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 below, $\log(x)$ means $\log_2(x)$. You might find this fact useful:

$\log(xy) = \log x + \log y$.

1. Consider the following algorithm:

   Algorithm `traverse`(G, u)
   Input: Undirected, **connected** graph G, and vertex u of G.

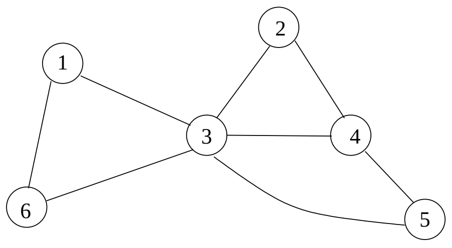
   ```
   Mark($u$)
   $c \leftarrow 0$
   For each vertex $v$ of $G$ do {
       For each vertex $w$ of $G$ do
           if $v \neq w$ then $c \leftarrow c + 1$
   }
   For each edge $(u, v)$ incident on $u$ do
       if $v$ is not marked then $c \leftarrow c$+traverse($G, v$)
   return $c$
   ```

   Assume that G is stored in an **adjacency matrix**. The graph has n vertices and m edges. What is the time complexity of the algorithm?

   (A) $O(n + m)$
   (B) $O(n^2 + m)$
   (C) $O(n^2)$
   (D) $O(n^2 + \text{degree}(u))$
   [✓ (E) $O(n^3)$]

2. Consider the following graph. Which one of the following **is NOT** a valid ordering of the vertices if the graph is traversed using breadth first search (BFS)?

   (A) 1, 6, 3, 5, 4, 2
   [✓ (B) 4, 3, 2, 1, 5, 6]
   (C) 5, 4, 3, 2, 1, 6
   (D) 3, 6, 4, 1, 2, 5
   (E) 2, 3, 4, 6, 5, 1

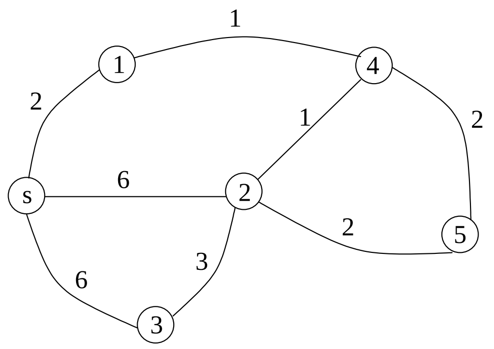


3. How many different AVL trees containing the four keys 1, 2, 3, 4 can be built?

   (A) 2
   (B) 3
   [✓ (C) 4]
   (D) 5
   (E) 6

4. Consider the following weighted 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, 4, 3, 2, 5
(C) s, 1, 4, 5, 2, 3
(D) s, 1, 4, 2, 3, 5
$\surd$(E) s, 1, 4, 2, 5, 3

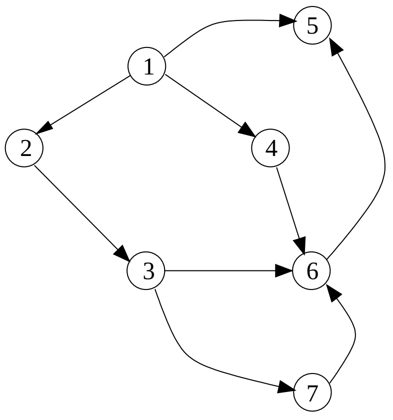


5. Consider the following pairs of functions $f(n), g(n)$. For which pair the functions are such that $f(n)$ is $O(g(n))$ and $g(n)$ is not $O(f(n))$?

(A) $f(n) = n^2, g(n) = n \log n$
(B) $f(n) = \log n, g(n) = 3 \log(n^2)$
(C) $f(n) = n, g(n) = 1000n + 5$
$\surd$(D) $f(n) = 17, g(n) = n$
(E) $f(n) = n, g(n) = \log(n^4)$

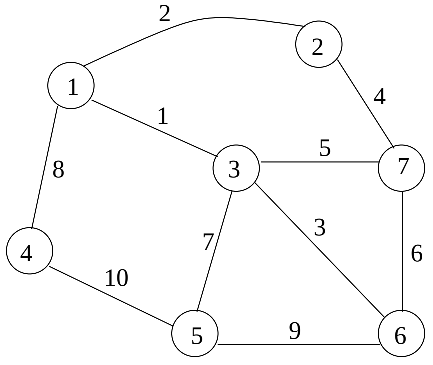
6. Which of the following statements is true **for all** depth first search traversals of the following graph starting at vertex 1?

$\surd$(A) Vertex 3 cannot be visited immediately before vertex 5.
(B) Vertex 3 must be visited before vertex 6.
(C) Vertex 2 must be the second vertex visited.
(D) Vertex 6 cannot be visited before vertex 5.
(E) Vertex 7 cannot be the last vertex visited.



7. Consider the following graph. Which edges, and in which order, are selected by Prim's algorithm if it stars at vertex 1?

$\surd$(A) (1,3), (1,2), (3,6), (2,7), (3,5), (1,4)
(B) (1,3), (1,2), (3,6), (2,7), (3,7), (7,6)
(C) (1,3), (3,6), (1,2), (2,7), (3,5), (1,4)
(D) (1,2), (1,3), (3,6), (2,7), (3,7), (7,6)
(E) (1,3), (1,2), (2,7), (3,7), (7,6), (3,5)



8. The following algorithm receives as input an array A storing n different integer values and it sorts A in increasing order of value.

```
Algorithm sort($A, n$)
Input: Array $A$ storing $n$ different integer values.
  for $i \leftarrow 1$ to $n-1$ do {
      $t \leftarrow A[i]$
      $j \leftarrow i-1$
      while $(j \geq 0)$ and $(A[j] > t)$ do {
          $A[j+1] \leftarrow A[j]$
          $j \leftarrow j-1$
      }
      $A[j+1] \leftarrow t$
  }
```

What is the complexity of this algorithm in the worst case and in the best case?

(A) Worst case: $O(n)$ and best case $O(n)$
✓(B) Worst case: $O(n^2)$ and best case $O(n)$
(C) Worst case: $O(ni)$ and best case $O(nj)$
(D) Worst case: $O(n^2)$ and best case $O(n^2)$
(E) Worst case: $O(n^2)$ and best case $O(ni)$

9. 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 edge
              if $B(G, v) = true$ then return true
          }
          else {
              Label $(u, v)$ as back edge
              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 G has at least 2 edges and it returns *false* otherwise.

10. 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 data structure should be used to implement a min-max priority queue to minimize the worst-case time complexity of the two above operations?

    (A) A binary search tree
    (B) A hash table with separate chaining
    (C) A hash table with double hashing
    ✓(D) An AVL-tree
    (E) An unordered array

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

    10, 3, 13, 17, 11.

    In which entry of the table is the key 11 stored?

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

12. 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 ← T(left child of r)
        q ← T(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.

13. 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 traverse($G, u$)
Input: Undirected, connected graph $G$, and vertex $u$ of $G$.

Mark $u$
For each edge $(u, v)$ incident on $u$ do {
    trim($u, v$)
    if $v$ is not marked then traverse($G, v$)
}
```

Algorithm `trim`(u, v) performs kmn operations, where k is a constant value. Assume that G is stored in an adjacency list. What is the time complexity of algorithm `traverse`(G, u) in the worst case?

(A) $O(m + n)$
(B) $O(mn)$
(C) $O(mn^2)$
[✓ (D) $O(m^2n)$]
(E) $O(n^2)$

Part 2: Written Answers

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

14. (4.5 marks) Solve the following recurrence equation. Show all intermediate steps.

$T(1) = 1$

$T(n) = 1 + n + T(n-1)$, for $n > 1$

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

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

$T(n-2) = 1 + (n-2) + T(n-3)$

$\vdots$

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

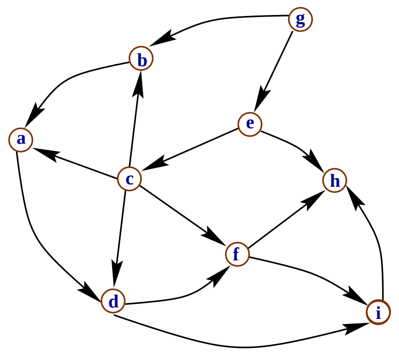
Hence,

$$T(n) = (1+n) + (1+n-1) + (1+n-2) + \cdots + (1+2) + 1$$
$$= (n+1) + n + (n-1) + (n-2) + \cdots + 3 + 1$$
$$= \left(\sum_{i=3}^{n+1} i\right) + 1 = \left(\sum_{i=1}^{n+1} i\right) - 2 = \frac{(n+1)(n+2)}{2} - 2 = \frac{n^2}{2} + \frac{3n}{2} - 1$$

15. (4 marks) Consider the following algorithm:

```
Algorithm topologicalOrdering(G)
Input: Directed graph G
Output: Topological ordering for the vertices of G
while G is not empty do {    if there is a vertex u with no incoming edges then {        Output
u
      Remove from G vertex u and all edges incident on it.
   }
   else return
}
```

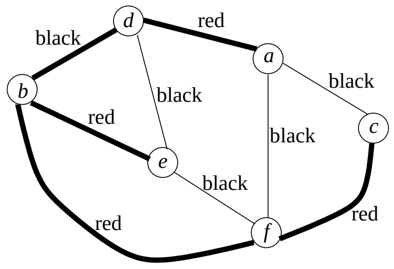
Compute a topological ordering for the following digraph.



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

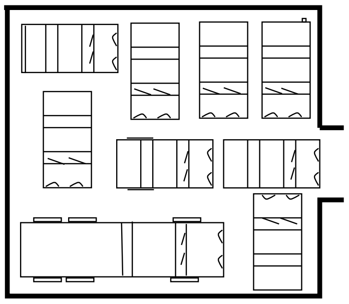
For the following 2 questions you can use any of the algorithms studied in class. You do not have to write full descriptions of the algorithms. Just indicate which algorithm you would use, and which changes you need to make to the algorithm to answer each question. Indicate also how to pre-process the 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.

16. (5 marks) Consider an undirected connected graph $G = (V, E)$. Every edge (u, v) of G is colored either red or black. We want to select a smallest subset S of edges that interconnects all the vertices of the graph (so the subgraph $G' = (V, S)$ is connected) and for which the total number of black edges is minimum. For example, for the following graph the solution is given by the set of edges in bold. Give an efficient algorithm for solving this problem.



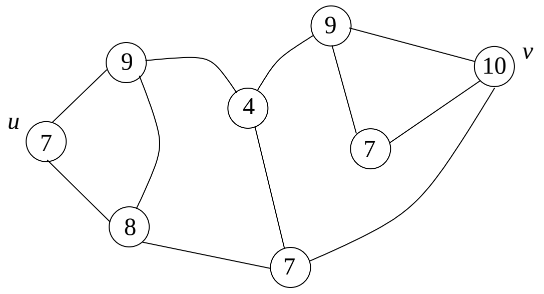
Assign to every red edge length 1 and assign to every black edge length n. Then, compute a minimum spanning tree T of the resulting graph by using Prim's algorithm. This tree T interconnects all vertices of G using the minimum possible number of black edges.

17. (5 marks) Consider a set of n cars parked in a small parking lot with a single exit. The problem is to find a way of moving all the cars out of the parking lot. You build a directed graph $G = (V, E)$ in which every vertex corresponds to a car and there is an edge from vertex u to vertex v if car u is blocking car v. Describe an algorithm that uses the information in G to determine the order in which the cars need to move out of the parking lot.



Since there is an edge from vertex u to vertex v if car u blocks car v, then a valid order for taking the cars out of the parking lot is given by a topological ordering of the vertices of G. Use the algorithm in question 15 for computing a topological order for V and take the cars out of the parking lot in this order.

18. (11 marks) Let $G = (V, E)$ be a connected graph in which every node u stores an integer key u.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.key. For example, for the following graph, the composed weight of node u is $9 + 8 + 9 + 10 = 36$ and the composed weight of node v is 0. Write an algorithm `CompWeight`(u, k), that given a node u of G and its key, it returns the composed weight of u. The initial invocation is `CompWeight`$(u, u.\text{key})$.



```
Algorithm CompWeight($u, k$)
Input: Vertex $u$ and value $k$
Output: Composed weight of $u$
Mark ($u$)
if $u$.key $> k$ then $c \leftarrow u$.key
else $c \leftarrow 0$
For each edge $(u, v)$ incident on $u$ do
   if $v$ is not marked then $c \leftarrow c$+CompWeight($v, k$)
return $c$
```

19. (4 marks) Compute the time complexity of your algorithm for the previous question in the worst case assuming that the graph is stored in an adjacency list. Explain how you computed the time complexity and give the order of the complexity.

Let us first ignore the recursive calls. Ignoring recursive calls the most expensive part of the algorithm is the **for** loop. Each iteration of the loop performs a constant number c' of operations and it is repeated degree(u) times as the graph is stored in an adjacency list, so the total number of operations performed by the **for** loop is $c' \times$ degree(u). Outside the loop an additional constant number c'' of operations are performed, so each invocation of the algorithm performs $c'' + c' \times$ degree(u) operations.

The recursive calls make the algorithm perform a depth first search traversal of the graph, so the algorithm makes one recursive call per node of the graph. Therefore, the total number of operations performed by the algorithm is

$$\sum_{u \in G} (c'' + c' \times \text{degree}(U)) = c''n + c'(2m) \text{ is } O(m+n).$$

20. (11 marks) Consider two arrays A and B, each storing n integer values. In both arrays the values are sorted in increasing order of value. Write an algorithm that merges A and B and stores their values in an array C in such a way that C is sorted in increasing order of value, **and there are no duplicated values in** C. For example, if A and B are as follows:

A

3	6	9	11	14
0	1	2	3	4

B

1	3	9	11	17
0	1	2	3	4

Then array C must be this:

1	3	6	9	11	14	17
0	1	2	3	4	5	6

Algorithm merge(A, B, C, n)

Input: Arrays A and B storing n integer values. Empty array C.

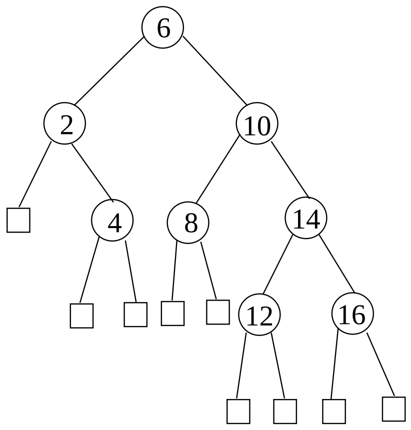
Out: Array C storing all non-repeated values in A and B sorted in increasing order.

```
$i_A \leftarrow 0$
$i_B \leftarrow 0$
$i_C \leftarrow 0$
While $(i_A \leq n-1)$ and $(i_B \leq n-1)$ do {
    if $A[i_A] < B[i_B]$ then {
        if $(i_C = 0)$ or $(A[i_A] \neq C[i_C])$ then{
            $C[i_C] \leftarrow A[i_A]$
            $i_A \leftarrow i_A + 1$
            $i_C \leftarrow i_C + 1$
        }
    }
    else if $(i_C = 0)$ or $(B[i_B] \neq C[i_C])$ then{
            $C[i_C] \leftarrow B[i_B]$
            $i_B \leftarrow i_B + 1$
            $i_C \leftarrow i_C + 1$
        }
}
if $i_A \leq n-1$ then {
    while $i_A \leq n-1$ do
        if $A[i_A] \neq C[i_C]$ then {
            $C[i_C] \leftarrow A[i_A]$
            $i_A \leftarrow i_A + 1$
            $i_C \leftarrow i_C + 1$
        }
}
else
    while $i_B \leq n-1$ do
        if $B[i_B] \neq C[i_C]$ then {
            $C[i_C] \leftarrow B[i_B]$
            $i_B \leftarrow i_B + 1$
            $i_C \leftarrow i_C + 1$
        }
```

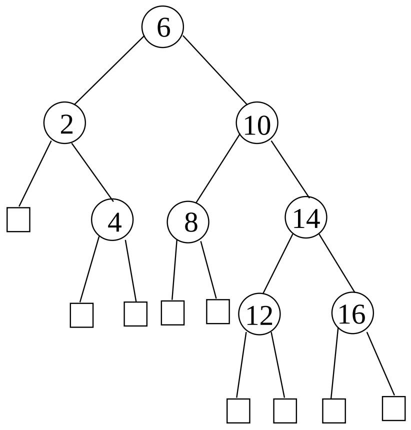
21. (4 marks) Compute the time complexity of your algorithm for the previous question in the worst case. Explain how you computed the time complexity.

    Note that each iteration of any of the 3 while loops performs a constant number c of operations. The algorithm performs the first while loop plus either the second or third while loops. In any case, the total number of iterations that the combined while loops perform is $2n$ since in each iteration of any of the while loops the value of one of the variables i_A or i_B is increased by 1, and the combined loops end when both, $i_A = n$ and $i_B = n$. Outside the first while loop a constant number c_1 of operations are performed. Therefore, the total number of operations performed by the algorithm is $c_1 + 2cn = O(n)$.

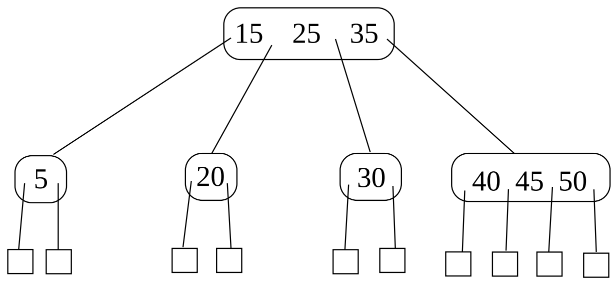
22. (5 marks) Consider the following AVL tree. Insert the key 11 into this tree and re-balance if needed. You **must** use the algorithms discussed in class, and if re-balancing is needed you must indicate the kind of rotation performed. Show all intermediate trees.



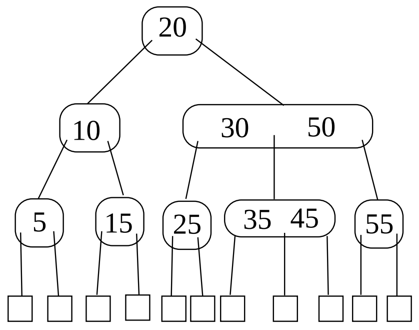
23. (5 marks) Consider the following AVL tree (it is the same as in the previous question). Remove the key 4 and re-balance the tree if needed. You **must** use the algorithms discussed in class, and if re-balancing is needed you must indicate the kind of rotation performed. Show **all** intermediate trees.



24. (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 and give the name of the operation performed in each intermediate tree.



25. (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 and give the name of the operation performed on each intermediate tree.



The University of Western Ontario
Department of Computer Science

PART I	
PART II	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
Total	

CS2210A Final Examination
15 pages, 25 questions
3 hours

Name: ____________________

Student Number: ____________________

Instructions

- Write your name and student number on the space provided.
- Please check that your exam is complete. It should have 15 pages and 25 questions.
- The examination has a total of 100.5 marks.
- When you are done, call one of the TA's and he will pick your exam up.