CSE 373 18au: Final Exam (key)

Name:

UW email address:

Instructions

- **Do not start the exam until told to do so**.
- You have 110 minutes to complete the exam.
- This exam is closed book and closed notes.
- You may not use a calculator, cell phone, or any other electronic devices.
- Write your answers neatly in the provided space.
  Be sure to leave some room in the margins: we will be scanning your answers.
- If you need extra space, use the back of the page.
  If you do use the back of the page, write the question number.
- If you have a question, raise your hand.
- This exam is 16 pages long. **Check that you have all the pages.**
- **Unless specified, when asked for a runtime give the worst-case tight big-$\mathcal{O}$ bound.**

Question	Points
1	17
2	14
Total:	31

Question	Points
3	19
4	21
Total:	40

Question	Points
5	9
6	11
7	11
8	8
9	8
10	12
11	0
Total:	59

Advice

- Read questions carefully. Understand a question before you start writing.
- Write down thoughts and intermediate steps so you can get partial credit.
- The questions are ordered roughly in the increasing order of difficulty.
- Relax. You are here to learn.

Warmup

This part will test the basic concepts in the course. This section has multiple choice questions and short-answer questions.

1. For each of the following questions, choose the best (most correct) option. Circle your answer.

(1.1) *(1 point)* What does ADT stands for?

i. Abstract Definition Type
ii. Assorted Data Types
iii. Abstract Data Type
iv. Arrays with Dynamic Typecasting

(1.2) *(1 point)* How many children can a node have in a binary tree?

i. 2
ii. 1 or 2
iii. 0, 1, or 2
iv. 2 except at the last level

(1.3) *(1 point)* Which of the following statement is true for a binary tree.

i. A child node is always greater than or equal to its parent node
ii. A child node is always smaller than or equal to its parent node
iii. A child node can be smaller, greater, or equal to its parent node.
iv. The left child is smaller than the right child.

(1.4) *(1 point)* What is the minimum height of an AVL tree with n nodes?

i. $\lfloor \log_2(n) \rfloor$
ii. $\lfloor n/2 \rfloor$
iii. $\lfloor \log_2(n) \rfloor - 1$
iv. $\lfloor \log_2(n) \rfloor + 1$

(1.5) *(1 point)* Why would we choose an AVL tree over a binary search tree?

i. To keep binary trees balanced
ii. To guarantee $\mathcal{O}(\log n)$ time operations on the tree
iii. To keep elements sorted
iv. To keep elements ordered to allow binary search

(1.6) *(1 point)* What is the most important requirement for a hash function?

i. It should be quick
ii. It should be stable
iii. It should be uniform
iv. It should be deterministic

(1.7) *(1 point)* What does it mean for a sorting algorithm to be stable?

i. For a given input, the output of the algorithm is always the same.
ii. For any input, the algorithm takes the same amount of time.
iii. For an input with duplicate elements, the original ordering is preserved.
iv. For any input, the algorithm takes a constant amount of additional memory.

(1.8) *(1 point)* Which of the following sorting algorithms is the fastest on average in practice?

i. Heap sort ii. Insertion sort iii. Merge sort iv. Quick sort

(1.9) *(1 point)* Among the following, choose a stable comparison sort with a worst-case $\mathcal{O}\left(n^2\right)$ runtime.

i. Heap sort ii. Insertion sort iii. Merge sort iv. Quick sort

(1.10) *(1 point)* Among the following, choose a sorting algorithm that is preferred when the underlying system is very fast at copying data, but slow in comparing elements.

i. Heap sort ii. Insertion sort iii. Merge sort iv. Quick sort

(1.11) *(2 points)* Consider the following function.

```
1  public int foo(int n) {
2      System.out.println('#');
3      if (n < 5) {
4          return n ;
5      } else if (n < 1000) {
6          return foo(n - 3);
7      } else {
8          return foo(n / 3);
9      }
10 }
```

When asked what is the worst-case tight big-$\mathcal{O}$ runtime of this function `foo()`, Vivian says it is $\mathcal{O}\left(\log_2 n\right)$. Nick says it is $\mathcal{O}\left(\log_3 n\right)$. Who is right?

i. Vivian is right (Vivian's answer: $\mathcal{O}\left(\log_2 n\right)$)

ii. Nick is right (Nick's answer: $\mathcal{O}\left(\log_3 n\right)$)

iii. Both are right

iv. Both are wrong

(1.12) *(1 point)* To topologically sort (order) a graph, what are the requirements for the graph?

i. The graph should be acyclic

ii. The graph should be directed

iii. Both (i) and (ii)

iv. None of the above

(1.13) *(1 point)* What is the time complexity of Dijkstra's algorithm with binary heap?

i. $\mathcal{O}\left(|V||E|\right)$

ii. $\mathcal{O}\left(|V|+|E|\right)$

iii. $\mathcal{O}\left(|V|\log|V|\right)$

iv. $\mathcal{O}\left(|E|+|V|\log|V|\right)$

v. $\mathcal{O}\left(|E|\log|V|+|V|\log|V|\right)$

(1.14) P vs. NP

P stands for Polynomial, and is defined as a set of all decision problems that have an algorithm that runs in polynomial time.

1.14a. *(2 points)* Which of the following time complexity classes are polynomial time? **(Select all correct choices.)**

i. $\mathcal{O}(n)$
ii. $\mathcal{O}(1)$
iii. $\mathcal{O}(2^n)$
iv. $\mathcal{O}(n^2)$
v. $\mathcal{O}(n^{2^8})$
vi. $\mathcal{O}(n^{\log n})$

1.14b. *(1 point)* What does NP stand for?

i. Not Polynomial
ii. Normally Polynomial
iii. Non-deterministic Polynomial
iv. Non-characteristically Polynomial

2. Short answer questions

Use the provided underlined space on the right to write your answer.

(2.1) *(5 points)* Give worst-case tight big-$\mathcal{O}$ bound for the following operations.

2.1a. Finding an element in an unsorted array

2.1a. $\mathcal{O}(n)$

2.1b. Finding an element in a sorted linked list

2.1b. $\mathcal{O}(n)$

2.1c. Finding an element in an AVL tree

2.1c. $\mathcal{O}(\log n)$

2.1d. Finding an element in a min-heap

2.1d. $\mathcal{O}(n)$

2.1e. Finding an element in a binary tree

2.1e. $\mathcal{O}(n)$

(2.2) *(1 point)* How long does it take to run a BFS on a graph? (Give the tight big-O bound).

(2.2) $\mathcal{O}(|V| + |E|)$

(2.3) *(1 point)* How long does it take to run topological sort on a graph? (Give the tight big-O bound).

(2.3) $\mathcal{O}(|V| + |E|)$

(2.4) *(2 points)* How long does it take to find strongly connected components in a directed graph? (Give the tight big-O bound).

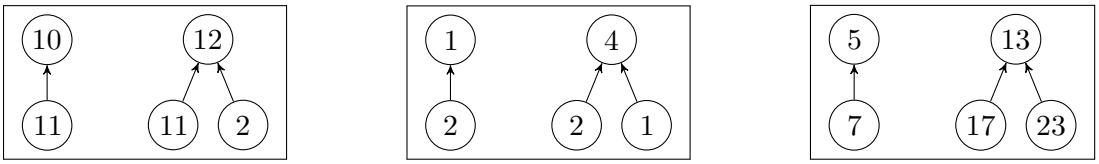
(2.4) $\mathcal{O}(|V| + |E|)$

(2.5) *(1 point)* Consider the following disjoint set. The **rank of the left tree is 2** and the **rank of the right tree is 1**. Suppose we do `union(2, 3)` using union-by-rank optimization. In the resulting union tree, how many nodes would point directly to node 7?

(2.5) 2

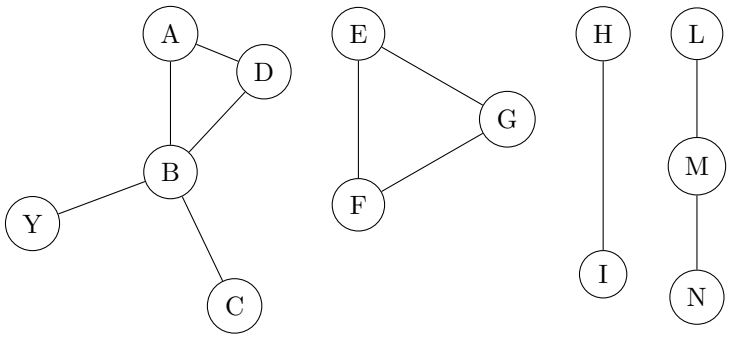
(2.6) *(1 point)* Among the following three collection of sets (a, b, and c), list the valid disjoint set(s).

(2.6) c



(a) (b) (c)

(2.7) Consider the following undirected, weighted graph (weight of each edge is 1).



Answer the following questions about the graph

2.7a. *(½ point)* What is the number of edges in this graph?

2.7a. 11

2.7b. *(½ point)* What is the maximum degree of the graph?

2.7b. 4

2.7c. *(1 point)* What is the length of the longest path in this graph?

2.7c. 3

2.7d. *(1 point)* What is the number of edges in a minimum spanning forest of the above graph?

2.7d. 9

Basic

This part will test your ability to apply basic techniques covered in lecture and reinforced through sections and homeworks.

3. For each of the following questions, choose the best option (most correct). Circle your answer.

(3.1) *(1 point)* Why it might be faster to iterate over elements in an array as compared to iterating over a linked list?

i. Temporal locality
ii. Spatial locality
iii. Both (A) and (B)
iv. It is faster asymptotically

(3.2) *(1 point)* Why it might be faster to check if an edge exists in an adjacency matrix representation of a graph as compared to an adjacency list representation?

i. Temporal locality
ii. Spatial locality
iii. Both (A) and (B)
iv. It is faster asymptotically

(3.3) *(1 point)* In a hash table, primary clustering is likely to happen with which collision resolution strategy?

i. Separating Chaining
ii. Linear Probing
iii. Quadratic Probing
iv. Double Hashing

(3.4) *(2 points)* In an open addressing hash table that uses quadratic probing as its collision resolution strategy, suppose the `delete()` method is naively implemented such that the key that should be deleted is simply removed from the array leaving its slot empty (or null). Which of the followings things could happen with this hash table?

(Select all correct choices)

i. The number of collisions increase
ii. That empty slot never gets used, wasting table space
iii. The hash table inadvertently stores duplicate keys
iv. The `get()` method does not return a value even if the key exists in the table

(3.5) *(2 points)* In an open addressing hash table that uses quadratic probing as its collision resolution strategy, what could happen if the hash table is resized only when $\lambda \geq 1$? (λ is the load factor of the hash table.) **(Select all correct choices)**

i. The table may never resizes
ii. The table size grows slowly [this answer is also acceptable]
iii. The `put()` method does not succeed even if there are empty slots in the table
iv. The `get()` method does not return a value even if if the key exists in the table

(3.6) *(2 points)* Suppose you have to store student records in a hash table. You decided to use student ID and name as the hash key for student records, and you decide to use this following class to store keys.

```
1 public class Student {
2     int studentID;
3     String name;
4 }
```

Which of the following java methods you need to implement/override in this `Student` class? **(Select all that apply.)**

i. `compareTo(T obj)`
ii. `equals(Object obj)`
iii. `hashCode()`
iv. `toString()`

(3.7) *(1 point)* How many MSTs can an undirected connected acyclic simple graph have?

i. Zero
ii. At least one
iii. Exactly one
iv. Zero or more (depends on the graph)

(3.8) *(2 points)* Consider an undirected weighted graph $G = (V, E)$, which has a MST. Suppose we add an edge that is heavier than all the edges in the graph. Which of the following things can we say about the graph?

i. The graph may no longer have a valid MST
ii. The weight of the graph's MST may decrease
iii. The weight of the graph's MST may increase
iv. The weight of the graph's MST remains unchanged

(3.9) *(1 point)* Suppose you have an undirected graph containing $|V|$ nodes and zero edges. What is the maximum number of edges you can add to this graph while maintaining it acyclic? (Assume that you are not allowed to add parallel edges.)

i. $\log |V|$
ii. $|V| - 1$
iii. $|V|$
iv. $|V| + 1$
v. $|V|^2$

(3.10) *(2 points)* Suppose you are given two functions P and Q such that

- $P \in \Omega(Q)$
- $Q \in \Omega(P)$

Which of the following things can we stay about P and Q?

(Recall that the notation $\in$ means 'is an element of'. Informally, you can think of $\in$ as 'is'.)

i. $P \in \Theta(Q)$
ii. $Q \in \Theta(P)$
iii. Both (i) and (ii)
iv. Neither

True or false questions. (1 point per question.)

(3.11) In a dictionary, a successful `put()` call always increases the dictionary size by 1.

i. True

ii. False

(3.12) The `get()` operation is asymptotically faster in an AVL tree than in a balanced BST.

i. True

ii. False

(3.13) The worst-case runtime of the most efficient comparison sort is $\mathcal{O}(n)$.

i. True

ii. False

(3.14) Every problem in P is also in NP.

i. True

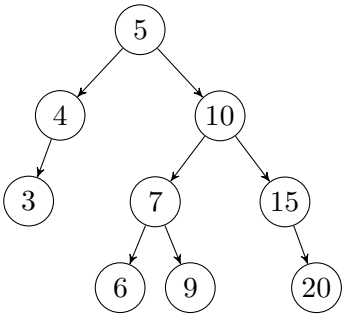
ii. False

4. Short answer questions

Use the provided underlined space on the right to write your answer.

(4.1) *(1 point)* In the following BST, if node 10 is deleted, which node(s) can take its place? (For full credit, list all possible nodes. Assume that only one node can be moved in the tree.)

(4.1) 9 or 15; just 9 is also acceptable



(4.2) *(4 points)* What is the **worst-case tight big-$\mathcal{O}$ bound** for the following operations.

4.2a. To find the maximum value in an AVL tree of size n

4.2a. $\mathcal{O}(\log n)$

4.2b. To find the maximum value in a min-heap of size n

4.2b. $\mathcal{O}(n)$

4.2c. Sorting an array of size n with heap sort

4.2c. $\mathcal{O}(n \log n)$

4.2d. Sorting an array of size n with quick sort

4.2d. $\mathcal{O}(n^2)$

(4.3) *(4 points)* For each of the recurrences below, give the big-Θ bound using the master theorem. No additional explanations are necessary. If the master theorem does not apply, write 'Does not apply'. (Master theorem is given on your reference sheet.)

4.3a.

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2T(n/2) + 1 & \text{otherwise} \end{cases}$$

4.3a. $\Theta(n)$

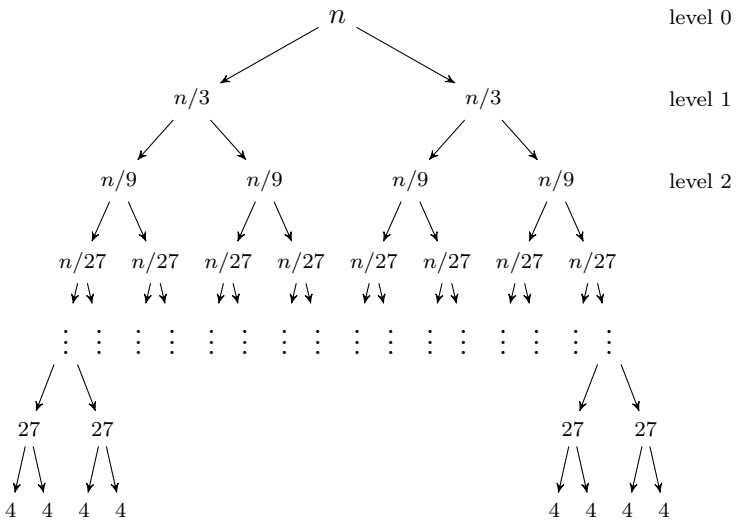
4.3b.

$$T(n) = \begin{cases} 1 & \text{if } n \leq 10 \\ 3T(n/3) + n & \text{otherwise} \end{cases}$$

4.3b. $\Theta(n \log n)$

(4.4) Consider this recurrence and its recurrence tree.

$$T(n) = \begin{cases} 4 & \text{if } n \leq 9 \\ 2T(n/3) + n & \text{otherwise} \end{cases}$$



4.4a. *(1 point)* What is the size of the input on level i

4.4a. $n/3^i$

4.4b. *(1 point)* What is the number of nodes on level i

4.4b. 2^i

4.4c. *(2 points)* What is the total work per recursive level on level i?

4.4c. $n(2/3)^i$

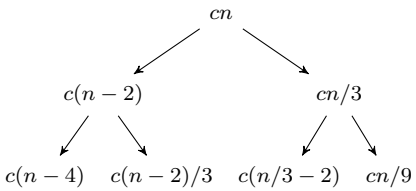
4.4d. *(2 points)* What is the last level of the tree?

4.4d. $\log_3 n - 2$

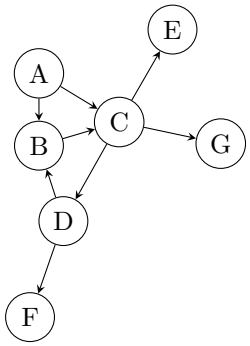
(4.5) *(5 points)* Draw the first three levels of the recurrence tree for the following recurrence. Draw all the nodes on level 0, level 1, and level 2 of the tree. You only need to draw level 0, 1, and 2. **No need to show level 3 or any higher level**. No need to show the last level of the tree.

$$T(n) = \begin{cases} 3 & \text{if } n = 1 \\ T(n-2) + T(n/3) + cn & \text{otherwise} \end{cases}$$

Solution:



(4.6) *(1 point)* List one topological ordering of the following graph. If no ordering exists, briefly explain why.



(4.6) None exists. There is a cycle: B-C-D-B

Applied

This section tests your ability to think a little bit more insightfully. The approaches necessary to solve these problems may not be immediately obvious. Remember to show your work and justify your claims.
For each of the answers, unless specified in the question, describe your algorithm in English or pseudocode. DO NOT write java code. For all questions, the expected answers are at most 4-5 sentences. Longer than necessary answers will not get full credit.

5. Reconstructing optimal BST from a sorted array.
Suppose you are given a sorted (in an increasing order) integer array with no duplicate keys, and from this array you need to create a binary search tree with the minimum possible height.

One approach is to use AVL trees, but that would take $\mathcal{O}(n \log n)$. We can do better (in linear time) by leveraging the fact that the array is sorted and carefully choosing the sequence of values we insert in our BST.

Answer the following the questions as you work your way to identifying the start of that sequence.

(5.1) *(3 points)* Consider this test input array.

0	1	2	3	4	5	6
10	25	41	49	52	74	77

5.1a. What is the minimum possible height for this input array?

5.1a. 2

5.1b. If you just insert the values from the array in the sorted order in a BST, the resulting BST will be a degenerate tree of size 7. To create a BST with minimal height, we need to first insert the value that will be at the root of the minimum height BST. Which value should be inserted first in an empty BST?

5.1b. 49

5.1c. Now, list the next two values should be inserted next in the BST?

5.1c. 25, 74

(5.2) *(3 points)* Now applying similar strategy for an array of size n, list the array indices (in terms of n) of the first three values that should be inserted in a BST? (Don't worry about one-off errors. Just list approximate values.)

(5.2) $\lfloor n/2 \rfloor$, $\lfloor n/4 \rfloor$, $\lfloor 3n/4 \rfloor$

(5.3) *(3 points)* Think about how you might apply divide-and-conquer strategy to solve this problem, and write the recurrence expression for a possible solution.

$$T(n) = \begin{cases} c_1 & \text{if } n \leq 1 \\ 2T(n/2) + c_2 & \text{otherwise} \end{cases}$$

6. Graph problems

(6.1) *(4 points)* Describe how you would find a maximum spanning tree in a given graph G.

A maximum spanning tree of a graph G is a spanning tree of the graph with the maximum sum of edge weights.

Solution: Use Kruskal's algorithm, but sort the edges in reverse (decreasing) order.
Another approach is to multiply all the edge weights by -1 and then find a MST (using either Kruskal's or Prim's algorithm).

(6.2) *(7 points)* Suppose you are given an undirected connected graph $G = (V, E)$, and a Tree T that Charlie claims is a MST of the graph G. Describe how you will verify whether T is a minimum spanning tree of G. What is the runtime of your solution?

Solution: We need to check:

1. T is actually a subgraph of G (i.e., all vertices and edges in T are also in G)
2. T is a tree (i.e., T is connected and acyclic)
3. T is a minimum spanning tree (i.e., weight of T is the same as the weight of a MST of G)

Algorithm:

- Iterate through T and check that every vertex and edge is in G
- Do a DFS and check that T is connected and does not have a cycle.
- Find a MST of G, say S (using either Prim's or Kruskal's algorithm) and then check whether weight of T is the same as the weight of S.
- If the above three statements are true, then T is a MST of G.

Runtime: $\mathcal{O}((|E| + |V|) \log V)$, assuming we used Prim's algorithm to find S. The other steps are graph linear.

Note: There are more efficient (graph linear time) solutions to this problem, but those are beyond the scope of this course.

7. Suppose we model Facebook users with the following graph representation. An undirected unweighted graph where vertices are users and edges represent 'friend' relationships. Thus, if two users are friends, then we have an edge between those two users.

Given this graph representation, explain how will solve the following problems.

(7.1) *(2 points)* How would you find a person with the most friends in the network? State the runtime of your solution.

Solution: Iterate through the graph and find a vertex with the maximum degree.
Runtime: $\mathcal{O}(|V|)$ if using adjacency list representation.
Runtime: $\mathcal{O}(|V| + |E|)$ if doing a BFS/DFS to iterate through the entire graph.

(7.2) *(3 points)* The company wants to introduce a new metric for each user called *networkSize*. The networkSize of a user u is the number of users v such that $u \rightsquigarrow v$, i.e., there is a path from u to v. Describe how you would calculate networkSize for a user u. State the runtime of your solution.

Solution: Run BFS or DFS from u and calculate the number of unique nodes that are visited.
Runtime: $\mathcal{O}(|V| + |E|)$

(7.3) *(6 points)* Suppose you won a competition at Facebook, and you are granted three free friend requests, i.e., any three users you choose will automatically become your friends. You want to choose three users who can maximize your networkSize. Describe an efficient solution for choosing three such users. State the runtime of your approach.

Solution:

- Identify all the components in the graph and compute the size of each component (number of vertices in each component). Can do this with BFS/DFS on the entire graph.
- Pick top three components by their size (i.e., number of users in a component), ignoring the component in which you are present.
- Pick a (any) user from each of those three components.

Runtime: $\mathcal{O}(|V| + |E|)$

8. Finding potential sources of misinformation.
Misinformation is a big concern on social networks. Some claim that there are only a few users who actively spread misinformation on social networks. If we represent a social network as a graph, we can identify potential sources by computing the optimal sources to spread misinformation by tracking the flow of information in the graph.

Let us consider Twitter and represent it as graph. In Twitter, as we know, a relationship between users is not symmetric, i.e., a user u can follow user v, but v may not follow u. So we represent Twitter network with a directed unweighted graph, where vertices are users and edges represent followers, i.e., an edge from u to v means u is being followed by v.

Assume that misinformation flows from users to their followers, to the followers of their followers, and so forth. That is, when a user tweets (or posts) some false information, assume that all followers retweet that information, and the process continues. Thus, misinformation posted by user u spreads to all users t if there is a path from u to t.

(8.1) *(1 point)* Given our graph representation, what is the number of followers of a user u?

i. In-degree of u
ii. Out-degree of u
iii. Total degree of u (sum of in-degree and out-degree)
iv. None of the above

(8.2) *(2 points)* Consider the example graph in Figure 2, which shows a small set of Twitter users and their relationships.

To spread misinformation to all the users in this graph using the least number of users, which user or users should be selected to tweet that misinformation?

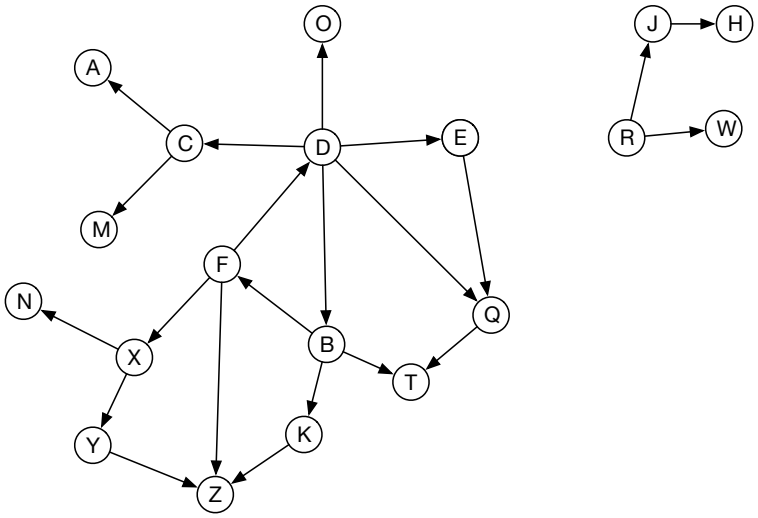


Figure 2: Sample Twitter graph.

Solution: D or F or B, and R

(8.3) *(5 points)* How would you solve the above problem for any graph G. That is, given a directed graph G, describe how would you identify the smallest set of users who can spread misinformation to all the users in the graph. State the runtime of your approach.

Solution:

- Let S be the set of users who can spread misinformation
- Run the SCC algorithm to get the meta graph with all the strongly connected components
- Iterate through the meta graph and for each SCC that has no incoming edge, pick a (any) user from it, and add the user to set S

Runtime: $\mathcal{O}(|V| + |E|)$

9. Bob runs a pizza shop in a small town in Greenland. There are 100 houses in the town connected by roads. Bob delivers to every house in the town. Unfortunately, Bob can only carry one pizza at a time, so after delivering to one house, he has to come back to the shop to pick up another delivery.

   Last week, there was a storm that left 4 feet of snow on all the roads in the town. Clearing snow on a road costs Bob some money, but once the snow is cleared the cost to travel on the cleared road (in either direction) is zero. Bob decides that instead of clearing snow from all the roads on his delivery routes at once, he will clear the snow while delivering pizzas.

   (9.1) *(4 points)* Bob knows he will get an order for each of the 100 houses (they're all regular customers), so he'd like to plan in advance which roads he is going to clear. How should he choose which roads to use to minimize the total cost? State the runtime of your solution.

Solution: Bob should find a MST for the town.

(9.2) *(4 points)* When each new order comes in, how should Bob determine what roads to take to get to that house?

Solution: Bob should run BFS or DFS *on the MST*.
Since in a MST there is only one path between two vertices, that is the shortest path and BFS/DFS would find that path. Running Dijkstra's on a MST also works but is a suboptimal solution than running BFS/DFS on the MST.

10. Somewhat tricky questions

Advice: These are somewhat tricky questions. If you are stuck on any of these problem, come back to those later!

(10.1) *(2 points)* What is the maximum height of an AVL tree with n nodes (assume $n > 10$)?

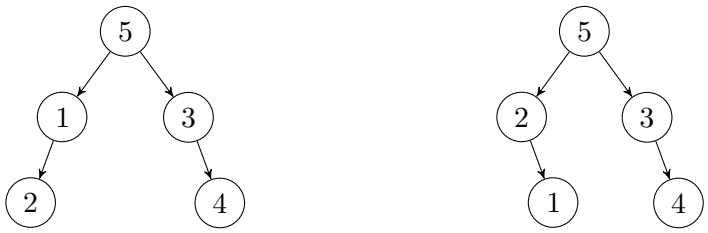
i. $\lfloor \log_2(n) \rfloor$
ii. n
iii. $\lfloor \log_2(n) \rfloor - 1$
iv. $\lfloor \log_2(n) \rfloor + 1$
v. None of the above

(10.2) *(4 points)* Suppose there is an integer binary tree whose in-order traversal sequence is

`2 1 5 3 4`

Draw two binary trees of height 2 with the same in-order traversal sequence. If there is only one such binary tree, mention that in your answer.

Solution: (There are more than two answers.)



(10.3) Following is a pseudocode for a sorting algorithm. Study the code and answer the following questions about the this sorting algorithm.

Note: You have not seen this sorting algorithm in the class. Part of the exercise is to apply your analysis skills. Read the questions before analyzing the code.

```
1 public void fooSort(int arr[])
2 {
3     int n = arr.length;
4     for (int i = 0; i < n-1; i++) {
5         for (int j = 0; j < n-i-1; j++) {
6             if (arr[j] > arr[j+1])
7             {
```

```
8                   // swap temp and arr[i]
9                   int temp = arr[j];
10                  arr[j] = arr[j+1];
11                  arr[j+1] = temp;
12              }
13          }
14      }
15  }
```

10.3a. *(2 points)* What is the worst-case tight big-O runtime?

10.3a. $\mathcal{O}\left(n^2\right)$

10.3b. *(2 points)* Does the above algorithm do an in-place sort?

10.3b. Yes

10.3c. *(2 points)* Does the above algorithm do a stable sort?

10.3c. Yes

11. (2 points (**Extra credit**)) Draw your TA in their natural habitat.