

## Part 1: Multiple Choice

Enter your answers on the Scantron sheet.

We **will not** mark answers that have been entered on this sheet.

Each multiple choice question is worth **2.5 marks**.

In all the questions,  $\log(x)$  means  $\log_2(x)$ . You might find this fact useful:  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$ .

1. Let  $G = (V, E)$  be an undirected, connected graph with  $n$  vertices and  $m$  edges. All vertices are initially un-marked. Consider the following algorithm:

**Algorithm**  $\text{traverse}(G, u)$

**Input:** Undirected, connected graph  $G$ , and vertex  $u$  of  $G$ .

Mark  $u$

$c \leftarrow 0$

**For** each edge  $(u, v)$  incident on  $u$  **do** {

**For** each vertex  $w$  of  $G$  **do**  $c \leftarrow c + 1$

**if**  $v$  is not marked **then**  $c \leftarrow c + \text{traverse}(G, v)$

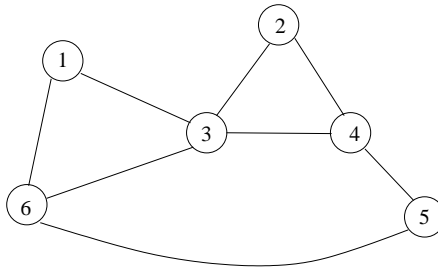
}

**return**  $c$

Assume that  $G$  is stored in an adjacency list. What is the time complexity of the above algorithm in the worst case?

- (A)  $O(m)$   
(B)  $O(n + m)$   
(C)  $O(n^2)$   
(D)  $O(nm)$   
(E)  $O(n^3)$
2. Consider the following graph. Which one of the following is a valid ordering of the vertices if the graph is traversed using breadth first search (BFS)?

- (A) 1, 2, 6, 3, 5, 4  
(B) 4, 3, 2, 1, 5, 6  
(C) 2, 3, 1, 4, 6, 5  
(D) 3, 6, 4, 1, 5, 2  
(E) 5, 4, 6, 2, 3, 1



3. Let  $G = (V, E)$  be an undirected graph. Initially all vertices of  $G$  are un-marked and all edges are un-labelled. Consider the following algorithm.

**Algorithm**  $B(G, u)$

**Input:** Undirected graph  $G$ , and vertex  $u$  of  $G$ .

```

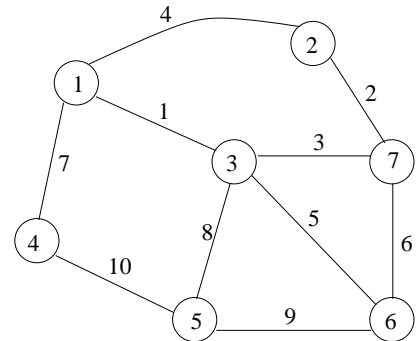
Mark  $u$ 
For each edge  $(u, v)$  incident on  $u$  do {
    if  $(u, v)$  is not labelled then {
        if  $v$  is not marked then {
            Label  $(u, v)$  as discovery
            if  $B(G, v) = true$  then return true
        }
    }
else return true
}
return false

```

What does the algorithm do?

- (A) It returns *true* if none of the edges incident on  $u$  is labelled as *discovery* and it returns *false* otherwise.  
 ✓(B) It returns *true* if  $G$  has at least one cycle and it returns *false* otherwise.  
 (C) It returns *true* if  $G$  is connected and it returns *false* otherwise.  
 (D) It returns *true* if  $u$  has at least one edge labelled *discovery* and it returns *false* otherwise.  
 (E) It returns *true* if any vertex  $v$  is marked and it returns *false* otherwise.
4. Consider the following graph. Which edges, and in which order, are selected by Prim's algorithm if it starts at vertex 1?

- (A) (1,3), (3,7), (2,7), (7,6), (1,4), (3,5)  
 (B) (1,3), (2,7), (3,7), (3,5), (7,6), (1,4)  
 ✓(C) (1,3), (3,7), (2,7), (3,6), (1,4), (3,5)  
 (D) (1,3), (2,7), (3,7), (1,4), (3,6), (3,5)  
 (E) (1,3), (3,7), (2,7), (3,5), (7,6), (1,4)



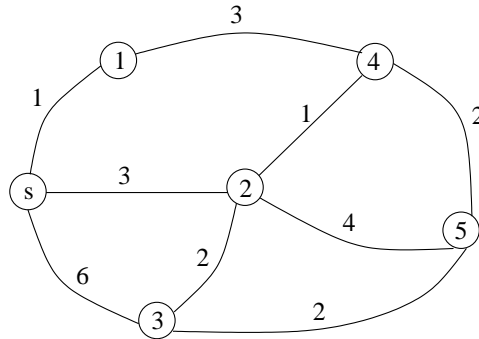
5. Suppose that  $A$  is an array storing  $n$  identical integer values. Which of the following sorting algorithms has the smallest running time when given as input this array  $A$ ?

- ✓(A) Insertion Sort  
 (B) Selection Sort  
 (C) Ordered-dictionary sort implemented using an AVL tree.  
 (D) Ordered-dictionary sort implemented using a (2,4)-tree.  
 (E) Ordered-dictionary sort implemented using a hash table with separate chaining.

As we discussed in class, ordered-dictionary sort first stores the values in an ordered dictionary and then repeatedly removes one element of smallest value from the dictionary and inserts it in the next position of the array.

6. Consider the following graph. Assume that we use Dijkstra's algorithm to find shortest paths from vertex  $s$  to the other vertices in the graph. In which order are the final distance labels  $u.d$  computed? (Or in other words, in which order are the shortest paths computed?)

- (A)  $s, 1, 2, 3, 4, 5$   
 (B)  $s, 1, 2, 4, 5, 3$   
 (C)  $s, 1, 4, 2, 3, 5$   
 ✓(D)  $s, 1, 2, 4, 3, 5$   
 (E)  $s, 1, 4, 3, 2, 5$



7. Consider the following pairs of functions  $f(n), g(n)$ . For which pair the functions are such that  $f(n)$  is  $O(g(n))$  and  $g(n)$  is not  $O(f(n))$ ?

- (A)  $f(n) = n^3, g(n) = n^2 \log(n^2)$   
 (B)  $f(n) = \log n, g(n) = 10 \log n$   
 ✓(C)  $f(n) = 1, g(n) = \log n$   
 (D)  $f(n) = n^2, g(n) = 10n \log n$   
 (E)  $f(n) = n^3, g(n) = n^3 + n$

8. A very large collection of data items is to be stored in an ordered dictionary that resides in the hard disk of a computer. Which of the following data structures provides the most efficient implementation in the worst case for the ordered dictionary?

- (A) A (2,4)-tree.  
 (B) A hash table that uses separate chaining.  
 (C) An AVL tree.  
 ✓(D) A B-tree.  
 (E) A sorted array.

9. The following values are inserted, in the given order, in a hash table of size 7 that uses linear probing and hash function  $h(k) = k \bmod 7$ :

4, 11, 5, 12, 6.

In which entry of the table is the key 6 stored?

- (A) 0  
 ✓(B) 1  
 (C) 4  
 (D) 5  
 (E) 6

10. Consider the following algorithm:

**Algorithm**  $T(r)$

**Input:** Root  $r$  of a proper binary tree.

```
if  $r$  is a leaf then return 0
else {
     $p \leftarrow T(\text{left child of } r)$ 
     $q \leftarrow T(\text{right child of } r)$ 
    if  $p > q$  then return  $p + 1$ 
    else return  $q + 1$ 
}
```

What does the algorithm compute?

- (A) The number of nodes in the tree.
  - (B) The number of internal nodes in the tree.
  - (C) The number of descendants of  $r$ .
  - (D) The number of nodes in the largest subtree of  $r$ .
  - ✓(E) The height of the tree.
11. How many different  $(2, 4)$ -trees containing the keys 1, 2, 3, 4, and 5 exist (each key must appear once in each one of these  $(2, 4)$ -trees)?
- (A) 2
  - (B) 3
  - ✓(C) 4
  - (D) 5
  - (E) 6
12. Let  $G = (V, E)$  be an undirected graph. We are interested in selecting a data structure for representing  $G$  that allows us to implement the operations `areAdjacent(u, v)` and `incidentEdges(u)`. Let  $n$  be the number of vertices and  $m$  be the number of edges. Which of the following is true?
- (A) An edge list allows us to perform `incidentEdges(u)` in  $O(\text{degree}(u))$  time and `areAdjacent(u, v)` in  $O(1)$  time.
  - (B) With an adjacency list `areAdjacent(u, v)` can be implemented in  $O(1)$  time and `incidentEdges(u)` in  $O(\text{degree}(u))$  time.
  - (C) An adjacency matrix allows us to implement both operations in  $O(1)$  time.
  - (D) An adjacency list requires  $O(1)$  time for implementing `incidentEdges(u)` and  $O(1)$  time for implementing `areAdjacent(u, v)`.
  - ✓(E) An adjacency matrix can implement `areAdjacent(u, v)` in  $O(1)$  time and `incidentEdges(u, v)` in  $O(n)$  time.

13. Consider the following sorting algorithm.

**Algorithm** Sort( $A, n$ )

**Input:** Array  $A$  containing  $n$  different integer values.

**Out:** Array  $A$  sorted in increasing order of value.

```
for  $i \leftarrow 1$  to  $n - 1$  do {  
     $t \leftarrow A[i]$   
     $j \leftarrow i - 1$   
    (*)  
     $A[j + 1] \leftarrow t$   
}  
return  $A$ 
```

Which of the following instructions must be inserted at the point marked (\*) so that the algorithm correctly sorts the values stored in  $A$  in increasing order of value?

- (A) **while** ( $j \geq 0$ ) **and** ( $A[j] < t$ ) **do**  
    { $A[j] \leftarrow A[j + 1]; j \leftarrow j - 1$  }
- ✓(B) **while** ( $j \geq 0$ ) **and** ( $A[j] > t$ ) **do**  
    { $A[j + 1] \leftarrow A[j]; j \leftarrow j - 1$  }
- (C) **while** ( $j \geq 0$ ) **and** ( $A[j] > t$ ) **do**  
    **if**  $A[j] < t$  **then**  $t \leftarrow A[j]; j \leftarrow j - 1$  }
- (D) **for**  $j \leftarrow 0$  to  $i - 1$  **do**  
    **if**  $A[j] > A[j + 1]$  **then**  $A[j + 1] \leftarrow A[j]$
- (E) **for**  $j \leftarrow 0$  to  $n - 1$  **do**  
    **if**  $A[j] < t$  **then**  $A[j + 1] \leftarrow A[j]$
14. A *min-max* priority queue is an ADT which allows, both, the efficient removal of the largest and the smallest elements from a set. Which one of the following data structures should be used to implement a min-max priority queue to minimize the worst-case time complexity of the two above operations?
- ✓(A) A (2,4)-tree
- (B) A binary search tree
- (C) A hash table with separate chaining
- (D) A hash table with double hashing
- (E) An unordered array

## Part 2: Written Answers

Write your answers **directly on these sheets.**

15. (7 marks) Solve the following recurrence equation. Show how you solved the equation.

$$T(0) = 0$$

$$T(n) = 2n + T(n - 1), \text{ for } n \geq 1.$$

$$T(n - 1) = 2(n - 1) + T(n - 2)$$

$$T(n - 2) = 2(n - 2) + T(n - 3)$$

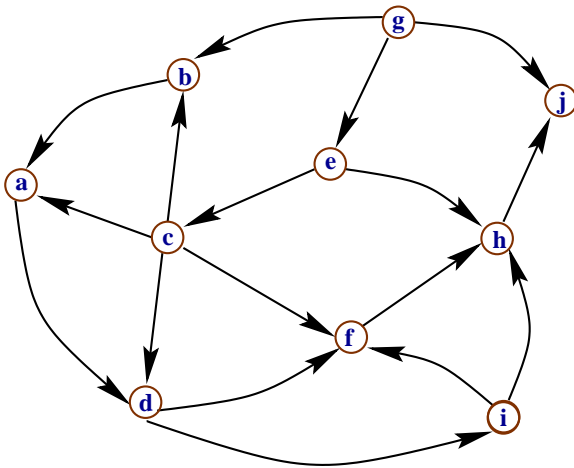
⋮

$$T(1) = 2 \times 1 + T(0) = 2$$

Hence,

$$T(n) = 2n + 2(n - 1) + 2(n - 2) + \cdots + 2(1) + 0 = 2 \sum_{i=1}^n i = 2 \frac{n(n+1)}{2} = n^2 + n$$

16. (5 marks) Compute a topological ordering for the following digraph (i.e. list the vertices in an order consistent with a topological ordering).



g, e, c, b, a, d, i, f, h, j.

For the following 2 questions you can use any of the algorithms studied in class. You do not have to write full descriptions of the algorithms. Just indicate which algorithm you would use, and which changes you need to make to the algorithm to answer each question. Indicate also how to pre-process the input for the algorithm. For example, given a road map with distances between cities, to find the shortest way of driving from city A to city B, your answer might be: build a graph in which every node is a city and an edge represents a road. The length of an edge is the length of the corresponding road. Use Dijkstra's algorithm to find the shortest path from A to B. This is the shortest route that should be taken.

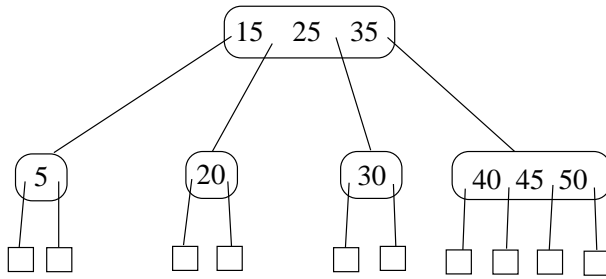
17. (5 marks) Let  $G = (V, E)$  be a graph representing a communication network in which the edges are labelled either *slow* or *fast*. Describe an algorithm for finding a path from vertex  $s$  to vertex  $t$  with the smallest number of *slow* edges (it does not matter how many *fast* edges there are in the path as long as the number of *slow* edges is as small as possible).

Assign length 0 to the fast edges and length 1 to the slow edges. Then use Dijkstra's algorithm to compute a shortest path from  $s$  to  $t$ . This path will have the smallest number of slow edges.

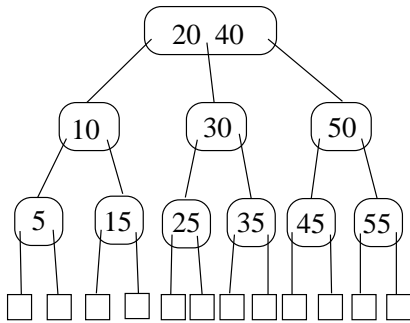
18. (5 marks) Consider a graph  $G = (V, E)$  representing a communication network in which the cost of leasing a link  $(u, v)$  is  $c(u, v)$ . We want to select the minimum cost set of edges that allows a given vertex  $s$  to send information to all the other vertices in  $G$ . Describe an efficient algorithm for solving this problem.

For every edge  $(u, v)$ , use  $c(u, v)$  as the length of the edge and then use the algorithm of Prim and Jarnik to compute a minimum spanning tree. This is the minimum cost set of edges that connects all the vertices.

19. (5 marks) Insert the key 36 in the following (2,4)-tree. You **must** use the algorithm discussed in class for inserting information in a (2,4)-tree. Show **all** intermediate trees.



20. (5 marks) Delete the value 15 from the following (2,4)-tree. You **must** use the algorithm discussed in class for removing information from a (2,4)-tree. Show **all** intermediate trees.



21. (12 marks) An array  $A$  storing  $n$  different positive integer values is *increasing-decreasing* if there is an index  $t$ ,  $0 < t < n - 1$ , such that the items  $A[0], \dots, A[t]$  are sorted in increasing order of value and the items  $A[t], \dots, A[n - 1]$  are sorted in decreasing order of value.

For example, consider the following arrays. Array  $B$  is increasing-decreasing as  $B[0] < B[1] < B[2]$  and  $B[2] > B[3] > B[4] > B[5]$ . Arrays  $A$  and  $C$  are not increasing-decreasing.

$A$	2	3	4	$B$	2	6	8	7	3	1	$C$	2	6	1	3	8
	0	1	2		0	1	2	3	4	5		0	1	2	3	4

Write an algorithm  $\text{inc-dec}(A, n)$  that receives as input an array  $A$  of size  $n$  and it returns the value **true** if  $A$  is increasing-decreasing and the value **false** otherwise.

**Algorithm**  $\text{inc-dec}(A, n)$

**In:** an array  $A$  storing  $n$  different positive integers

**Out:** true if  $A$  is increasing-decreasing; false otherwise.

$j \leftarrow 0$

$inc \leftarrow \text{true}$

**while**  $(j \leq n - 2)$  **and**  $(inc = \text{true})$  **do**{

**if**  $A[j] < A[j + 1]$  **then**  $j \leftarrow j + 1$

**else**  $inc \leftarrow \text{false}$

}

**if**  $j = n - 1$  **then return** false

**else** {

**while**  $(j \leq n - 2)$  **and**  $(inc = \text{false})$  **do** {

**if**  $A[j] > A[j + 1]$  **then**  $j \leftarrow j + 1$

**else**  $inc \leftarrow \text{true}$

}

**if**  $j = n - 1$  **then return** true

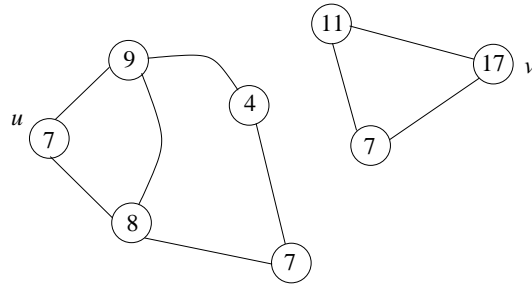
**else return** false

}

22. (2 marks) What is the worst case for the above algorithm?

(3 marks) Compute the time complexity of your algorithm for the previous question in the worst case. Explain how you computed the time complexity and give the order of the complexity.

23. (12 marks) Let  $G = (V, E)$  be a graph in which every node  $u$  stores an integer key  $u.\text{key}$ . We define the *composed weight* of node  $u$  as the sum of the keys of all nodes that can be reached from  $u$  and have key values **larger** than  $u.\text{key}$ . For example, for the following graph, the composed weight of node  $u$  is 17 and the composed weight of node  $v$  is 0. Write an algorithm `composed-weight( $G, u, \text{key}$ )`, that given an undirected graph  $G = (V, E)$ , a node  $u \in V$ , and its key, it returns the composed weight of  $u$ .



In the first call,  $k = u.\text{key}$ .

**Algorithm** `composed-weight( $G, u, k$ )`

**In:** undirected graph  $G = (V, E)$ , a node  $u \in V$ , and a key  $k$

**Out:** the composed weight of  $u$

Mark  $u$

**if**  $u.\text{key} > k$  **then**  $c \leftarrow u.\text{key}$

**else**  $c \leftarrow 0$

**For** each edge  $(u, v)$  incident on  $u$  **do**

**if**  $v$  is un-marked **then**  $c \leftarrow c + \text{composed-weight}(G, v, k)$

**return**  $c$

24. (4 marks) Compute the time complexity of your algorithm for the previous question in the worst case. Explain how you computed the time complexity and give the order of the complexity.

The University of Western Ontario  
Department of Computer Science

CS2210A Final Examination  
Wednesday, December 15, 2010  
14 pages, 24 questions  
3 hours

Name: \_\_\_\_\_

Student Number: \_\_\_\_\_

PART I	
PART II	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
Total	

**Instructions**

- Write your name and student number on the space provided.
- Please check that your exam is complete. It should have 14 pages and 24 questions.
- The examination has a total of 100 marks.
- When you are done, call one of the TA's and they will pick your exam up.