CSE 373 A 19 Sp: Final June 11th, 2019

Name KEY

Student ID#

Instructions

1. This test is closed book, closed calculators, closed electronics.
2. You are permitted a single sheet of 8.5" x 11" notes
3. Do not start the exam until told to do so. You will receive a 10-point deduction if you work before the exam starts or keep working after the instructor calls for papers
4. Please write your answers neatly in the provided space
5. Be sure to **leave some room in the margins** as we will be scanning your answers and it's easy to miss markings too close to the edge of the paper
6. Please note we will only be scanning the front of each page, so please **do not put any work you wish to be graded on the back of a page**
7. If you have a question, please raise your hand to ask the course staff for clarification

Advice

1. **Write down assumptions, thoughts and intermediate steps so you can get partial credit.** Clearly circle your final answer
2. The questions are not necessarily in order of difficulty, skip around

. Graph algorithms

Answer the following questions about using Kruskal's algorithm to find the Minimum Spanning Tree of the Graph in Figure 1.

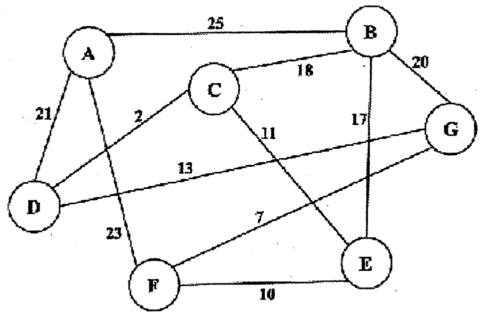


Figure 1 – Graph on which to perform Kruskal's

(A) Sorting Edges

Insert the edges from Figure 2 in the following order into a minimum binary heap using Floyd's Build Heap algorithm.

21 13 20 17 7 10 11 18 23 25 2

1. Fill in Figure 2 with the **initial** internal state of the heap before any percolations.

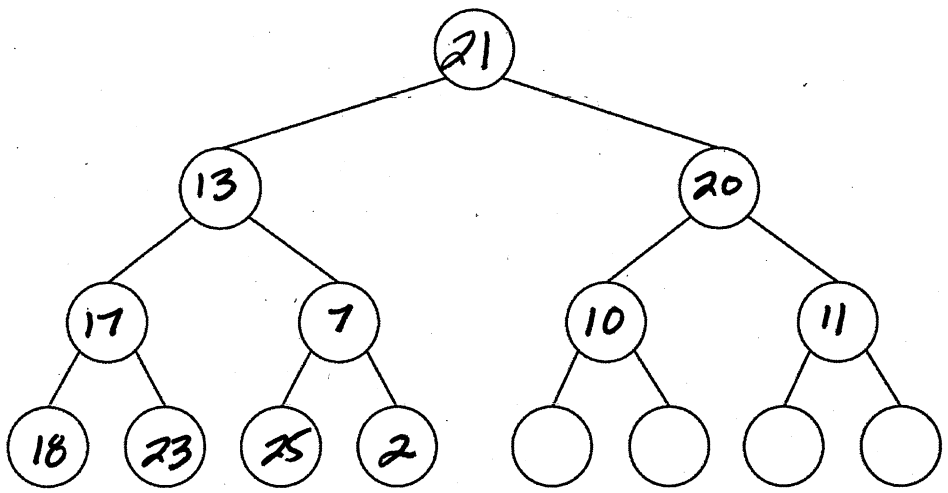


Figure 2 – **Initial** state of heap before any percolations.

2. Fill in Figure 3 with the **final** internal state of the heap. Show your work to receive partial credit.

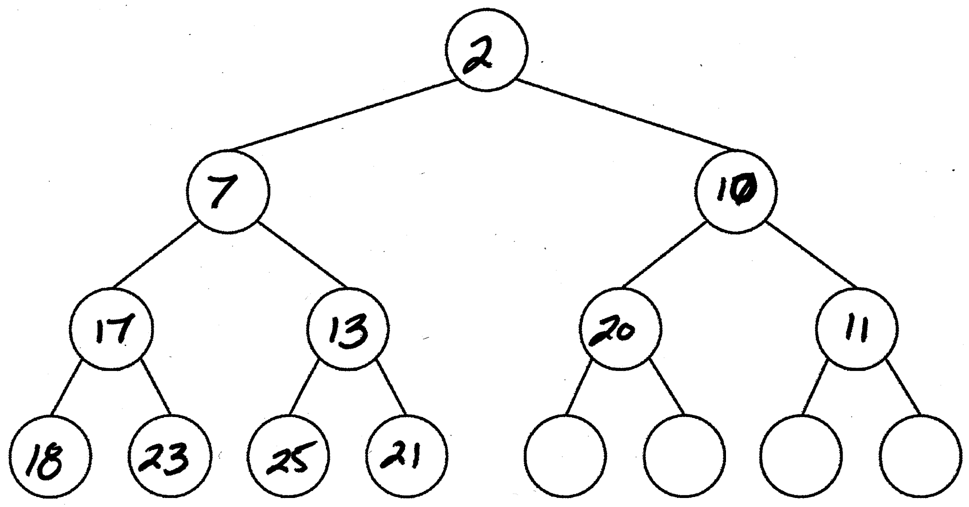
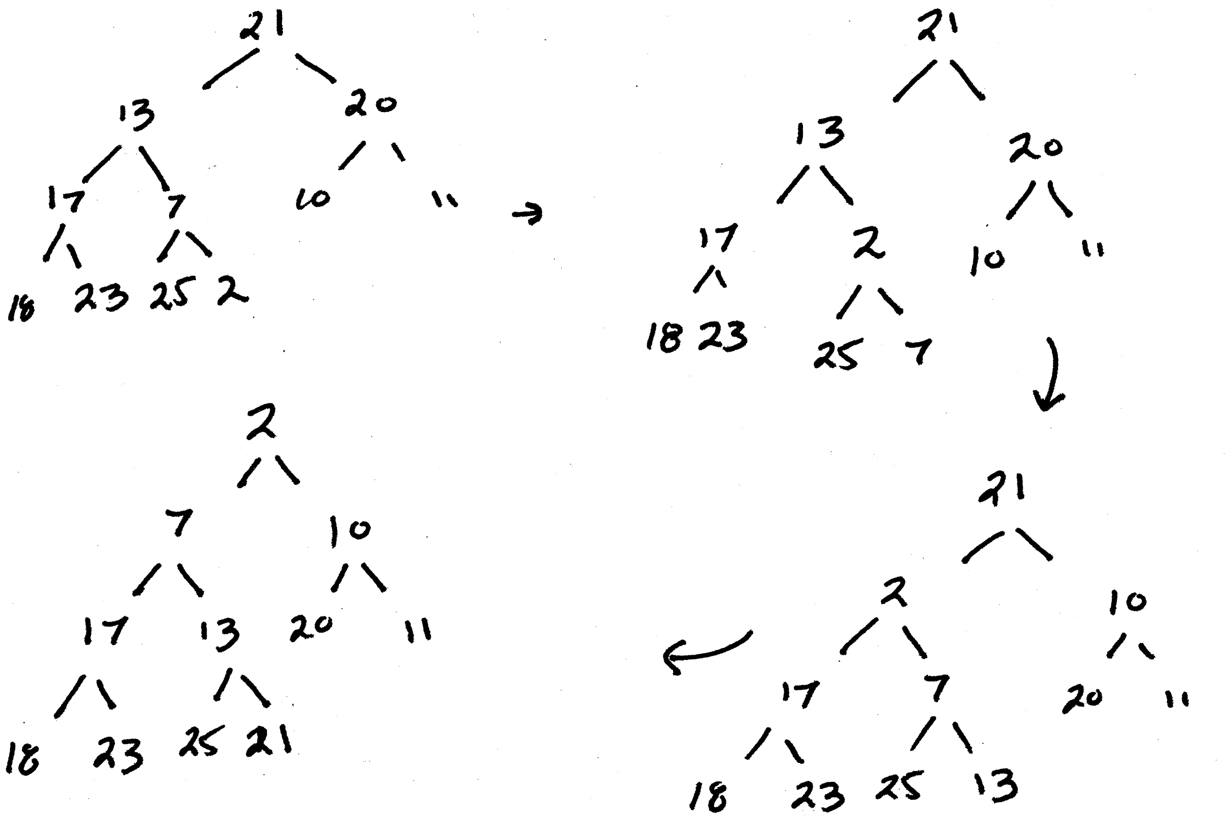


Figure 3 – **Final** state of heap ~~before~~ AFTER any percolations.



3. Fill in the following array representing the final state of the heap from the previous question.

0	1	2	3	4	5	6	7	8	9	10
2	7	10	17	13	20	11	18	23	25	21

(B) Disjoint Set

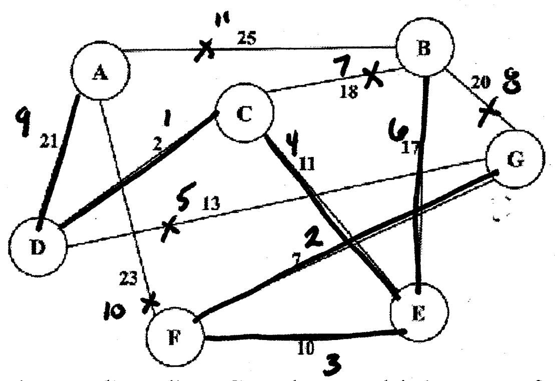
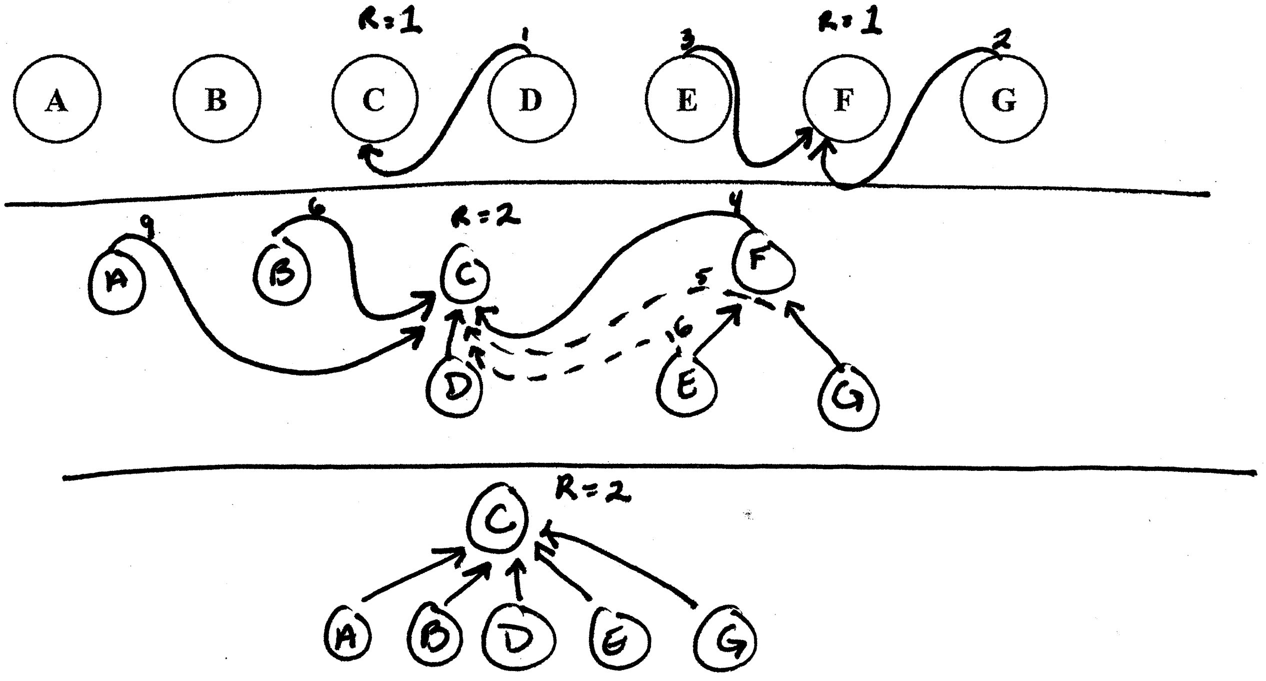


Figure 1 (reproduced) – Graph on which to perform Kruskal's

1. Show the final internal state of the disjoint set after running Kruskal's. Assume the disjoint set uses both **union-by-rank** and **path compression** optimizations. Break ties when unioning alphabetically such that A > B. To receive partial credit, show your work by drawing the intermediary stages of the disjoint set. Clearly circle your final image.



2. Fill in Figure 5 representing the final state of your disjoint set from above. Assume your disjoint set includes the following mapping between vertex value and array index:

Vertex Value	A	B	C	D	E	F	G
Array Index	0	1	2	3	4	5	6

Figure 4 - Mapping from Vertex value to array index

0	1	2	3	4	5	6
2	2	-3	2	2	2	2

Figure 5 - Final state of Disjoint Set (**NOTE** your answers should be int values)

(C) Minimum Spanning Tree

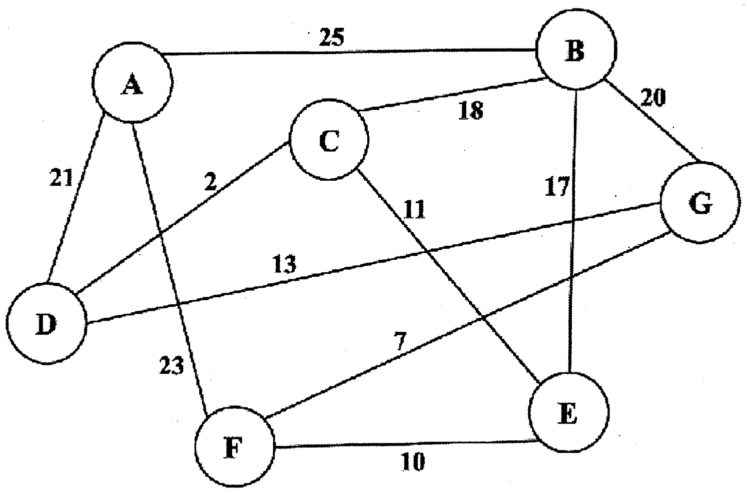


Figure 1 (reproduced) – Graph on which to perform Kruskal's

Draw out the final MST by adding the missing edges to the following figure.

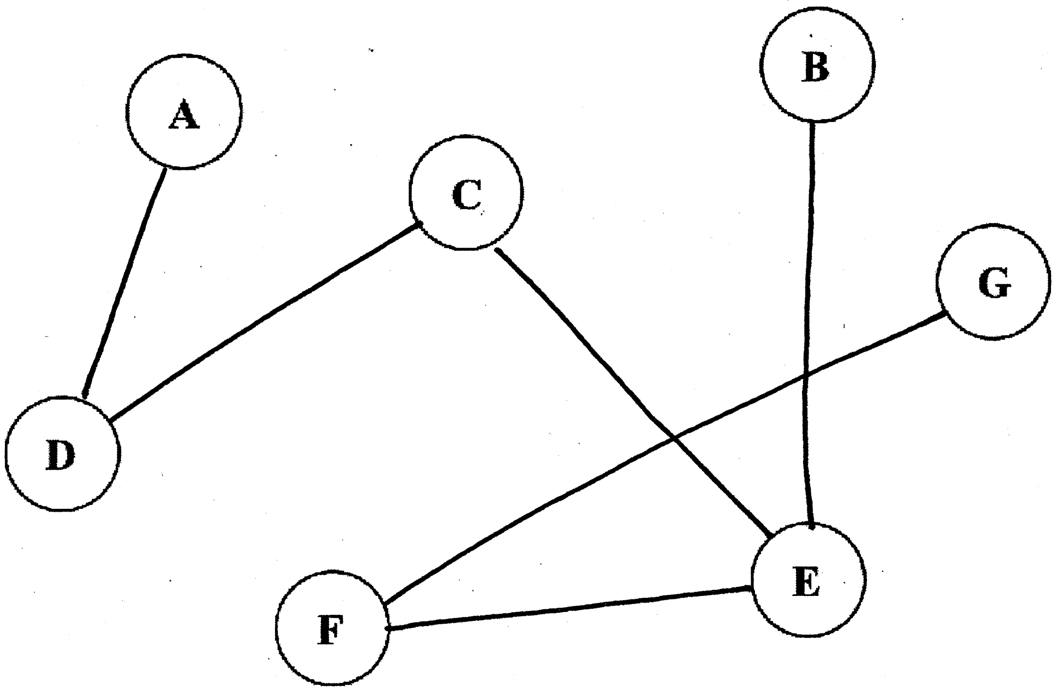


Figure 6 – Final MST of graph in Figure 1

. Code Modeling

(A) Creating Students

```
1  public ChainedHashSet<Students> q1(
                         ArrayDictionary<String, Double> classList,
                         ChainedHashDictionary<String, Double> gradeChange) {
2    ArrayHeap<Student> ranking = new ArrayHeap<Student>();
3    for (KVPair<String, Double> entry : classList) {
4      double gpa = entry.getValue();
5      try {
6        gpa += gradeChange.get(entry.getKey());
7      } catch (NoSuchKeyException e) { }
8      Student s = new Student(entry.getKey(), gpa);
9      ranking.add(s);
10   }
11
12   ChainedHashSet<Student> students = new ChainedHashSet<Student>();
13   while (!ranking.isEmpty()) {
14     Student partner1 = ranking.removeMin();
15     if (!ranking.isEmpty()) {
16       Student partner2 = ranking.removeMin();
17       partner1.partners.add(partner2);
18       partner2.partners.add(partner1);
19       students.add(partner1);
20       students.add(partner2);
21     }
22   }
23   return students;
24 }
```

Consider the following declaration of the `Student` object

```
public class Student implements Comparable<Student> {
    public String name;
    public double GPA;
    public DoubleLinkedList<Student> partners;
    ...
```

Assume the count of entries in the `ArrayDictionary` "`classList`" is **n**. The "`gradeChange`" dictionary will have a maximum size of n. Assume the `hashCode()` function for `String` and `Student` runs in constant O(1).

1. What is the simplified tight O bound of the <u>worst</u>-case runtime of line 6?	O(n)	**5.** What is the simplified tight O bound of the <u>worst</u>-case runtime of line 17 (same as line 18)?	O(1)
2. What is the simplified tight O bound of the <u>average</u>-case runtime of line 9 (assume no resizing)?	O(log n)	**6.** What is the simplified tight O bound of the <u>average</u>-case runtime of line 19 (assume no resizing)?	O(1) || O(λ)
3. What is the simplified tight O bound of the <u>worst</u>-case runtime of the for loop between lines 3 and 10?	$O(n^2)$	**7.** What is the simplified tight O bound of the <u>worst</u>-case runtime of while loop between lines 13 and 22?	$O(n^2)$
4. What is the simplified tight O bound of the <u>worst</u>-case runtime of line 14 (same as line 16)?	O(log n)	**8.** What is the simplified tight O bound of the <u>worst</u>-case runtime of q1 method? (make no assumptions)	$O(n^2)$

(B) Graph Connections

```
1  public static Student q2(Graph<Student, Partnership> g) {
2    ArrayHeap<StudentValue> mostConnected = new ArrayHeap<StudentValue>();
3    ChainedHashDictionary<Student, ISet<Partnership>> adjacencyList = g.graph;
4    for (KVPair<Student, ISet<Partnership>> v : adjacencyList) {
5      IList<Student> connections = new DoubleLinkedList<Student>();
6      Student current = v.getKey();
7      connections.add(current);
8      ChainedHashSet<Student> visited = new ChainedHashSet<Student>();
9      visited.add(current);
10     while (connections.size() > 0) {
11       current = connections.remove();
12       ChainedHashSet<Partnership> partners = adjacencyList.get(current);
13       for (Partnership p : partners) {
14         Student other = p.getOtherVertex(current);
15         if (!visited.contains(other)) {
16           connections.add(other);
17           visited.add(other);
18         }
19       }
20     }
21     StudentValue entry = new StudentValue(current, visited.size());
22     mostConnected.add(entry);
23   }
24   while (mostConnected.size() > 1) {
25     mostConnected.removeMin();
26   }
27   return mostConnected.removeMin().stu;
28 }
```

Consider the following declaration of the `Partnership` object

```
public class Partnership implements IEdge<Student>, Comparable<Partnership> {
   private Student v1;
   private Student v2;
   private int weight;
   ...
```

And the `StudentValue` object

```
public class StudentValue implements Comparable<StudentValue> {
   public Student stu;
   public int val;
   ...
```

Assume the number of `Student` vertices in the `Graph` "g" is **V** and the number of Partnership edges **E**

1. What is the simplified tight O bound of the <u>worst</u>-case runtime of line 15?	$O(V)$	**5.** What is the simplified tight O bound of the <u>worst</u>-case runtime of line 25?	$O(\log V)$
2. What is the simplified tight O bound of the <u>worst</u>-case runtime of line 16?	$O(1)$	**6.** What is the simplified tight O bound of the <u>worst</u>-case runtime of a **single iteration** of the for loop between lines 4 and 23? → BFS	$O(V+E)$
3. What is the simplified tight O bound of the <u>average</u>-case runtime of line 17? (assume no resizing)	$O(1)$ or $O(\lambda)$	**7.** What is the **maximum number of iterations** of the for loop between lines 4 and 23?	V
4. What is the simplified tight O bound of the <u>average</u>-case runtime of line 22? (assume no resizing)	$O(\log V)$	**8.** What is the simplified tight O bound of the <u>worst</u>-case runtime of the q2 method? (make no assumptions)	$O(V^2+VE)$

. Graph Modeling

(A) The British Are Coming!

It is 1775 and you are a patriot in the American Revolution. Tensions have been building with the British and the revolutionaries suspect that King George will soon take military action to squash the rebellion. Swift communication will be the key to victory, but spies are everywhere. You have been tasked with mapping out a set of routes that connect all the rebellion's secret strongholds while minimizing the danger of being intercepted by British spies.

You need to calculate the danger level of each route and then select a set of routes that connect all strongholds while minimizing danger. The danger of a route is directly proportional to the sum of British soldiers stationed in the two towns that route connects.

Explain how you would model this scenario as a graph by answering the questions below in bullet points with short sentences.

1. What would your vertices and edges represent?

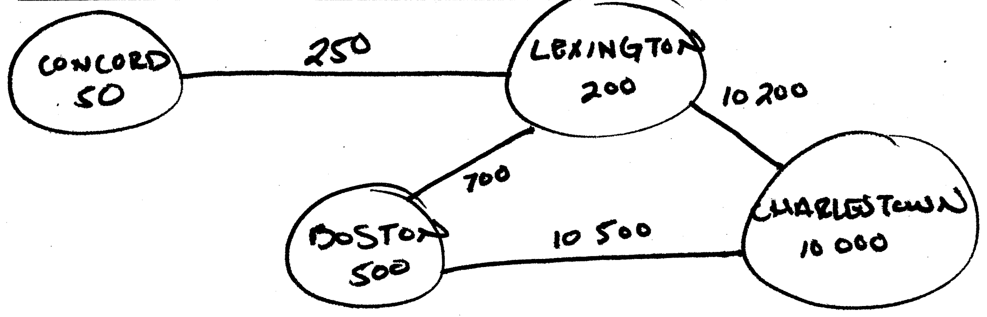
vertices = towns
edges = roads

2. What information would you store for each vertex and each edge?

vertices = name + # of soldiers
edges = sum of soldiers at either vertex

3. Draw out a portion of your graph that would represent the following four towns:

Town	# of British Soldiers	Connected to
Concord	50	Lexington
Lexington	200	Concord, Charlestown, Boston
Boston	500	Lexington, Charlestown
Charlestown	10000	Lexington, Boston



4. Select a graph algorithm and explain how you would you apply it to your graph to determine the danger of the routes based on the presence of British soldiers. Explain how you would store this information.

BFS, as traversing calculate sum of soldiers in V1 + V2 of each edge, store as edge weight.

5. Select a graph algorithm and explain how you would apply it to your graph to decide upon the best set of routes for rebel communications.

KRUSKAL'S, run on graph to find MST with all cities connected, but via routes with least possible danger. (in this case leave out Boston → Charlestown because of high danger)

(B) Note Passing

Imagine you are an American High School student. You have a very important note to pass to your crush, but the two of you do not share a class so you need to rely on a chain of friends to pass the note along for you. A note can only be passed from one student to another when they share a class, meaning when two students have the same teacher during the same class period.

Unfortunately, the school administration is not as romantic as you, and passing notes is against the rules. If a teacher sees a note, they will take it and destroy it. Some teachers are better at intercepting notes than others. The more notes a teacher has intercepted, the more likely it is they will take yours and it will never get to your crush. Find a sequence of handoffs least likely to be intercepted by a teacher. (You do not need to consider time of delivery as a factor)

Explain how you would model this scenario as a graph by answering the questions below in bullet points with short sentences.

1. What would your vertices and edges represent?

vertices = students

edges = classes students share

2. What information would you store for each vertex and each edge?

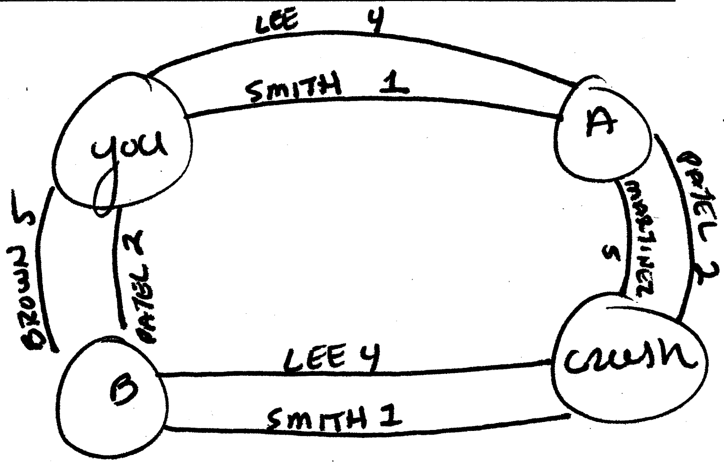
vertices = student name

edge = class info, # of notes intercepted

3. Draw out a portion of your graph that would represent the following four students and their classes:

	Period 1	Period 2	Period 3	Period 4
You	Smith	Patel	Brown	Lee
Friend A	Smith	Martinez	Patel	Lee
Friend B	Lee	Patel	Brown	Smith
Crush	Lee	Martinez	Patel	Smith

Teacher	Notes Intercepted
Smith	1
Martinez	3
Lee	4
Brown	5
Patel	2



4. What graph algorithm could you leverage to find a way to get your note from you all the way to your crush?

DIJKSTRA'S

5. Briefly explain how the algorithm you selected above would leverage the data in your vertices and edges to produce a final solution.

start with "you" as source, find shortest path to target "crush". "shortest" means classes with least number of intercepted notes.

. Design Decisions

(A) ADTs and Data Structures

For each of the given scenarios select an ADT and a specific data structure implementation you would use for each scenario, you can select as many (or as few) from each list as you find appropriate. Be sure to justify your answers with a short sentence or two as appropriate.

ADTs: Dictionary, Priority Queue, Disjoint Set, Set
Data Structures: AVL Tree, ArrayHeap, Hash Table, Array, Linked List, Binary Search Tree

1. You are building a tournament bracket for a friendly softball league. You want to pair up teams to play that are relatively well matched. You can estimate a team's abilities based on their record from the previous year. For example, you want to pair the two teams with the worst record and the two teams with the best records. You can measure how good a team's record is by subtracting the number of games lost from the number of games won. You can assume there is an even number of teams.

(i) Which ADT is the best fit to organize the teams to facilitate creating the matches?

PRIORITY QUEUE

(ii) Describe how you will apply your chosen ADT to create the matches.

put all teams into queue, compareTo based on record, pull out (remove min) 2 at a time to make matches.

(iii) Which data structure would you use to implement your chosen ADT?

ARRAY HEAP

(iv) What about this data structure is uniquely suited for this scenario?

leverages compareTo to efficiently sort into min → max, especially if using Floyd's initially

(v) Describe the algorithm your code would implement to pair up the teams for the first round of games.

compareTo (this.wongames - this.lostgames > other.wongames - other.lostgames)
use floyd's put all teams in heap. pull out teams 2 at a time for relatively even pairings.

2. You are creating a program to manage partner assignments for programming projects in 373, however this quarter it has been decided that you must choose a new partner for each project. A Student object stores a collection of that student's past partners. Your job is to take in a collection of all enrolled students and create pairings such that each student receives a partner they have not worked with in the past.

(i) Which ADT is the best fit for the collection of enrolled students?

SET

(ii) Which ADT is the best fit for the internal collection of past partners in the Student object?

SET

(iii) Describe how you will apply your chosen ADTs to this scenario.

1 set of all students yet to be partnered. Each student has set of previous partners.

(iv) Which data structure would you use to implement the ADT for the enrolled student collection?

Hash Table -

(v) What about this data structure is uniquely suited for this scenario?

allow for traversals and quick removals as you pair students

(vi) Describe the algorithm your code would implement to move through the collection of enrolled students and update the Student objects with a new partner, ensuring each Student is paired with someone they haven't worked with before.

loop over set of enrolled students, pull out 2. Check their inner sets to make sure they haven't partnered. If clear - delete both from enrolled set + update internal sets. If not allowed, pull until valid pairing.

3. You have been tasked with creating a software version of a popular ice breaker - Rock Paper Scissors Battle (RPSB). RPSB is played by a large group of people who initially break into pairs and play a single game of Rock Paper Scissors. The losers of that initial game then attach themselves to the winner and become their cheerleader. The winners of that initial round then find another winner (accompanied by their own cheerleader) and the two battle. The loser of that round and their cheerleader then become cheerleaders of the second-round champion. This continues as the groups grow larger and larger until the final showdown between the final two champions, and the rest of the original players, all of whom have become cheerleaders for one of the final players.

(i) Which ADT is the best fit to organize the champions and their cheerleaders in the tournament?

DISJOINT SET

(ii) Describe how you will apply your chosen ADT to this scenario?

each student/player starts as own set. Loser is unioned into winner representing "champion" and "cheerleaders".

(iii) Which data structure would you use to implement your chosen ADT?

ARRAY

(iv) Describe the algorithm your code would implement to build the teams of cheerleaders based on the results of a given round of Rock Paper Scissors

after 2 champions compete, loser is unioned into winner, leaving winner representitive of their set and loser + their cheerleaders all become cheerleaders of winner.

(B) Assessing Designs

Google is evaluating possible designs for a new CAPTCHA system with two goals, first to check whether users are human (i.e. not robots) and second to improve the categorization of images.

- **Phase 1:** The system will present users with a set of images and ask them to select all that contain a specific item (e.g., "select all images that contain cars"). Users who do not correctly identify the images are deemed "robots". Users who pass the first prompt are deemed "human" and move onto the second prompt.
- **Phase 2:** Human uses are shown a new prompt of the same type in phase 1, but they are asked to identify an item for which those images are not currently tagged. For the images they select that item is added to the set of items known to be contained within that image. An image may contain any number of items.

Here are five possible designs for how to organize the data, some of which make more sense than others. Evaluate the viability and optimality of each design by answering the questions afterwards.

Design A: AVL dictionary - keys: item types - values: array sets of images that contain that item type	**Design D:** adjacency-list graph - vertices: item types - edges: images; each image will have multiple edges connecting the item types it contains
Design B: hash dictionary - keys: images - values: array lists of item types found in that image	**Design E:** adjacency-matrix graph - rows of matrix: images - columns of matrix: item types - values of matrix: whether the image for that row contains the item for that column
Design C: hash dictionary - keys: item types - values: array lists of images containing that item type	**Design F:** an adjacency-list graph - vertices: both items and images - edges: directed edges leaving the vertex of the item type and ending on the vertex of the image

List the letter of each design that satisfies the following.
For "efficient" only consider runtimes of O(nlogn) or faster.

1. Which of the designs would facilitate efficient validation of phase 1 responses if n represents the count of images containing the given item type? *validate that the user chose all images for the given item type*	E (ACCEPT ALSO C, D, F)
2. Which of the designs would facilitate efficient updating of item types contained in an image based on phase 2 responses where n represents the set of images that need to be updated? *update the set of images associated with a given item type based on user responses*	B C (ACCEPT ALSO E)
3. Which of the designs would allow for efficient additions of new images for pre-existing item types where n represents the number of images to be added? *add new images to data set that contain item types already represented in data*	B C
4. Which of the designs would allow for efficient additions of new item types for pre-existing images where n represents the number of item types to be added? *add in new set of item types to be categorized by phase 2 responses at a later time*	B C D F

. Multiple Choice

For each of the following questions, choose the single answer that fits best. **Clearly circle your answer.**

1. If the overall root of a binary search tree is removed, where in the tree would one find the value that could be swapped in as a replacement maintain the binary search property?
   (A) All the way to the left of the node to be deleted
   (B) All the way to the right of the node to be deleted
   [✓ **(C)**] One to the left and then all the way to the right of the node to be deleted
   (D) The deleted node's left child

2. Topological sort is used to...
   (A) Determine the number of cycles in a graph
   [✓ **(B)**] Create a linear ordering of vertices
   (C) Determine if a graph is a DAG
   (D) Create a simplified meta-graph to reduce runtime of complex scenarios

3. Strongly connected components are...
   (A) The sub-sets contained within a disjoint set
   (B) The portions of a graph that are connected to one another
   [✓ **(C)**] Sub-components of a graph in which each pair of vertices in the sub-component are connected via some path in both directions
   (D) A method of reducing a graph to a form that would enable the application of topological sort

4. A given algorithm is in the NP Complete complexity class if...
   (A) It is in the NP complexity class
   (B) It can be reduced to 3 SAT
   (C) Its solution can be verified in polynomial time
   [✓ **(D)**] All of the above

5. Imagine you are given the same exact data set stored in both an ArrayList and a LinkedList. You perform a linear search on both looking through each entry looking for the same specific value. When timed, the ArrayList implementation finishes more quickly. What might be an explanation for this despite both implementations having the same tight Big O runtime?
   (A) In Java, ArrayLists run faster than LinkedLists
   [✓ **(B)**] The ArrayList implementation benefits from caching
   (C) The LinkedList implementation requires more lines of code and thus runs longer
   (D) In Java, ArrayLists leverage hashing for more optimal look up time

6. Which of the following are valid reasons to use selection sort instead of insertion sort?
   (A) selection sort has the same big-O in the best and worst cases, so it is a stable sort
   (B) selection sort is more efficient for almost-sorted arrays
   [✓ **(C)**] selection sort requires fewer array write operations, so it could be faster when write operations are slow (e.g., writing directly to a hard disk)
   (D) selection sort is in-place, unlike insertion sort, so it uses less memory

[illegible] Extra Credit (1 point)

Draw an artistic representation of your TA

(extra work space provided)