Final Exam

Graded

Student

Adam Fenjiro

Total Points

57 / 100 pts

Question 1

T/F and MC **10** / 10 pts

1.1 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect

1.2 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect

1.3 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect

1.4 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect

1.5 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect

1.6 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect

1.7 **(no title)** **2** / 2 pts

✔ **– 0 pts** Correct

– 0.5 pts Missing one answer

– 1 pt Missing two answers

– 1.5 pts Missing three answers

1.8 **(no title)** **2** / 2 pts

✔ **– 0 pts** Correct

– 0.6 pts Mostly Correct

– 1.3 pts Mostly incorrect

– 2 pts Incorrect

Question 2

Graph and getEdge **1** / 10 pts

2.1 **A** **0** / 2.5 pts

– **0 pts** Correct

✔ – **1.5 pts** Did not add edge k correctly

✔ – **1 pt** Time complexity incorrect

– **0.5 pts** Small typo

2.2 **B** **0** / 2.5 pts

– **0 pts** Correct

✔ – **0.5 pts** 1st position for k is wrong

✔ – **0.5 pts** 2nd position for k is wrong

– **1 pt** Time complexity incomplete

✔ – **1.5 pts** Time complexity wrong

2.3 **C** **0** / 2.5 pts

– **0 pts** Correct

✔ – **0.75 pts** 1st position for k incorrect

✔ – **0.75 pts** 2nd position for k incorrect

✔ – **1 pt** Wrong time complexity

– **0.5 pts** Small typo

2.4 **D** **1** / 2.5 pts

– **0 pts** Correct

✔ – **0.75 pts** 1st k in wrong position

✔ – **0.75 pts** 2nd k in wrong position

– **1 pt** Incorrect time complexity

Question 3

(no title) **13** / 15 pts

3.1 **f0** **1** / 1 pt

✔ **– 0 pts** Correct

3.2 **f1** **2** / 2 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect

3.3 **f2** **2** / 2 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect

3.4 **f3** **2** / 2 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect, should be C

3.5 **f4** **2** / 2 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect, should be B

3.6 **f5** **2** / 2 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect should be D

3.7 **f6** **0** / 2 pts

– 0 pts Correct

✔ **– 2 pts** Incorrect, should be B

3.8 **f7** **2** / 2 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect, should be F

Question 4

AVL **4.5** / 10 pts

4.1 **(no title)** **3** / 3 pts

- ✔ **– 0 pts** Correct
- **– 1.5 pts** Partial Credit, leaf nodes may be off
- **– 3 pts** Incorrect, should be [30 20 40 10 25 empty 50] as a tree

4.2 **(no title)** **1.5** / 3 pts

- **– 0 pts** Correct
- ✔ **– 1.5 pts** Wrong Root or 1 node swapped
- **– 3 pts** Incorrect

4.3 **(no title)** **0** / 4 pts

- **– 0 pts** Correct
- **– 2 pts** Half correct, should be 2 tri node reconstructions and 14 in the root node
- ✔ **– 4 pts** Incorrect, should be 2 and 14

Question 5

24 Tree **2** / 6 pts

5.1 **1** / 3 pts

- **– 0 pts** Correct
- **– 1 pt** Partial Credit
- ✔ **– 2 pts** Partial Credit
- **– 3 pts** Incorrect

5.2 **1** / 3 pts

- **– 0 pts** Correct
- **– 1 pt** Partial credit
- ✔ **– 2 pts** Partial credit
- **– 3 pts** Incorrect

Question 6

Heap **6** / 6 pts

6.1 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 0.5 pts Assumes 1-indexed array

– 1 pt Incorrect

6.2 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 0.5 pts Assumes array index starts at 1

– 1 pt Incorrect

6.3 **(no title)** **1** / 1 pt

✔ **– 0 pts** Correct

– 0.5 pts Rounding/offset exists, but is slightly off

– 1 pt Incorrect

6.4 **(no title)** **3** / 3 pts

✔ **– 0 pts** Correct

– 1.5 pts Half Correct

– 3 pts Incorrect

Question 7

Huffman **2** / 4 pts

– 0 pts Correct

– 1 pt Partial Credit

✔ **– 2 pts** Partial Credit

– 3 pts Partial Credit

– 4 pts Incorrect

Question 8

Symbol table **2** / 4 pts

8.1 (no title) **0** / 1 pt

– **0 pts** Correct

✔ – **1 pt** Incorrect

8.2 (no title) **1** / 1 pt

✔ – **0 pts** Correct

– **1 pt** Incorrect

8.3 (no title) **0** / 1 pt

– **0 pts** Correct

✔ – **1 pt** Incorrect

8.4 (no title) **1** / 1 pt

✔ – **0 pts** Correct

– **1 pt** Incorrect

Question 9

Sorting Alg. **5** / 5 pts

✔ – **0 pts** Correct

– **2.5 pts** Did not provide a sufficient argument against other sorting algorithms, need to mention partially sorted data or is off about how insertion sort works

– **5 pts** Incorrect, should be A

Question 10

(no title) **2** / 10 pts

10.1 **(a)** **0** / 3 pts

– **0 pts** Correct

– **1 pt** Partial Credit

– **2 pts** Partial Credit

✔ – **3 pts** Incorrect

10.2 **(b)** **2** / 3 pts

– **0 pts** Correct

✔ – **1 pt** Partial Credit

– **2 pts** Partial Credit

– **3 pts** Incorrect

10.3 **(c)** **0** / 2 pts

– **0 pts** Correct

✔ – **2 pts** Incorrect

10.4 **(d)** **0** / 2 pts

– **0 pts** Correct

✔ – **2 pts** Incorrect

Question 11

Shorted Path **6.5** / 10 pts

11.1 **(no title)** **5.5** / 7 pts

- **– 0 pts** Correct
- **– 0.5 pts** Partial Credit
- **– 1 pt** Partial Credit
- ✔ **– 1.5 pts** Partial Credit
- **– 2 pts** Partial Credit
- **– 3 pts** Partial Credit
- **– 3.5 pts** Partial Credit
- **– 4 pts** Partial Credit
- **– 5 pts** Partial Credit
- **– 6 pts** Partial Credit
- **– 7 pts** Incorrect

11.2 **(no title)** **1** / 3 pts

- **– 0 pts** Correct
- **– 1 pt** 1 error
- ✔ **– 2 pts** 2 errors
- **– 3 pts** Incorrect

Question 12

Transitive Closure **2** / 5 pts

12.1 **List** **2** / 2 pts

- ✔ **– 0 pts** Correct
- **– 1 pt** listed at least one group, but not both
- **– 2 pts** Incorrect, should have been (a,b,c), (e,f), and optionally (d)

12.2 **total cost** **0** / 3 pts

- **– 0 pts** Correct
- **– 1 pt** Partial Credit
- **– 2 pts** Partial Credit
- ✔ **– 3 pts** Incorrect

Question 13

Code

1 / 5 pts

- **– 0 pts** Correct
- **– 5 pts** blank or major flaws
- **– 1 pt** no correct enqueue left/right
- **– 1 pt** no enqueue root
- **– 1 pt** no correct dequeue()
- **– 1 pt** no checking if left/right is null
- **– 0.5 pts** other minor mistake
- **– 1 pt** no correct loop
- **– 1 pt** Uses incorrect data structure
- ✔ **– 4 pts** some effort, major flaw did not print level by level
- **– 1.5 pts** print at the wrong place

Data Structures, Spring 2023

Final Exam (100points)

Please write your name and user name clearly. Account name/User name is the part before @ in your email address. It is not your M number. For example, my email is ruihong@mtu.edu and my username is ***ruihong***

Your Name: Adam FENJIRO

Account Name/User Name: afenjiro

Note:

1. Please write your answers clearly.
2. Please use scratch paper for your intermediate work.
3. You will be given some scratch paper. If you need more scratch paper, please ask for it.
4. Please hand in your exam and scratch paper as well at the end of the exam.

1. **(10) True/False and multiple choices questions**

1.1. The collision happens when two entries in the map have the same key. True or (false)?
False

1.2. Quick Sort and Merge Sort both use divide and conquer technique. (True) or false?
True

1.3. Huffman Coding, Dijkstra's Algorithm and Fractional Knapsack are all greedy algorithms. (True) or false? True

1.4. The main purpose of B tree is to reduce the number of I/O operations. (True) or false?
True

1.5. Binary search algorithm can be performed on any ordered linear data structure, such as ordered array and ordered doubly linked list. True or (false)? False

1.6. Priority Queue<Key, Value> data structure requires the key is unique. True or (false)? False

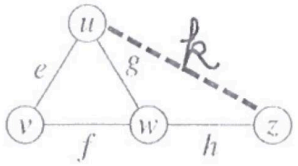
1.7. (2p) Which statements are true about Dijkstra's algorithm? Choose all correct ones.
A, B, C, D

(A.) Dijkstra is the computer scientist who developed this algorithm.
(B.) This algorithm found the shortest path from a given source to all vertices.
(C.) This algorithm does not work on the graph with negative weighted edges
(D.) Dijkstra's algorithm runs in O(m * log n) time on an adjacency list/map structure.

1.8. (2p) Which trees have the height of O(log n) ? Choose all correct ones.
A, B, C

(A.) Heap
(B.) AVL
(C.) (2-4) tree
D. Binary search tree

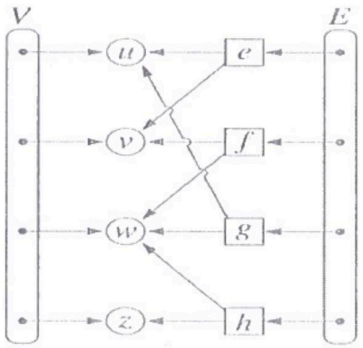
2. (10 points) Questions about different undirected graph implementations and cost of getedge(u,v). Let's add an edge k from vertex u to vertex z to the graph below. See the dashed bold line.



Please make relevant data changes to the different implementation of graph. You don't need to redraw the illustration, just make the changes on top of the given data structure illustration. Please follow the notation as illustrated for the graph before the change.

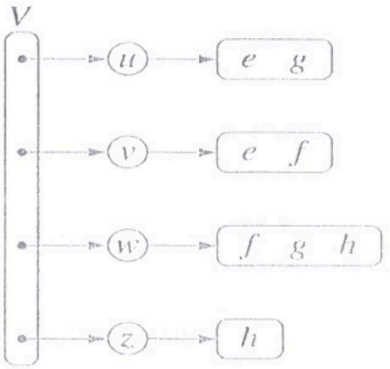
Please also answer the question about the time complexity of method getEdge(u,v) for each implementation in terms of **n, m, deg(u),** where n is number of vertices, m is the number of edges, deg(u) is the number of adjacency edges of vertex u.

A. Edge List:



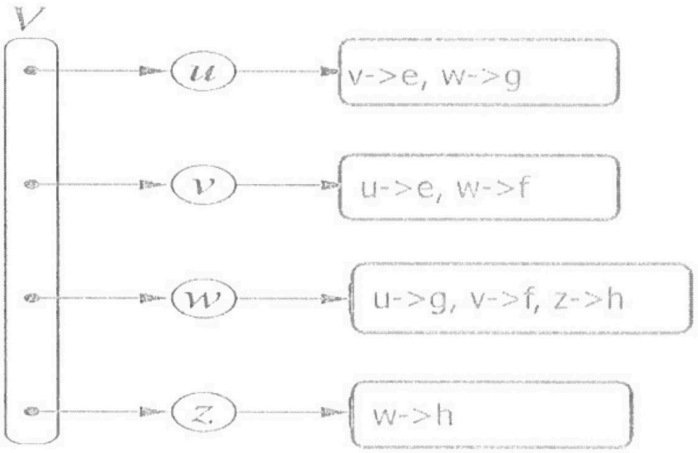
What is the time complexity of getEdge(u,v)? O(1)

B. Adjacency List Structure



What is the time complexity of getEdge(u,v)? O(n)

C. Adjacency Map Structure



What is the expected time complexity of getEdge(u,v)? $O(deg(u), deg(v))$

D. Adjacency Matrix

		0	1	2	3
u	0		e	g	
v	1	e		f	
w	2	g	f		h
z	3			h	

What is the time complexity of getEdge(u,v)? $O(1)$

3. (15 points) For each function (f0 – f7) below, give the best matching order of growth of the running time on the right. Write your choice on the top of dashed lines.

A

```
Alg f0(n)
   return 1
```

C

```
Alg f1(n)
   x ← 0
   for i from 1 to n-1
      x ← x+1
   return x
```

E

```
Alg f2(n)
   x ← 0;
   for i from 1 to n-1
      for j from i+1 to n-1
         x ← x+1
   return x
```

C

```
Alg  f3(n)
    if n=0 then return 0
    x ← n + f3(n-1)
    return x
```

B

```
Alg int f4(int n)
   if n=0 then return 0
   x ←1 + f4(n/2)
   return x
```

D

```
This algorithm calls f4()

Alg  f5(n)
   x ← 0
   for i from 1 to n-1
      x ← x + f4(n-1)
   return x
```

D

```
Alg  f6(n)
   x ← 0
   i ← n
   while i>0 do
      x ← x + i
      i ← i/2
   return x
```

F

```
Alg  f7(n)
  if n=0 or n=1 then return 1
  return f7(n-1) + f7(n-2)
```

A: O(1)

B: O(log n)

C: O(n)

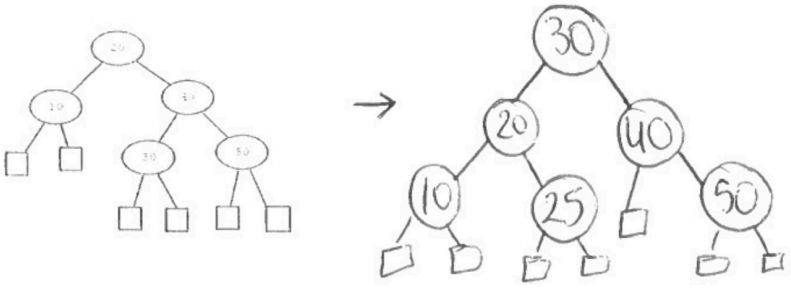
D: O(n log n)

E: $O(n^2)$

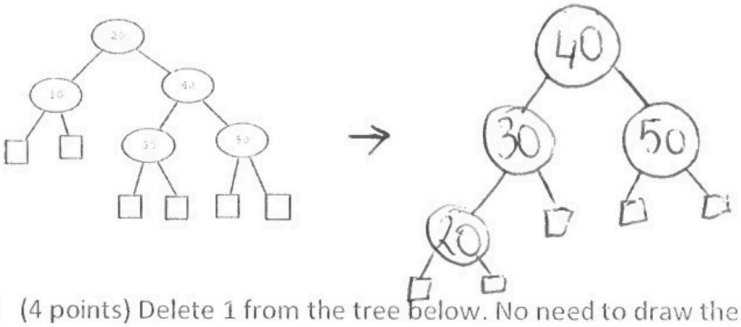
F: $O(2^n)$

4. (10 points) AVL

4.1 (3 points) Insert 25 to the AVL tree below and draw the new AVL tree.



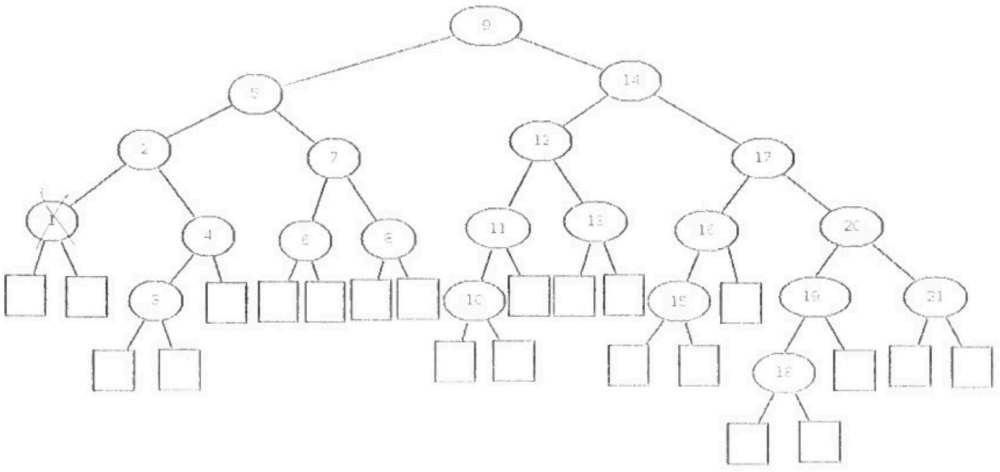
4.2 (3) Delete 10 from the tree below and draw the final new AVL tree below. If a successor needs to be selected, select the next largest, not the next smallest key.



4.3 (4 points) Delete 1 from the tree below. No need to draw the tree. Just answer the two questions.

How many tri-node reconstruction(s) is/are needed to balance this AVL tree? 0.

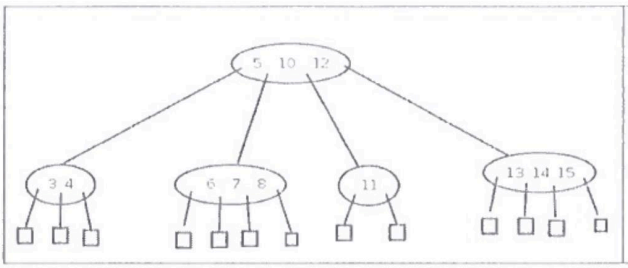
What is the data in the root node after all the reconstruction(s)? 9



Spring 2023 CS2321 Final Exam

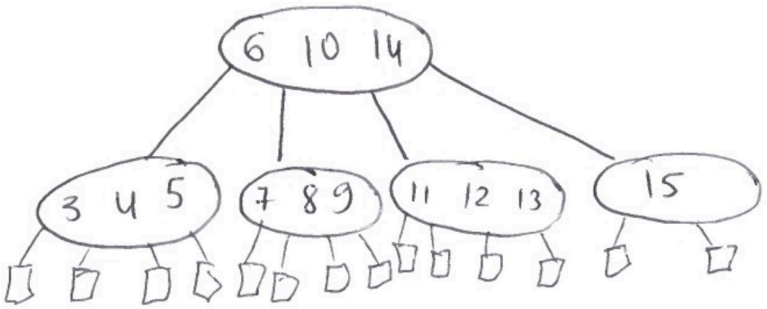
5. (4 points) (2-4) trees.

5.1 (3 points) Insert 9 to the (2,4) below and draw the new (2,4) tree.

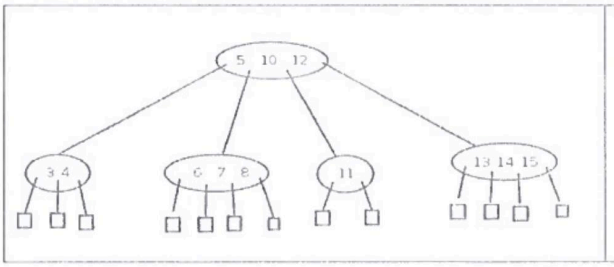


When a node is overflowed with 4 keys: (K1, K2, K3, K4):

- Split the data into two nodes.
- The existing node will hold (k1, K2)
- The new code will hold (k4).
- K3 should be added to parent node.

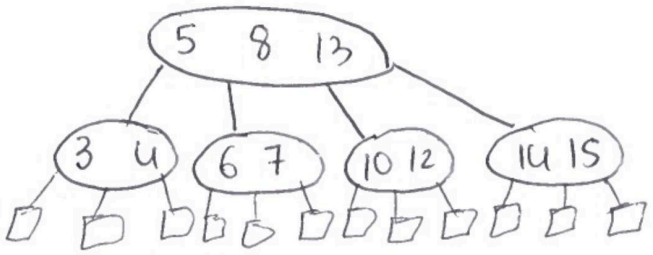


5.2 (3 points) remove 11 from the (2,4) below and draw the new (2,4) tree. If a successor needs to be selected, select the next largest, not the next smallest key.



If an underflow situation happens, do the following actions in this order:

- If left sibling is 2-node, merge.
- If right sibling is 2-node, merge
- If left sibling is 3 or 4-node, transfer
- If right sibling is 3 or 4-node, transfer



6. (6 points) Heap

6.1. (3 points) Use array to implement the heap data structures. Given the index of a node in the heap is *i*, answer the following:

The index of this node's left child is: $2i + 1$

The index of this node's right child is: $2i + 2$

The index of this nodes' parent is: $\frac{i-1}{2}$

6.2. (3 points) Insert the following keys 5, 8, 9, 3, 2, 1 in order into an empty **minimum** heap one by one. Write the first 6 elements in the array after the inserting.

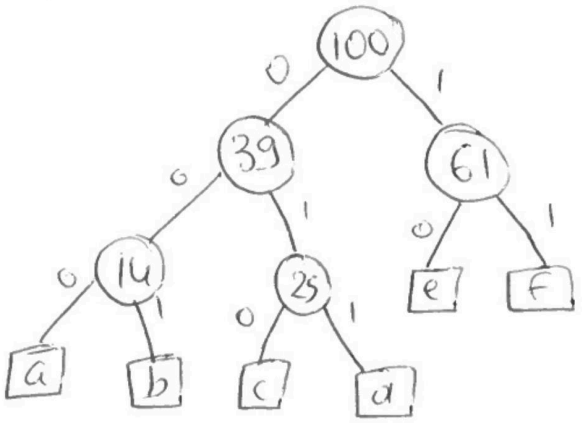
1	3	2	8	5	9

7. **(4 points) Data compression**

A message contains 6 unique symbols (a,b,c,d,e,f) and has total of 100 chars. The frequency of each symbol is given in the table below.

a	b	c	d	e	f
5	9	12	13	16	45

We will use Huffman code to compress this message. Draw the Huffman coding tree. The answer is not unique. Just draw one of them.



a = 000
b = 001
c = 010
d = 011
e = 10
f = 11

8. (4 points) In Computer Science, a symbol table is a data structure used by compiler in order to keep track of semantics of variables, such as type, scope, storage address. Numerous data structures are available for implementing the symbol tables.

   For each of the operations on the left, list which symbol table implementation(s) is/are most efficient in terms of expected upper bound time complexity for the worst case.

C Search for the information of a symbol	A. Unordered Linked list
A Adding a symbol when a new variable is declared	B. Ordered Array
B Delete a symbol when the scope of the variable is closed	C. Binary Search Tree
D Overall, which implementation is best for symbol table?	D. Separate-chaining hash table

9. (5 points) You were told to fix the order of a given array of size n. The data in the array is almost sorted with the exception of a smaller number of items (say k) are out of order. k is much smaller than n.

   Example: 3, 4, **10**, 6, 7, 8, 9, 11, 12, 13, 14, **18**, 16, 17, 35 …
   All numbers except 10 and 18 are in the wrong spots.

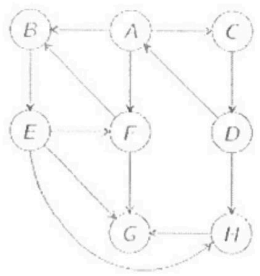
   Which sorting algorithm is the best one to be used to sort the data again? A

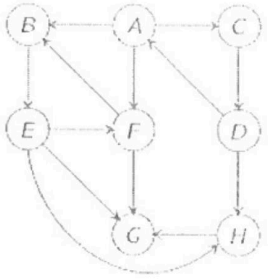
   (A.) Insertion sort
   B. Selection sort
   C. Merge sort

   Why did you choose that particular soring algorithm over the other 2 algorithms? Please give specific reason. Don't say something like "it is better, it is efficient".

Insertion sort is the best sorting algorithm in this case because it is much more efficient with small input size data as well as partially sorted arrays, which is exactly the case here.

10. (10 points) Consider the following directed graph. Two copies of exact same graph.





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

DFS from A

(a) Run depth-first search, starting at vertex A. Assume the adjacency lists are in alphabetical order, e.g., when exploring vertex A, consider A->B first, then A->C, then A->F. Complete the list of vertices in the order they are first discovered by DFS. The first two have been given.

A B E F G ___ ___ ___

BFS

(b) Run breadth-first search, starting at vertex A. Assume the adjacency lists are in alphabetical order, e.g., when exploring vertex A, consider A->B first, then A->C, then A->F. Complete the list of vertices in the order in which they are discovered by BFS. The first two have been given.

A B C F E D H G

(c) Which traversal algorithm finds the shortest path in terms of number of edges?

DFS

(d) What is the time complexity of DFS algorithm in terms of number of vertices n and number of edges m?. C

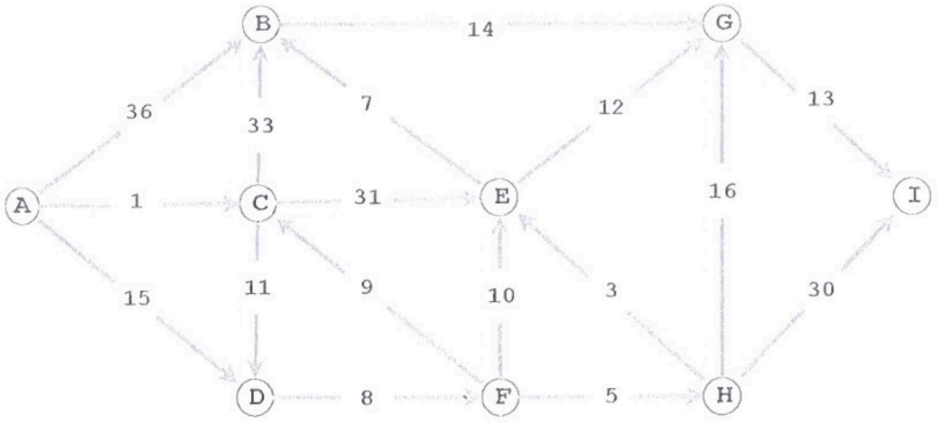
A. O(n*m)

B. O(n+m)

(C.) O((n+m)logn)

11. Shortest paths. (10 points)

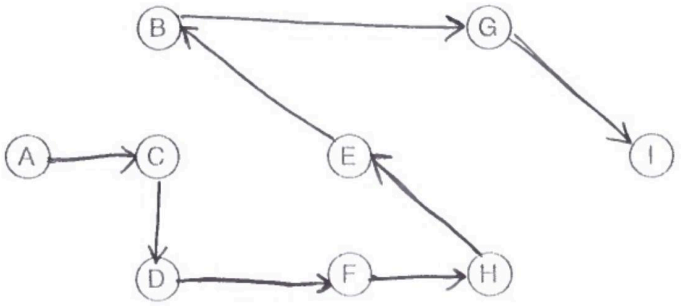
Run Dijkstra's algorithm on the weighted digraph below, starting at vertex A.



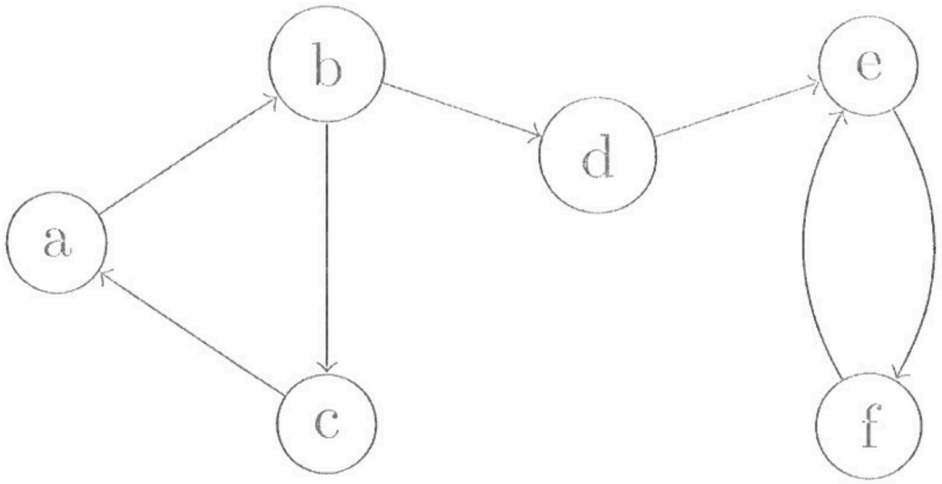
11.1 List the vertices in the order in which the vertices are dequeued from the priority queue and give the length of the shortest path from A.

vertex:	A	C	D	F	H	E	B	G	I
distance:	0	1	12	20	25	28	35	49	62

11.2 Illustrate all the edges in the graph below for all the shortest path from vertex A to all other vertices.



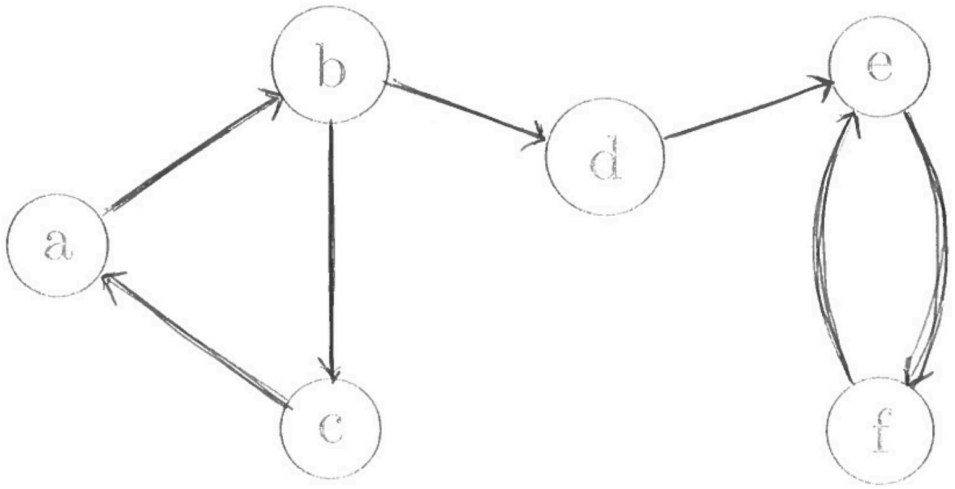
12. Connectivity and Transitive Closure (5 points). Consider the following directed graph:



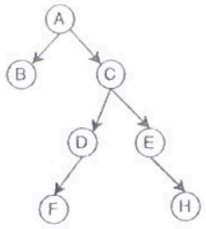
12.1 List the strongly connected components of this graph.

{a, b, c} and {e, f}

12.2 This is a copy of the above graph. Draw the edges of the transitive closure of this graph that would be calculated after running the Floyd-Warshall algorithm.



13. (5 points) Given a binary tree, write pseudo code to print the nodes from root to leaf level by level. For example,



We will print root node A first, then print A's children which are B and C, then print B and C's children which are D and E, keeping doing it until there is no more nodes. The output will be a list: A, B, C, D, E, F, H.

You may call any tree methods. Feel free to create helper functions and use temporary auxiliary data structures (such as stack, queue, PQ, heap, map, graph) that we studied this semester.

To print the data in a tree node v, simply call `print(v.getElement())`

```
Algorithm printTree( T )
    Input: binary tree T
    Output: a list of all nodes in the order of level by level.
```

```
// we are going to create a stack S
S = new stack;
// we will push the root into the stack
S.push(T.root);
// while the stack is not empty, we will recursively
 add the right and left child into the stack
 then pop: pre-order traversal.
while (!S.isEmpty()){
    temp = S.pop();
    if (temp.hasRightChild){ temp.right.push(); }
    if (temp.hasLeftChild){ temp.left.push(); }
    printTree(temp)
}
end.
```