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 Exam

Duration: 3 hours
December 7^{th}, 2013
Professors: L. Moura, R. Laganière and H. Al Osman
Page 1 of 13

Family name: _____

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	Nombre de Points
PAGE 2	out of 4
PAGE 3	out of 5
PAGE 4	out of 7
PAGE 5	out of 3
PAGE 6	out of 2
PAGE 7	out of 5
PAGE 8	out of 6
PAGE 9	out of 8
PAGE 10	out of 2
PAGE 11	out of 6
PAGE 12	out of 0
PAGE 13	out of 2
TOTAL	out of 50

Question 1 [2 points] What are the running time complexities of the algorithms in the following pseudo-code fragments (in big-Oh notation)? Choose the best big-Oh estimates if more than one answer is possible.
Note, in the following, n is the input size, which we consider to grow for the big-Oh analysis.

Algorithm IAMDUMMY(A)
Let A be an array of size n
for i $\leftarrow$ 0 to $(n-1)$ **do**
    for j $\leftarrow$ 1 to 3^{i} **do**
        A[i] $\leftarrow$ j*j
    end for
end for

a) $O(n \log n)$ *b)* $O(n^3)$ *c)* $O(n^6)$ *d)* $O(3^n)$ *e)* $O(j \bmod n)$ *f)* $O(n2^n)$

Algorithm AmIDUMMY(A)
Let A be an array of size n
for i $\leftarrow$ 0 to $(n * n * n)$ **do**
    for j $\leftarrow$ 0 to (i/2) **do**
        A[j mod n] $\leftarrow$ j
    end for
end for

a) $O(n^3 \log n)$ *b)* $O(n^3)$ *c)* $O(n^6)$ *d)* $O(3^n)$ *e)* $O(j \bmod n)$ *f)* $O(n2^n)$

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

1. Unsorted sequence (array implementation where maximum array size is never exceeded)
   a) O(1) b) O(log n) c) O(n) d) O(n log n)
2. Queue implemented with a doubly linked list with beginning and end pointers:
   a) O(1) b) O(log n) c) O(n) d) O(n log n)
3. Binary search tree:
   a) O(1) b) O(log n) c) O(n) d) O(n log n)
4. 2-4 tree:
   a) O(1) b) O(log n) c) O(n) d) O(n log n)

Question 3 [3 points]

1. (2 pt) Draw all possible **binary search trees** that can store keys 1, 2, 3, and for each one indicate if it is or it isn't an AVL tree:

2. (1 pt) Draw all possible 2-4 trees that can store keys 1, 2, 3, 4, 5, 6.

Question 4 [2 points] You are given the following array A which represents a max-Heap:

Index i:	1	2	3	4	5	6	7	8
A[i]:	20	18	19	4	2	3	5	1

a) Write the content of the array A after performing **removeMax()**.

Index i:	1	2	3	4	5	6	7
A[i]:							

b) Write the content of the array after performing **Add(25)** to the **original** max-Heap A.

Index i:	1	2	3	4	5	6	7	8	9
A[i]:									

Question 5 [1 point] Draw the binary tree where each node contains a letter and its in-order traversal prints "ABCDE" and its post-order traversal prints "ACBED"

Question 6 [1 point] Suppose you have a hash map of size 31 that has a load factor of 0.75. In order to reduce the load factor to 0.5, you decide to re-hash your elements into a new hash table. What would be a good choice for the new table in order to ensure a load factor of 0.5 and also avoid collisions.

Question 7 [1 point] A hash map of size 15 has been constructed using Cubic hashing by applying $h_j(k_i) = [h(k_i) + j^3] \text{mod } 15$. Perform `Insert(33)` and show the resulting table.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Value	21	16		48	79		36	81		43		11		28	44

Question 8 [1 point] A hash map of size 13 has been constructed using $h(k) = k \text{ mod } 13$ and linear probing for collision resolution. Assuming no elements have been deleted in the table, indicate in what order the following keys could have been inserted.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Value		27	14			31	58	57	34	135			

Question 9 [3 points] Consider the graph given in Figure 1 and consider the following adjacency list representation of the graph. Taking into account the order of the nodes in each adjacency list and starting from node A,

1. give, in order, the list of visited nodes in the case of a depth-first search;

2. give, in order, the list of visited nodes in the case of a breath-first search.

Node	list of adjacent nodes
A	C
B	C F
C	E D B
D	F E
E	F D
F	B D E

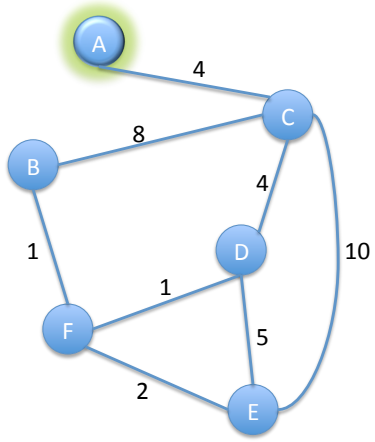


Figure 1: A weighted graph

Question 10 [1 point]

You want to replace the value associated with each node of a Torus graph (i.e. all nodes in the graph have a degree of 4) by the sum of the weights of its incident edges. You do this by visiting each node of the graph and update its value by summing the edges' weight. What would be the complexity of this algorithm in terms of the **exact number of addition operations** as a function of the number of nodes in the graph. *Note:* adding $a+b+c+d$ requires three additions.

Question 11 [2 point]

Consider the following array of elements

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

1. Apply a partition on this array by using the value 6 as a pivot (using algorithm shown below).

2. Apply a recursive Quicksort algorithm (based the partition algorithm described below). Always use the first element of each sequence sent to the partition as the pivot. Show the state of the array at each step of the algorithm.

```
private  static void sortByPivot(int[] a, int pivot)
{
    int i =0 , j = a.length -1;

    while(true) {

            while(a[i] <= pivot) i++;

            while( j >0 && a[j] >= pivot) j--;

            if(i <= j) {
                break;
            } else {
                swap(a,i,j); // swap values at indices i and j of array a
            }
    }
}
```

Question 12 [1 point] Suppose you have an array of elements sorted in increasing order. You now use a sort algorithm to sort them in decreasing order. What would be the *big Oh* complexity of this sort if:

- Quicksort is used (with pivot in partition being the first element of the sequence).
- Insertion sort is used
- Selection sort is used
- Merge sort is used

Question 13 [1 point] You want to sort integer numbers in the range $[0, 999]$ by using a 3-tuple lexicographic sort based on the 3 digits of the numbers (base 10). In each of the 3 iterations of the lexicographic sort, the MergeSort algorithm is used.

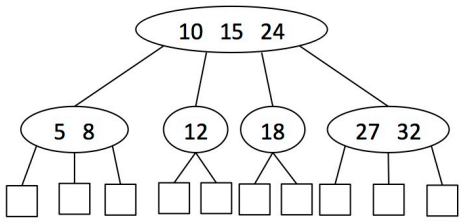
1. To perform this lexicographic search, should we start by the least significant digit or by the most significant digit? Justify your answer.

2. What would be the *big Oh* complexity of the described algorithm?

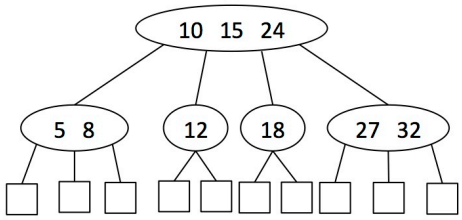
3. Would it be a good idea to use Quicksort instead of MergeSort? Explain.

Question 14 [3 points] Consider the following 2-4 tree:

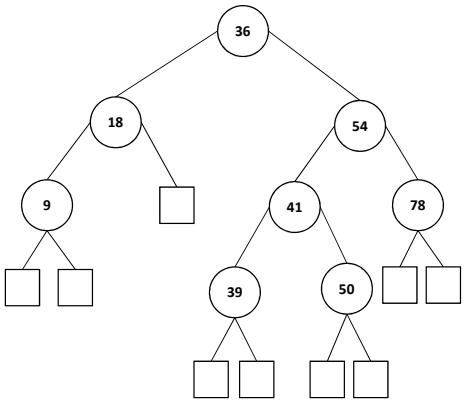
1. (1 point) Insert 2 into the following 2-4 tree and show the resulting tree beside it.



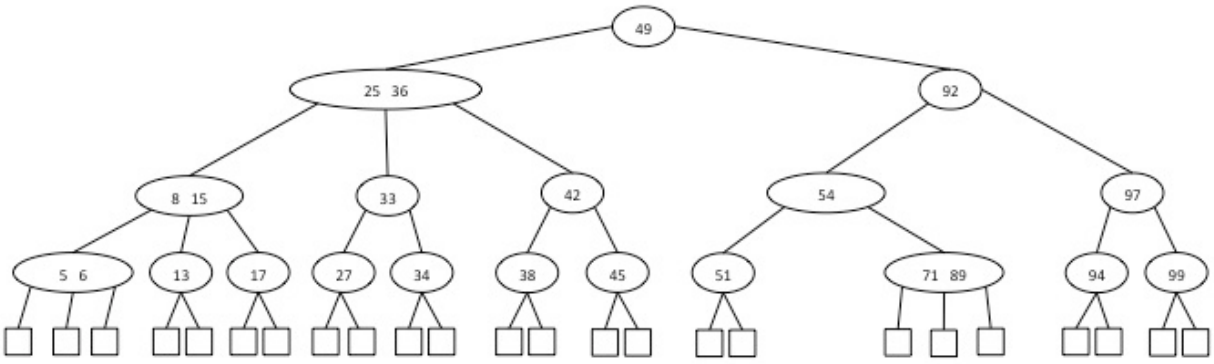
2. (2 points) Delete 12 from the following 2-4 tree and show the resulting tree beside it.



Question 15 [2 points] Insert key 53 into the AVL tree below using the algorithm seen in class. Indicate x, y, z and rename them a, b, c, indicate the sub-trees T_0, T_1, T_2 and T_3, and draw the resulting tree after rebalancing beside it (if necessary).



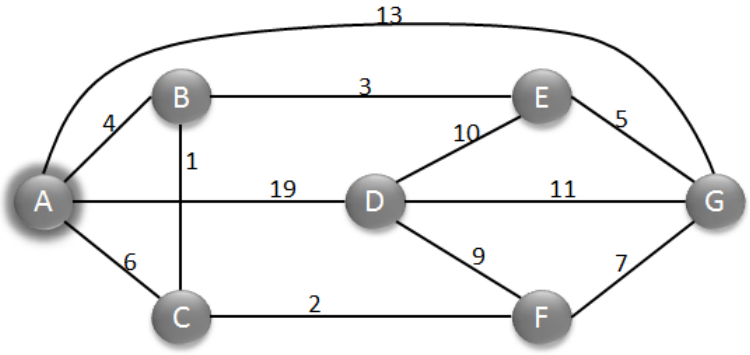
Question 16 [3 points] Remove key 99 in the following (2,4) tree and draw the resulting tree after reorganization in the space below.



Question 17 [3 points] Suppose you have a deque D containing the numbers (1,2,3,4,5,6,7,8) in this order. Suppose further that you have an initially empty queue Q. Give a pseudo-code description of a method that uses only D and Q (and no other variables or objects) and results in Q storing the elements (1,2,3,4,5,6,7,8), in this order.

The methods provided for each ADT are listed below:
`deque:` `addFirst(e), addLast(e), removeFirst(), removeLast(), size(), isEmpty()`
`queue:` `enqueue(e), dequeue(), size(), isEmpty()`

Question 18 [3 points] For the graph given in the figure below, use the Dijkstra algorithm to find the shortest path spanning tree of the graph starting from node A.



Fill the table 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	A	B	C	D	E	F	G	New Edge
A	0	4	6	19	∞	∞	13	–

Question 19 [3 points] For the same graph of question 18 and using Kruskal's algorithm for finding the minimum spanning tree of the connected graph:

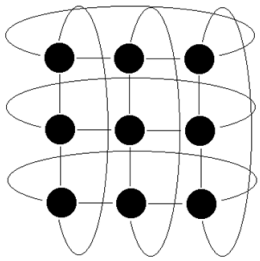
1. Write the list of edges in the order they are chosen to be part of the spanning tree.

2. What is the total weight of the spanning tree?

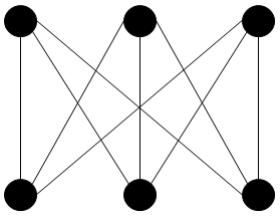
3. What is the complexity of the algorithm as a function of n and m?

Question 20 [2 points] In this question, all graphs are implemented using an Adjacency List structure.

1. For a 2D torus (an example is shown in the figure below), what it the big Oh complexity of Prim-Jarniks algorithm for the computation of the Minimum Spanning Tree. The complexity should be computed as a function of the number of vertices only.



2. For a complete and balanced bipartite graph (an example is shown in the figure below), what is the big Oh complexity of Kruskals algorithm for the computation of the Minimum Spanning Tree. The complexity should be computed as a function of the number of vertices only. Note that the vertices of a complete and balanced bipartite graph are divided into two sets containing equal number of vertices. Each vertex from one set connects only to all other ones from the other set.



Question 21 [2 points] The Boyer-Moore algorithm is used to find a pattern P in a 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	19	20	21	22	23	24
T =	a	-	c	a	n	a	r	y	-	o	n	-	a	-	b	i	n	a	r	y	-	t	r	e	e
P =	b	i	n	a	r	y																			

Comparison #	i	j	$T[i]$	$P[j]$
1	5	5	a	y
2				
3				
4				
5				
6				
7				

Question 22 [3 points]

True or False Questions:

1. A spanning tree of a graph G contains all the vertices of G.
   TRUE **FALSE**

2. A minimum spanning tree rooted at vertex u represents the shortest path between u and all other vertices in the graph.
   TRUE **FALSE**

3. Kruskals algorithm for finding the minimum spanning forest assumes that the input graph G is connected
   TRUE **FALSE**

Short Answers Questions:

1. How many connected components a connected graph has?

2. For a Text of length n and a Pattern of length $n/4$, what is the big Oh complexity of the brute force pattern matching algorithm in terms of n only?

3. For a graph with n vertices and $2n$ edges, what is the big Oh complexity of Dijkstra's algorithm for finding the shortest paths (from a given node) in terms of n only?

Question 23 [2 points] Consider the Knuth-Morris-Pratt Algorithm to find pattern P in string T.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T =	b	a	c	c	b	c	a	a	b	a	c	c	b	a	c
P =	b	a	c	c	b	a	c								

- Compute the failure function for pattern P by completing the table provided below:

j	0	1	2	3	4	5	6
P[j]	b	a	c	c	b	a	c
F[j]							

- When the first mismatch occurs between T[i] and P[j], what are the values of:

  i= j= T[i]= P[j]=

- At the next comparison after the first mismatch, what are the values of:

  i= j= T[i]= P[j]=

Question 24 [3 points] The algorithm `isConnected` tests whether a graph is connected and returns a Boolean (`true` if the graph is connected and `false` otherwise). In the body of the `isConnected` algorithm, Dijkstra's Shortest Path algorithm is called. The result obtained from `DijkstraShortestPath` call should be used to check if the graph is connected.

The algorithm `DijkstraShortestPath` takes a graph G and a vertex v as parameters and returns a sequence D specifying the distance from v to each vertex in the graph.

Complete the body for the algorithm `isConnected`. Note that you will only complete **Pseudocode Snippet 1**. **Pseudocode Snippet 2** is provided for reference only.

Pseudocode snippet 1 (complete this algorithm)

```
Algorithm isConnected (G):

    Input: A weighted graph G.
    Output: True if the graph is connected, false otherwise.

    Let v be a vertex in graph G.
    connected = true
    D = DijkstraShortestPath(G, v)
    //add your code below









    return connected
```

Pseudocode snippet 2 (provided for reference only)

```
Algorithm DijkstraShortestPath(G, v):

    Input: A weighted graph G and a vertex v of G.
    Output: A label D[u], for each vertex that u of G, such that D[u]
    is the length of a shortest path from v to u in G.

    initialize D[v] = 0 and D[u] = INFINITY for each vertex u != v

    let Q be a priority queue that contains all of the vertices of G
    using the D labels as keys.

    while !Q.isEmpty () do
        u = Q.removeMinElement()
        for each vertex z adjacent to u such that z is in Q do
            if D[u] + w((u, z)) < D[z] then
                D[z] = D[u] + w((u, z))
                Change the key value of z in Q to D[z]

    return the label D[u] of each vertex u.
```