$
1	6	6	n	g
2	7	6	g	g
3	6	5	n	n
4	5	4	i	i
5	4	3	r	n
6	11	6	-	g
7	18	6	g	g

Question 10 [0.5 point] Let G be a graph with n vertices and m edges. What is the time complexity of the depth-first search of G, if G is represented by an adjacency matrix ?

O(n^2)

Question 11 [1.5 points] Let G be a graph with n vertices and $\frac{n^2}{3}$ edges represented with an adjacency list. You want to employ Prim's algorithm to compute the Minimum Spanning tree of G. Which data structure is preferable to represent the priority queue used by the algorithm: a **Heap** or an **Unsorted Sequence** ?

Unsorted sequence

Give the corresponding worst case big-Oh complexity (as a function of n) of the algorithm using your chosen stucture as Priority Queue.

O(n^2)

Extra explanation why: there are O(n) RemoveMin operations each O(n) plus O(m) ReducePriority operations each O(1) - so O(n*n + m)=O(n^2)

Extra explanation:
I we used a heap, we would get O((n+m) log n), and since m=O(n^2) here, we would get complexity O(n^2 log n); in this case, unsorted sequence is better.

Question 12 [0.5 point] Which of the two traversals (BFS or DFS) of a heap - starting from the root - would return the keys in the order they appear in the array representation of the heap ?

BFS

Question 13 [3 points] True/False

Finding a key in a hash table is **always** faster than finding a key in a Binary Search Tree. True ☐ False ☐

Reason: worst case for hash is O(n) while BST O(log n)

In a graph where all the weights are equal, breadth-first search and depth-first search will visit the nodes in the same order (when starting from the same node). True ☐ False ☐

Certainly not; many examples where this is not the case.

a vertex or an adge

Searching for an item in a connected graph can always be done in $O(m)$ (where m is the number of edges), choosing a good representation for the graph. True ☐ False ☐

Use adjacency list data structure (that more complete structure that also has list of all edges and list of all vertices): searching for vertex O(n) is O(m) for connected graphs, search for edge is O(m).

A binary search tree with all leaves at the same level is also a 2-4 tree. True ☐ False ☐

Here it depends: answer 1: false, because some nodes in BST may have one child only.
answer 2: assuming BST tree is full (always accomplished by dummy nodes used in text), true.

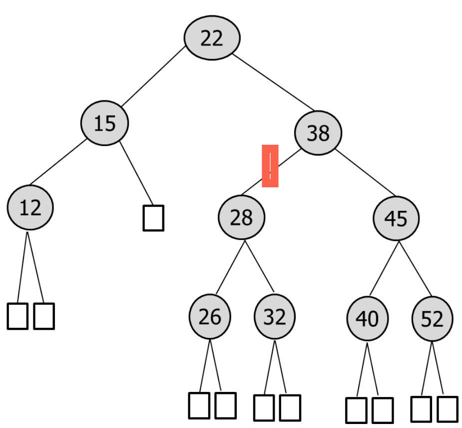
Finding the node with maximum out-degree in a directed graph represented with adjacency matrix costs $O(n^2)$. True ☐ False ☐

For each vertex (row), count number of 1's in that row

Finding whether an edge (u, v) exists in a graph implemented by an adjacency list is $O(n^3)$. True ☐ False ☐

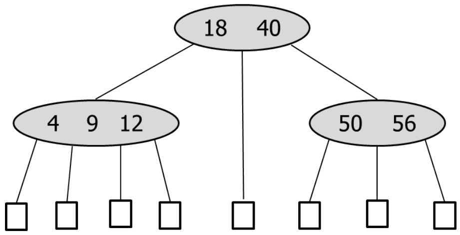
Go to the list of u and search for v; this is worst case O(n), which by the way is O(n^3).

Question 14 [2 points] Insert key 30 in the following AVL tree using the algorithm seen in class. Indicate x, y, z, rename them a, b, c, and draw the resulting tree after rebalancing (if necessary).



(see answer in separate file)

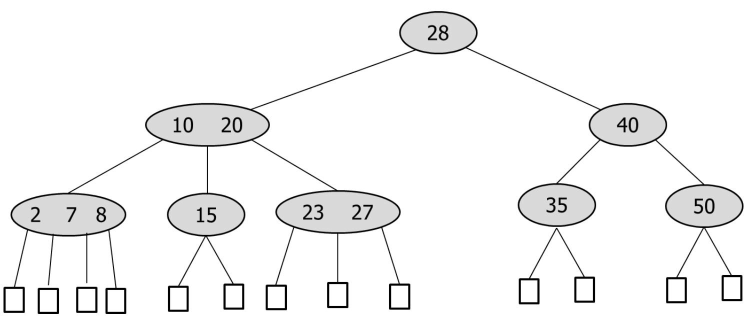
Question 15 [1 point] Reply with True or False:



This tree is a generalized search tree True ☐ False ☐

This tree is a (2,4) tree True ☐ False ☐ not all leaves at the same level

Question 16 [2.5 points] Delete 35 from the following (2,4) tree. Draw the resulting tree after rebalancing (if necessary).



(see answer in separate file)

What is the worst case complexity (in big Oh notation) of a key removal in a (2,4) tree that contains n keys ?

O(log n)

Question 17 [1.5 points] Perform the first phase of Bubblesort for the following sequence. Write the sequence obtained after each comparison:

10	24	18	39	7	31

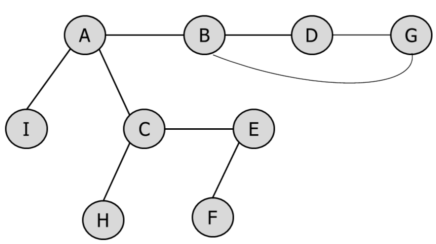
10	24	18	39	7	31
10	18	24	39	7	31
10	18	24	39	7	31
10	18	24	7	39	31
10	18	24	7	31	39

Question 18 [0.5 point] What is the running time complexity of the quick-sort algorithm if we are choosing the biggest element of each subsequence as its pivot

a) $O(log(n))$ b) $O(n)$ c) $O(nlog(n))$ d) $\boxed{O(n^2)}$ e) none of the above

Note for section csi2110A - Chooses Julien at each iteration :-)))

Question 19 [1.5 points] Reply with True or False:



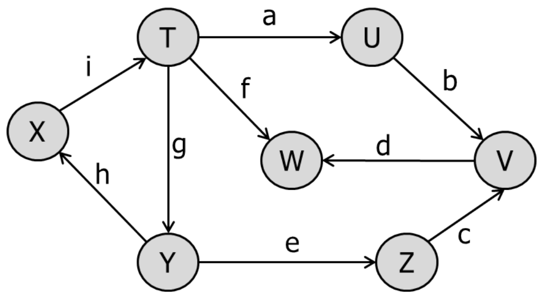
This graph is connected True ☐ False ☐

This graph is acyclic True ☐ False ☐

We can find 3 different spanning trees for this graph True ☐ False ☐

3 choices for which edge of the unique cycle is not in the spanning tree; all other edges are forced.

Question 20 [1.5 points] Fill the adjacency matrix, the adjacency list and the edges list below corresponding to the the following directed graph. Whenever you need to, list nodes and/or edges in *alphabetic order*.



a) Adjacency Matrix

	T	U					
T	0	1					

(complete the rest...
we could also have M[T,U]=a instead of 1
(a pointer to edge info for edge a)

b) Adjacency List

T	---> U, a --> W, f --> Y, g

(please complete the rest)

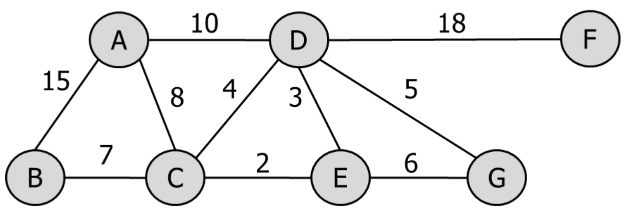
c) Edges List

(please complete the rest)

	b	c	d	e	f	g	h	i

T	U	V	W	X	Y	Z

Question 21 [3 points] Find the minimum spanning tree for the following graph using the Kruskal algorithm.



Fill the following table with the chosen edges:

Chosen edges
CE
ED
DG
CB
CA
DF

What is the total weight of this tree? 43

Question 22 [2 points] We want to sort the integers shown in the table below using their representation in base 3 and the radix-sort algorithm. Complete the table showing each pass of the bucket-sort.

Decimal	Base 3	Pass 1	Pass 2	Pass 3	Pass 4
30	1 0 1 0	1010	1001	1001	0002
2	0 0 0 2	0120	2201	0002	0120
28	1 0 0 1	1001	0002	1010	1001
15	0 1 2 0	2201	1010	0120	1010
73	2 2 1 0	0002	0120	2201	2201

Question 23 [1 point] Given the following sequence:

5	25	36	38	41	49	50

Which of the following algorithms sorts this sequence the fastest:

a) Bubble-sort b) Quick-sort c) Merge-sort d) Selection-sort

What is the running time complexity of the algorithm you chose when the initial sequence is already sorted ?

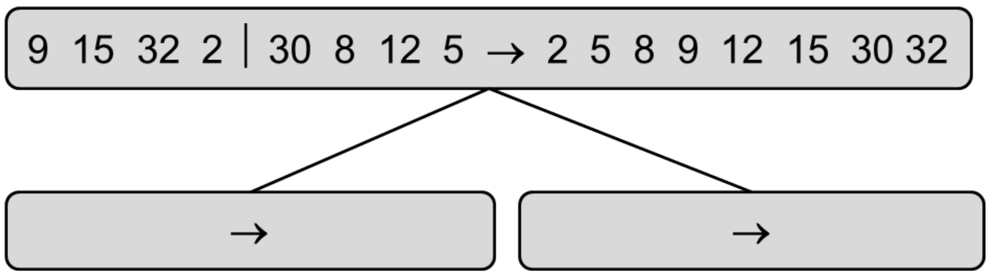
a) $O(log(n))$ b) $O(n)$ c) $O(nlog(n))$ d) $O(n^2)$ e) none

Question 24 [0.5 point] Given a **connected directed** graph with n vertices and m edges ; Which of these statements is always correct:

a) $(n-1)/2 \leq m \leq (n) \cdot (n-1)/2$
b) $(n-1) \leq m \leq (n) \cdot (n-1)$
c) $(n-1)/2 \leq m \leq (n) \cdot (n-1)$
d) $(n-1) \leq m \leq (n) \cdot (n-1)/2$
e) none of the above

Question 25 [2 points] Draw the merge-sort tree with the following array:
Remark: Only the nodes for the first partition are shown.

please do this on your own; there is a similar example in the slides



Question 26 [3 points] For a graph with n vertices and m edges what is the big-Oh runtime cost of the following methods:

	Adjacency Matrix (implemented with a 2D array)	Adjacency List (implemented with an array of linked lists)	Edge List (implemented with 2 arrays)
`insertVertex( v )`	O(n^2) if in the middle O(n) if at the end	O(n) if in the middle; O(1) if at the end	O(n) if in the middle O(1) if at the end
`removeVertex( v )`	O(n^2)	O (n+m)	O(n+m)

Question 27 [1 point] The `DTrav(G,v)` method returns a sequence S of vertices visited during a depth first search traversal of the undirected graph G starting at a vertex v. The `SeqEq(S1,S2)` method returns a boolean indicating if the the two sequences $S1$,$S2$ contain the same elements.The `G.Vertices` method returns a sequence containing all the vertices of the graph G.

What does the algorithm `Unknown` described by the following pseudo-code indicate?

Algorithm `Unknown`

```
For a random vertex w of G
if SeqEq(DTrav(G,w),G.Vertices)
    return True
return False
```

Return id a graph is connected or not.