# CSE 247/502N Exam 2

Byeongchan Gwak

TOTAL POINTS

**90 / 100**

QUESTION 1

## Operations and Algorithms (Multi Choice & T/F) 24 pts

1.1 Best Comparison Sort Expectation **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.2 HashTable put() **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.3 BST insert() **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.4 AVL insert() **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.5 Adj. Matrix Outgoing Edges **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.6 Unordered Linked List Set exists() **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.7 Ordered Array List Set exists() **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.8 DAG & 1 Topo Order **0 / 2**

**- 0 pts** Correct

✓ **- 2 pts Incorrect**

**- 0 pts** Ok with explanation

1.9 Dijkstra's and All Sources Shortest Path **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.10 Prim's and Kruskal's same result? **1 / 1**

✓ **- 0 pts Correct**

**- 1 pts** Incorrect

1.11 Bucket Sort Stable? **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

1.12 DFS on Binary Search Tree: Level order **1 / 1**

✓ **- 0 pts Correct**

**- 1 pts** Incorrect

1.13 Is AVL Tree a Binary Search Tree **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

QUESTION 2

## Sorting Concepts 15 pts

2.1 Justify Lower Bound **4 / 4**

✓ **- 0 pts Correct**

**- 2 pts** Doesn't clearly express need to look at **every** item to determine if they are in order.

**- 4 pts** Substantially incorrect or blank

**- 2 pts** Flaw in argument (ex: largely based on sorts we've seen rather than concepts needed for sorting)

2.2 Updating just two out of place items **2 / 2**

✓ **- 0 pts Correct**

**- 2 pts** Incorrect

### 2.3 Finding a student in ordered array collection 2 / 2

✓ **- 0 pts** **Correct**

**- 2 pts** Incorrect

**- 1 pts** Answer is unclear, but partial credit for additional comment

**- 2 pts** Not a reasonable assumption

### 2.4 When Radix Sort = Heap Sort 0 / 3

**- 0 pts** Correct

✓ **- 3 pts** **Incorrect**

**- 0 pts** Correct (but base of log was 2)

**- 1 pts** Minor error / off by 1

**- 1 pts** Minor error (work shown is ok; Algebraic error)

**- 1 pts** Not simplified / $$n$$ expressed in terms of $$n$$, but work show is ok

**- 1 pts** Should include/consider impact of $$d$$ here.

### 2.5 Faster at "equal complexity" point 0 / 2

**- 0 pts** Correct

✓ **- 2 pts** **Incorrect**

### 2.6 Faster larger than "equal complexity" point 0 / 2

**- 0 pts** Correct

✓ **- 2 pts** **Incorrect**

QUESTION 3

## Graph Represntations 10 pts

### 3.1 Adj Matrix Min Space 2 / 2

✓ **- 0 pts** **Correct**

**- 2 pts** Incorrect

### 3.2 Edge List incidentEdges(v) 8 / 8

✓ **- 0 pts** **Correct**

**- 0.1 pts** Index based for-loop would be $$O(\lvert E \rvert^2)$$, rather than $$O(\lvert E \rvert)$$, which is possible with for-each style loop.

**- 1 pts** Check of edge fails to check both vertices in the `Edge`

**- 1 pts** Fails to create / return a list

**- 1 pts** Error in method signature (missing return type or parameter/parameter type), etc.

**- 3 pts** Fails to use parameter `v` to check Edges (ex: `if` statement missing or substantially incorrect)

**- 3 pts** Fails to iterate over `edges`

**- 1 pts** Fails to add edge object to list

**- 8 pts** Substantially incorrect / missing

**- 1 pts** Misc. inefficiency (extra loops / iterating over `vertices` etc).

QUESTION 4

## Maps 10 pts

### 4.1 Version 1 2 / 2

✓ **- 0 pts** **Correct**

**- 2 pts** Incorrect

### 4.2 Version 2 2 / 2

✓ **- 0 pts** **Correct**

**- 2 pts** Incorrect

### 4.3 Version 3 2 / 2

✓ **- 0 pts** **Correct**

**- 2 pts** Incorrect

### 4.4 Version 4 2 / 2

✓ **- 0 pts** **Correct**

**- 2 pts** Incorrect

### 4.5 n items in k buckets 2 / 2

✓ **- 0 pts** **Correct**

**- 2 pts** Incorrect

**- 1 pts** True, but can be more precise

QUESTION 5

## Binary Search Trees 5 pts

### 5.1 Is this a valid AVL tree? 1 / 1

✓ **- 0 pts** **Correct**

- **1 pts** Incorrect

### 5.2 Are all edges in MST? 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 5.3 Result of remove 12 3 / 3

✓ - **0 pts Correct**

- **1 pts** Used in-order predecessor (not the class convention), but otherwise correct.

- **1 pts** Minor error

- **3 pts** Blank or substantially incorrect

- **1 pts** Rebalance not needed

QUESTION 6

## AVL Trees 10 pts

### 6.1 insert(26) 3 / 4

- **0 pts** Correct

✓ - **1 pts Minor error**

- **4 pts** Blank or substantially incorrect

### 6.2 insert(10 4 / 4

✓ - **0 pts Correct**

- **1 pts** Minor error

- **2 pts** Incorrect rotation / rotation in wrong location

- **4 pts** Blank, substantially incorrect, or no rotation

### 6.3 find(key) time complexity upper bound 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 6.4 find(key) time complexity lower bound 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

QUESTION 7

## Graph 1 8 pts

### 7.1 BFS 1 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 7.2 BFS 2 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 7.3 BFS 3 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 7.4 DFS 1 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 7.5 DFS 2 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 7.6 DFS 3 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 7.7 Simple? 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

### 7.8 DAG? 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

QUESTION 8

## Graph 2 6 pts

### 8.1 MST Cost 2 / 2

✓ - **0 pts Correct**

- **2 pts** Incorrect

- **1 pts** Off by 1 / arithmetic error

### 8.2 MST Edges 1 / 1

✓ - **0 pts Correct**

- **1 pts** Incorrect

8.3 MST 1 **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

8.4 MST 2 **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

8.5 MST 3 **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

QUESTION 9

## Graph 3 3 pts

9.1 Topological Order 1? **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

9.2 Topological Order 2? **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

9.3 Topological Order 3? **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

QUESTION 10

## Graph 4 9 pts

10.1 Dijkstra's Visit Order **7 / 7**

✓ **- 0 pts** **Correct**

**- 1 pts** One vertex out of order

**- 2 pts** Two vertices out of order

**- 5 pts** Multiple vertices out of order

**- 7 pts** Blank or substantially incorrect

10.2 Length of shortest page from A to F **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

**- 0.5 pts** Path given, not length (but correct path)

10.3 Edges on shortest path from A to B **1 / 1**

✓ **- 0 pts** **Correct**

**- 1 pts** Incorrect

CSE 247 Fall 2021 Exam 2 Due at end of session

This exam is: **closed-book**, **NO electronic devices allowed**, and **closed-notes**. The exception is the "sage page" of the designated size on which you may have notes to consult during the exam.

Be sure you: **Provide legible answers in designated areas (credit will not be given for work that is difficult to read or not where expected); Clearly fill in circles (●) on multiple choice questions. Questions with circles require one choice; Leave the exam stapled together in its original order; Do *NOT* attach any other pages to the exam.** You are welcome to use the blank space on the exam for any scratch work.

**If there are multiple "correct" answers for complexity, always pick the one that's the "best fit" (the lowest of the valid upper bounds or the highest of the valid lower bounds).:**

If you need to leave the room for any reason prior to turning in your exam, you must leave your exam and any electronic devices with a proctor. **We do not clarify or explain anything during the exam session. State your assumptions if something is unclear and do the best you can.**

| Question: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Points: | 24 | 15 | 10 | 10 | 5 | 10 | 8 | 6 | 3 | 9 | 100 |

**You must complete all the identifying information below correctly. Failure to do so is grounds for a zero on this exam:**

1. Name (print clearly): Byeongchan Gwak

2. Student ID (print clearly; 1 digit per underline): 5 0 1 0 2 6

3. You must sign the pledge below for your exam to count. The penalty for cheating will be decided during academic integrity review, but the instructors will recommend an F in this course as the minimum penalty.

**I have read the instructions on this page and I will neither give nor receive any unauthorized aid on this exam.**

Byeongchan Gwak 곽

(Sign above)

⟹ ***Do not proceed until told to do so!*** ⟸

⟹ ***Initial the top right of each page before starting*** ⟸

1. (24 points) Data structure operations and common algorithms (true/false and multiple choice: See instructions on cover!):

(1) The best bound (in terms of $n$) that can be *guaranteed* for any sequential, comparison-based sort on unknown data (assume data is "worst case possible" for the algorithm) is:

○ $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ● $O(n \cdot \log(n))$

(2) The upper bound for a put() in a hash table-based map that uses separate chaining (there are no other assurances) is:

○ $O(\log(n))$ ○ $O(1)$ ● $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

(3) In terms of the height, $h$, the upper bound on an insert in a binary search tree is:

○ $O(\log(h))$ ○ $O(1)$ ● $O(h)$ ○ $O(h^2)$ ○ $O(h \cdot \log(h))$

(4) In terms of the height, $h$, the upper bound on an insert in an *AVL search tree* is:

○ $O(\log(h))$ ○ $O(1)$ ● $O(h)$ ○ $O(h^2)$ ○ $O(h \cdot \log(h))$

(5) An adjacency matrix is used to represent a simple, directed graph. The time complexity of identifying all the outgoing edges from a specific vertex is:

○ $O(\log(|V|))$ ○ $O(1)$ ● $O(|V|)$ ○ $O(|V|^2)$ ○ $O(|V| \cdot \log(|V|))$

(6) An *unordered linked list* is used to implement a *set*. The exists(k) method would have a time complexity upper bound of:

○ $O(\log(n))$ ○ $O(1)$ ● $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

(7) An *ordered array list* is used to implement a *set*. The exists(k) method would have a time complexity upper bound of:

● $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

(8) All directed acyclic graphs have a single valid topological order:

● True ○ False

(9) A single execution of Dijkstra's algorithm will identify the shortest path between any two vertices:

○ True ● False

(10) When run on any graph that does have a topological order, both Prim's algorithm and Kruskal's algorithm will always result in the same set of edges for the spanning tree:

○ True ● False

(11) A "Bucket Sort" distributes items to buckets based on their key. This phase of the sort is stable.

● True ○ False

(12) Depth First Search starting from the root of a binary tree will explore nodes in-order by level (that is, all the root's children will be processed/visited before its grandchildren, etc.):

○ True ● False

(13) All valid AVL trees are also valid Binary Search Trees:

● True ○ False

2. Sorting out concepts

✓ (1) (4 points) Professor P makes the statement "The general lower bound on all sequential sorting algorithms is $\Omega(n)$" ("sequential" here means on a single processor and with the types of algorithms we've covered this semester on the types of computers we've been considering). Provide a justification for this statement. You don't have to provide a formal proof, but you should provide a compelling justification that other CSE 247 students would accept:

All Sequencial Sorting algorithms have two phase.

Get a value from a source and place the value in order.

Even if the getting and placing method have the time complexity of $O(1)$,

we need to lookup at least all the value $n$.

In short, $n$ values $\cdot$ $O(1) = O(n)$. Therefore, general lower bound is $\Omega(n)$

(2) Selecting and using sorts

Professor P, who teaches very large on-line courses, stores records on all their students (current and past) in an array. The records are kept in <u>ascending order</u> by student ID number, which is a 9 digit integer.

i. (2 points) Due to a data entry error, two students IDs were entered incorrectly. Those two records have been updated with the correct IDs, but they are now likely in the wrong place in the array. The best sort to fix this inconsistency in this case is:

○ Selection Sort ● Insertion Sort ○ Heap Sort ○ Merge Sort

ii. (2 points) Professor P wants to find if student X has taken a course from Prof. P. Given X's ID number, this can be determined in:

● $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

✓ iii. (3 points) Every semester Professor P randomly selects one student to get an "A" independent of their performance in the course. Unfortunately, Professor P's code had several errors that randomly swapped 90% of the records. That is, the values in the array are randomized. Professor P will use either Radix Sort of Heap Sort to re-order the data. Provide a clear equation of when the two are likely to have nearly equivalent performance in terms of $n$ (when they are equivalent in the asymptotic sense):

When $n$ = ________________ $n \cdot \log n$

✓ iv. (2 points) At the exact point where they are equivalent asymptotically, which is most likely to be "faster" due to the overhead of the other algorithm:

● Radix Sort ○ Heap Sort

✓ v. (2 points) Which sort is expected to perform best if $n$ was (somehow) significantly larger than the point where they are equivalent asymptotically:

○ Radix Sort ● Heap Sort

3. Graph Representations

(1) (2 points) The lower bound on *space complexity* used to store a graph in an adjacency *matrix* representation is:

○ $\Omega(\log(|V|))$ ○ $\Omega(1)$ ○ $\Omega(|V|)$ ● $\Omega(|V|^2)$ ○ $\Omega(|V| \cdot \log(|V|))$

(2) (8 points) Consider the following partial implementation for a *unordered* graph that utilizes an *edge list* implementation:

```
class Edge {
  Vertex start;
  Vertex stop;
}
public class Graph {
   LinkedList<Edge> edges;
   LinkedList<Vertex> vertices;
   ...
```

Provide a complete Java method (declaration/signature and body) for a incidentEdges (...) method that would return a list of all the incident edges to vertex (v), where v is provided as a parameter to the method:

```
LinkedList<Edge> incidentEdges(Vertex v) {
    LinkedList<Edge> res= new LinkedList<Edge>();

    for (Edge currE: edges) {
        if (currE.start.equals(v)) {
            res.add(currE);
        } else if (currE.stop.equals(v)) {
            res.add(currE);
        }
    }

    return res;
}
```

4. Mapping:

(1) The pseudo-code below will take elements from one map and store some of them in another map:

**Algorithm:** remapping(oldMap, newMap, listOfKeys)

**Input** : oldMap - a map of items
**Input** : newMap - a new map that doesn't contain any items
**Input** : listOfKeys - a list of all the keys to transfer to the new map

```
n ← listOfKeys.length()
for i ← 0 to n - 1 do
    k ← listOfKeys.get(i)
    v ← oldMap.get(k)
    newMap.put(k, v)
end
```

For each of the conditions below give a simplified asymptotic time complexity. *Assume that n represents all of the keys in the oldMap.*

i. (2 points) Version 1:

n·( list 1
o.get log n
n.put 1 )

- oldMap uses an ordered array
- newMap is a hash table with separate chaining that uses rehashing and the simple uniform hashing assumption (SUHA) applies for the keys being used
- listOfKeys uses an array

○ $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ● $O(n \cdot \log(n))$
○ None of the above

ii. (2 points) Version 2:

n ( list 1
o.get log n
n.put n )

- oldMap uses an ordered array
- newMap is a hash table with separate chaining that uses rehashing and the simple uniform hashing assumption (SUHA) *does NOT apply*
- listOfKeys uses an array

○ $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ● $O(n^2)$ ○ $O(n \cdot \log(n))$
○ None of the above

iii. (2 points) Version 3:

n· ( list n
o.get log n
n.put 1 )

- oldMap uses ordered array
- newMap is a hash table with separate chaining that uses rehashing and the simple uniform hashing assumption (SUHA) applies for the keys being used
- listOfKeys uses a *linked list*

○ $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ● $O(n^2)$ ○ $O(n \cdot \log(n))$
○ None of the above

iv. (2 points) Version 4:

- oldMap is a *hash table* with separate chaining and the simple uniform hashing assumption (SUHA) applies for the keys being used
- newMap is an *unordered linked list*
- listOfKeys uses an *array*

n. lin 1
o.get 1
n.put n

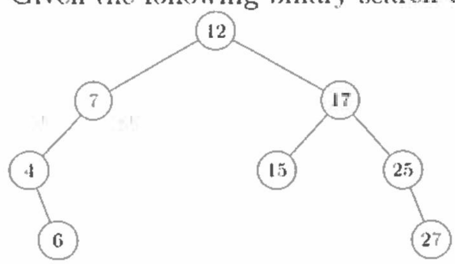
○ $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ● $O(n^2)$ ○ $O(n \cdot \log(n))$
○ None of the above

(2) (2 points) A hash table has $k$ buckets and $n$ values stored in it ($n$ is bigger than $k$). It does not utilize rehashing, but the simple uniform hashing assumption (SUHA) applies. How many items are expected to be in each bucket?

Items in each bucket: $\frac{n}{K}$

5. Binary Search Tree Operations

Given the following binary search tree:

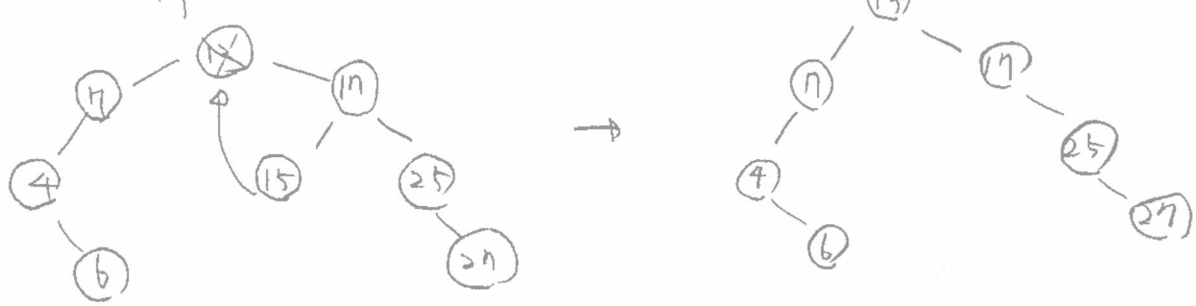



(1) (1 point) Would this be considered a valid AVL tree?
○ Yes ● No

(2) (1 point) *Any* binary search's Minimal Spanning Tree (MST) would include all the edges in the tree:
● True ○ False

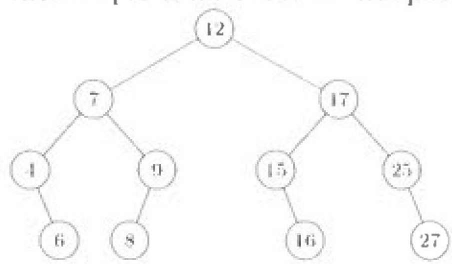
(3) (3 points) Show the tree that will result when 12 is removed (a BST remove, not an AVL remove). Use the conventions used in class for selecting any replacement nodes. Sketch the full, final tree in the space below:

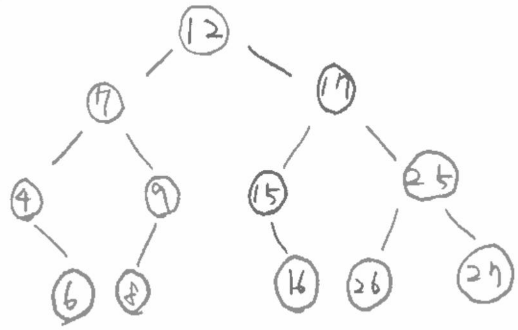
step 1 : find the 12's next node
step 2 : Swap the nodes.
step 3 : Delete the 12.



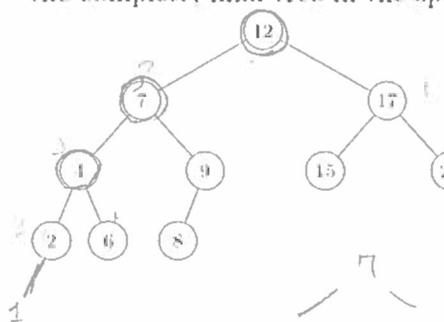
6. AVL Tree operations

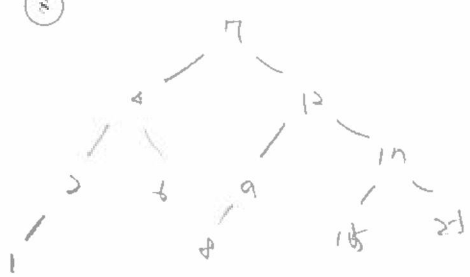
(1) (4 points) Given the following *AVL* tree, show the tree that will result when 26 is inserted (show the complete, final tree in the space to the right):

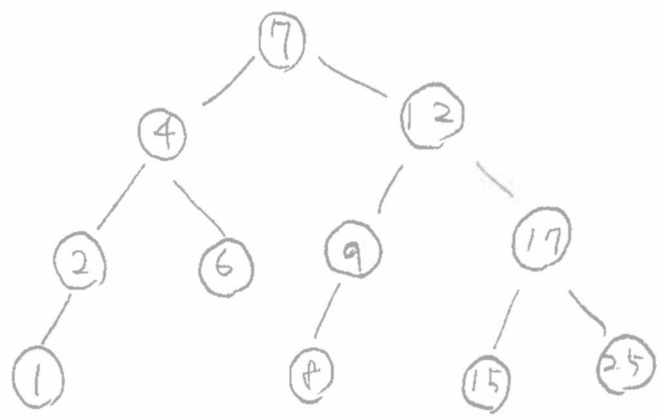



(2) (4 points) Given the following *AVL* tree, show the tree that will result when 1 is inserted (show the complete, final tree in the space to the right):




(3) (1 point) In terms of $n$, a find(key) operation on an AVL tree has an upper bound of:

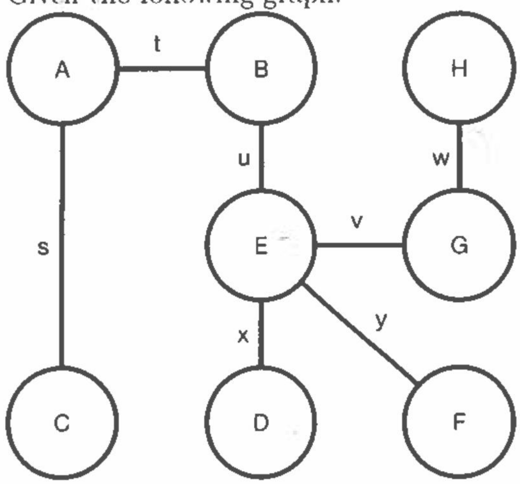
● $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

(4) (1 point) and the find(key) has a lower bound of:

○ $\Omega(\log(n))$ ● $\Omega(1)$ ○ $\Omega(n)$ ○ $\Omega(n^2)$ ○ $\Omega(n \cdot \log(n))$

7. Graphs and Graph Algorithms 1

Given the following graph:




Assume nothing is known about the order of edges within a vertex. The problems below talk about "visiting" nodes. This refers to when the algorithm would start processing the node's edges.

(1) Consider Breadth First Search (BFS) on this graph:

i. (1 point) BFS starting from $B$ *could* visit nodes in the order: $B, A, E, H, G, C, D, F$

○ True ● False

ii. (1 point) BFS starting from $H$ *could* visit nodes in the order: $H, G, E, D, F, B, A, C$

● True ○ False

iii. (1 point) BFS starting from $E$ *could* visit nodes in the order: $E, D, F, B, G, A, C, H$

○ True ● False

(2) Consider Depth First Search (DFS) on this graph:

i. (1 point) DFS starting from $B$ *could* visit nodes in the order: $B, A, C, H, G, E, D, F$

○ True ● False

ii. (1 point) DFS starting from $H$ *could* visit nodes in the order: $H, G, E, D, F, B, A, C$

● True ○ False

iii. (1 point) DFS starting from $E$ *could* visit nodes in the order: $E, D, F, B, G, A, C, H$

○ True ● False

(3) Graph classifications:

i. (1 point) This graph is simple:

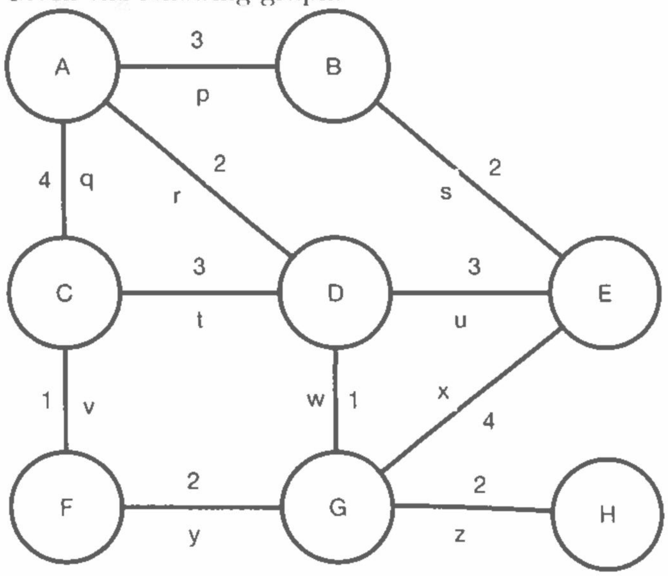
● True ○ False

ii. (1 point) This graph is a DAG:

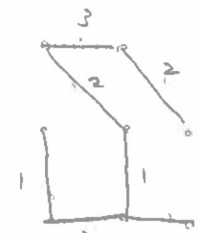
○ True ● False

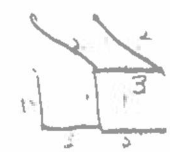
8. Graphs and Graph Algorithms 2

Given the following graph:




| | | |
|---|---|---|
| 1 | A | 0 |
| 7 | B | ~~∞~~ 3 |
| 6 | C | ~~∞~~ ~~4~~ ~~3~~ 1 |
| 2 | D | ~~∞~~ 2 |
| 8 | E | ~~∞~~ ~~3~~ 2 |
| 5 | F | ~~∞~~ 2 |
| 3 | G | ~~∞~~ 1 |
| 4 | H | ~~∞~~ 2 |





(1) (2 points) What is the sum of the weights in a minimal spanning tree for this graph: 13

1+1+2+2+2+2+3

(2) (1 point) How many edges are in the minimal spanning tree for this graph: 7

(3) (1 point) Do the edges $s, u, w, v, r, z, y$ represent a minimal spanning tree?

2 3 1 1 2 2 2

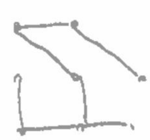
● Yes ○ No

(4) (1 point) Do the edges $v, z, s, r, w, y, t, q$ represent a minimal spanning tree?

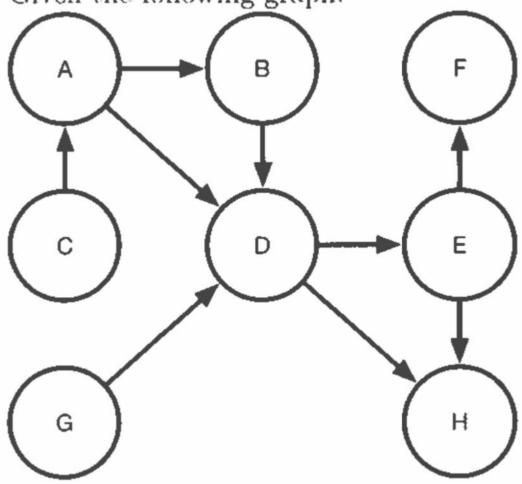
○ Yes ● No

(5) (1 point) Do the edges $p, r, s, w, v, y, z$ represent a minimal spanning tree?

● Yes ○ No



9. Graphs and Graph Algorithms 3
Given the following graph:



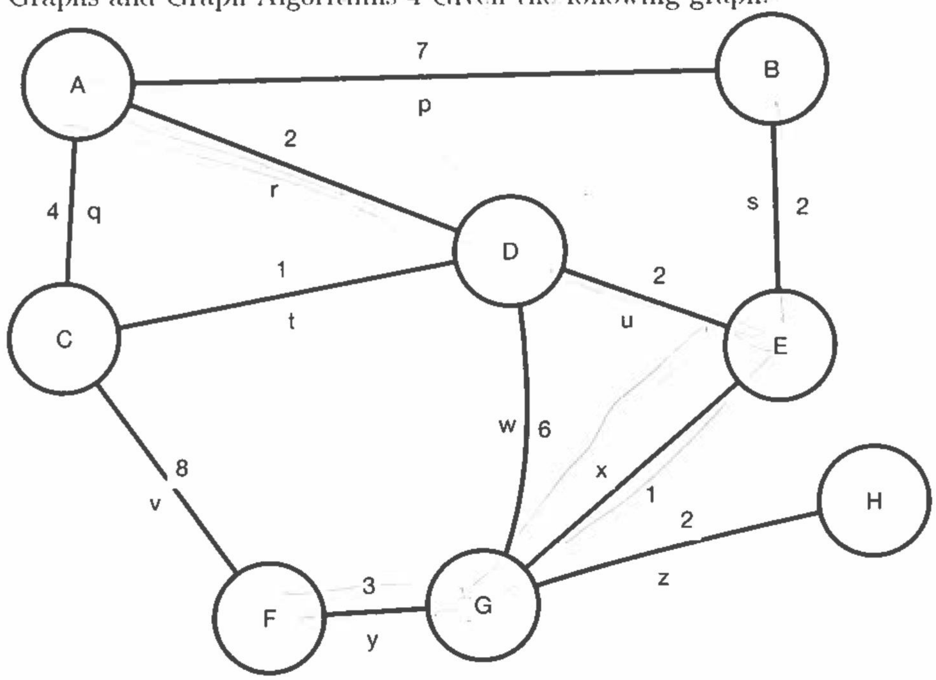

**Which of the following is a valid topological order:**

(1) (1 point) $C, G, A, B, D, E, H, F$
- [x] Yes
- [ ] No

(2) (1 point) $C, G, A, B, D, H, E, F$
- [ ] Yes
- [x] No

(3) (1 point) $C, A, B, G, D, E, H, F$
- [x] Yes
- [ ] No

10. Graphs and Graph Algorithms 4 Given the following graph:




① A | 0
⑥ B | ~~∞~~ ~~7~~ 6
③ C | ~~∞~~ ~~4~~ 3
② D | ~~∞~~ 2
④ E | ~~∞~~ 4
⑧ F | ~~∞~~ ~~11~~ 8
⑤ G | ~~∞~~ ~~8~~ 5
⑦ H | ~~∞~~ 7

(1) (7 points) Assuming Dijkstra's algorithm starts with node $A$ as the "source", which order will nodes be visited (removed from the priority queue). Provide a comma separated list. The first value has been provided for you:

Order of visits: A, D, C, E, G, B, H, F

(2) (1 point) The length of the shortest path from $A$ to $F$ is: 8

(3) (1 point) The number of edges on the shortest path from $A$ to $B$ is: 3