Master of Molecular Science and Software Engineering
Problem Set #2
CHEM 247B – Software Engineering Fundamentals for Molecular Science
DUE Date: Wednesday, October 30th, 2024 by 11:59 PM Pacific
Individual Assignment

The main goal of this assignment is to provide you with an opportunity to practice some of the concepts covered in weeks 4 through 7 of this course. Each problem has a set of questions and programming tasks. You will submit a PDF file with your answers. Make sure to identify the questions that you are answering (e.g., "Problem 1.2.1"). Written answers do not need to be full paragraph form, i.e., bullet-point format is permissible. You can use your favorite text editor (e.g., Goodle Doc, MS word, plain text editor, Latex, etc) but submit only the corresponding PDF file by uploading it to BCourses. Name your PDF file with the following format: <Berkeley ID>-Answers-Assignment2.pdf

Document your programs well (e.g., using meaningful names for variables and functions, file names, relevant documentation). Make sure your computer programs are readable (i.e., exercise writing computer programs with code readability in mind). If a problem requests that you provide any programming files, place them all within a zipped folder containing all requested scripts for this problem assignment. Title the folder with the following format: <Berkeley ID>-Answers-Assignment2-code and upload it to BCourses. If you use external code, e.g., StackOverflow (LLM outputs not allowed), please cite the source and provide rationale to prove you understand the code and that it is correct.

The following questions will require Gradescope submissions with similar instructions to those from problem set 1: 2, 4, 5 (Bonus), 16, 19 (Bonus)

Assignment Points: 100
Additional Bonus Points Possible: 22

Priority Queues & Trees

1. Binary Heaps & Binary Trees

1.1 [1 point] Consider the following list of integers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Show the binary min-heap resulting from inserting the integers one at a time.

1.2 [1 point] Consider the following list of integers: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]. Show the binary min-heap resulting from inserting the integers one at a time.

1.3 [1 point] Consider the following list of integers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Show the binary search tree resulting from inserting the integers in the list.

1.4 [1 point] Consider the following list of integers: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]. Show the binary search tree resulting from inserting the integers in the list.

1.5 [1 point] Consider the following list of integers: [68, 88, 61, 89, 94, 50, 4, 76, 66, 82]. Show the binary search tree resulting from inserting the integers in the list.

2. Heap of Challenges: MaxHeap Mastery

2.1 [4 points] Python heapq class provides a Min heap implementation. Using the starter code in maxheap.py as a basis, implement a Max heap class.

2.2 [3 points] Using the provided extract_max method for help, write a sorting function that can sort a list in O(n log n) time. The heap_sort_descending function should use your MaxHeap class to sort the list in descending order.

2.3 [4 points] Create a binary heap with a limited heap size.

In other words, the heap keeps track of the items that are of highest cost. If the heap grows in size to more than allowable items, the highest cost item is dropped.

3. I Dream of Binary Trees & Heaps: Provide a description of how you would solve each problem. The description can be in natural language, but you are free to provide additional information (diagrams, pseudocode/code, etc) if you want to. Explain how your algorithm works and analyze its time complexity.

3.1 [2 points] Dynamic median. Design a data type that supports *insert* in logarithmic time, *find-the-median* in constant time, and *remove-the-median* in logarithmic time. If the number of keys in the data type is even, find/remove the *lower median*. Note that, by dynamic median, we mean that there is a *stream* of data being input. We do not have all of the data up-front / in bulk.

3.2 [2 points] Check if a binary tree is a BST. Given a binary tree where each Node contains a key, determine whether it is a binary search tree. Use extra space proportional to the height of the tree.

3.3 [1 point] Red–black BST with no extra memory. Describe how to save the memory for storing the color information when implementing a red–black BST. *Hint*: modify the structure of the BST to encode the color information.

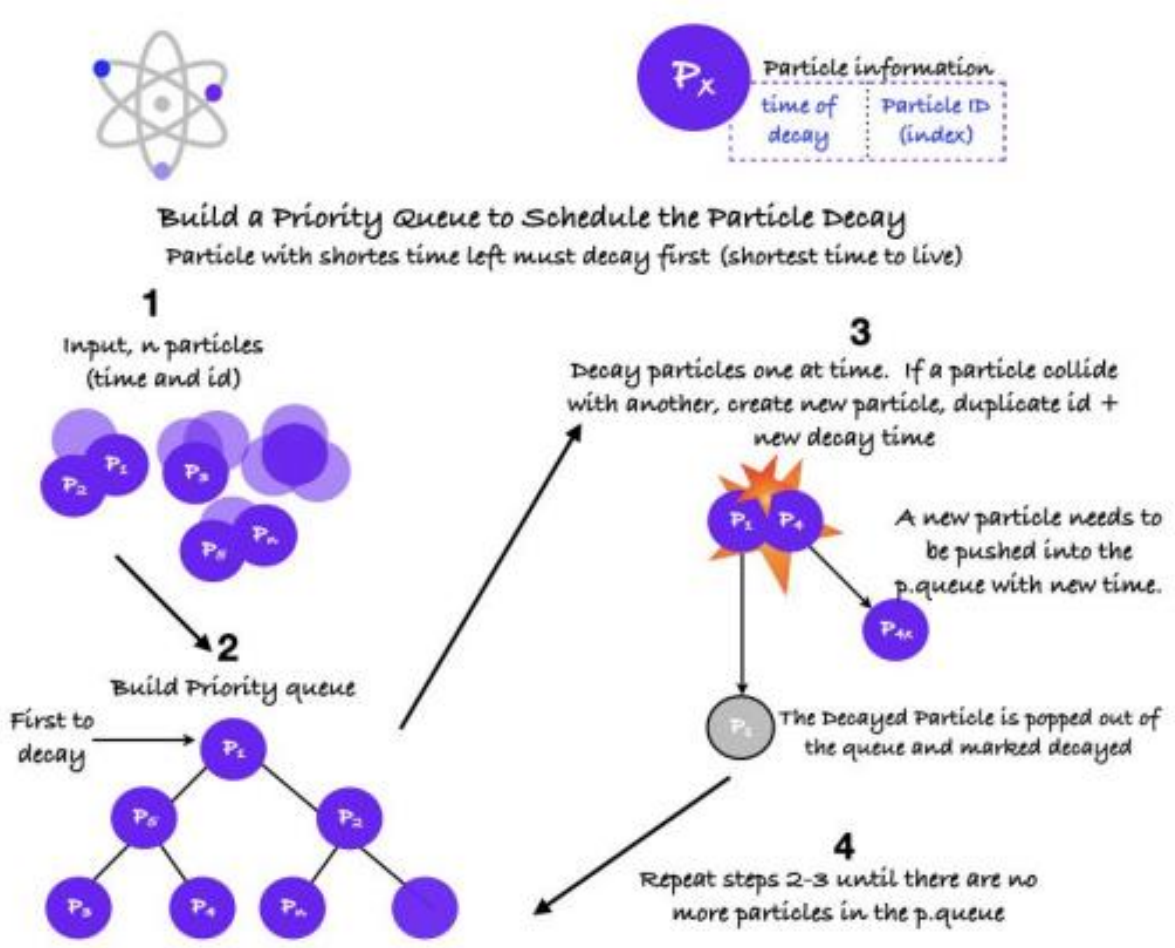
4. Trie Harder: Boggle is a word game played on an 44-by-44 grid of tiles, where each tile contains one letter in the alphabet. The goal is to find all words in the dictionary that can be made by following a path of adjacent tiles (with no tile repeated), where two tiles are adjacent if they are horizontal, vertical, or diagonal neighbors.

In the boggle.py starter code, we've implemented the TrieNode class, part of the Trie class, the BoggleBoard class, and part of the BoggleSolver class.

4.1 Trie Search Methods [4 points] Complete the implementation for the remaining two methods in the Trie class.

4.2 Solving Boggle [3 points] Complete the DFS functionality of the BoggleSolver class.

5. [BONUS] Particle Decay Simulator: This question tests your understanding of priority queues in the context of a particle decay simulator, which will function as illustrated in the below figure. This question walks through the implementation of several helper functions in the starter code file particle_simulator.py. Once these are implemented, the function, simulate_decay, which combines all of your previous implementations, will simulate the particle decay events that will occur given a list of particles.



5.1 [1 BONUS point] To get started, implement a Particle class where each particle will have an ID and a decay time. Implement the methods for the Particle class that are referenced in the starter code file for this problem.

5.2 [2 BONUS points] Implement the ParticleHeap class using Python's heapq module.

5.3 [1 BONUS point] Implement the function, generate_particles, that will generate a list of particles with random decay times.

5.4 [2 BONUS points] Implement the function, simulate_collision, to simulate the particle collisions.

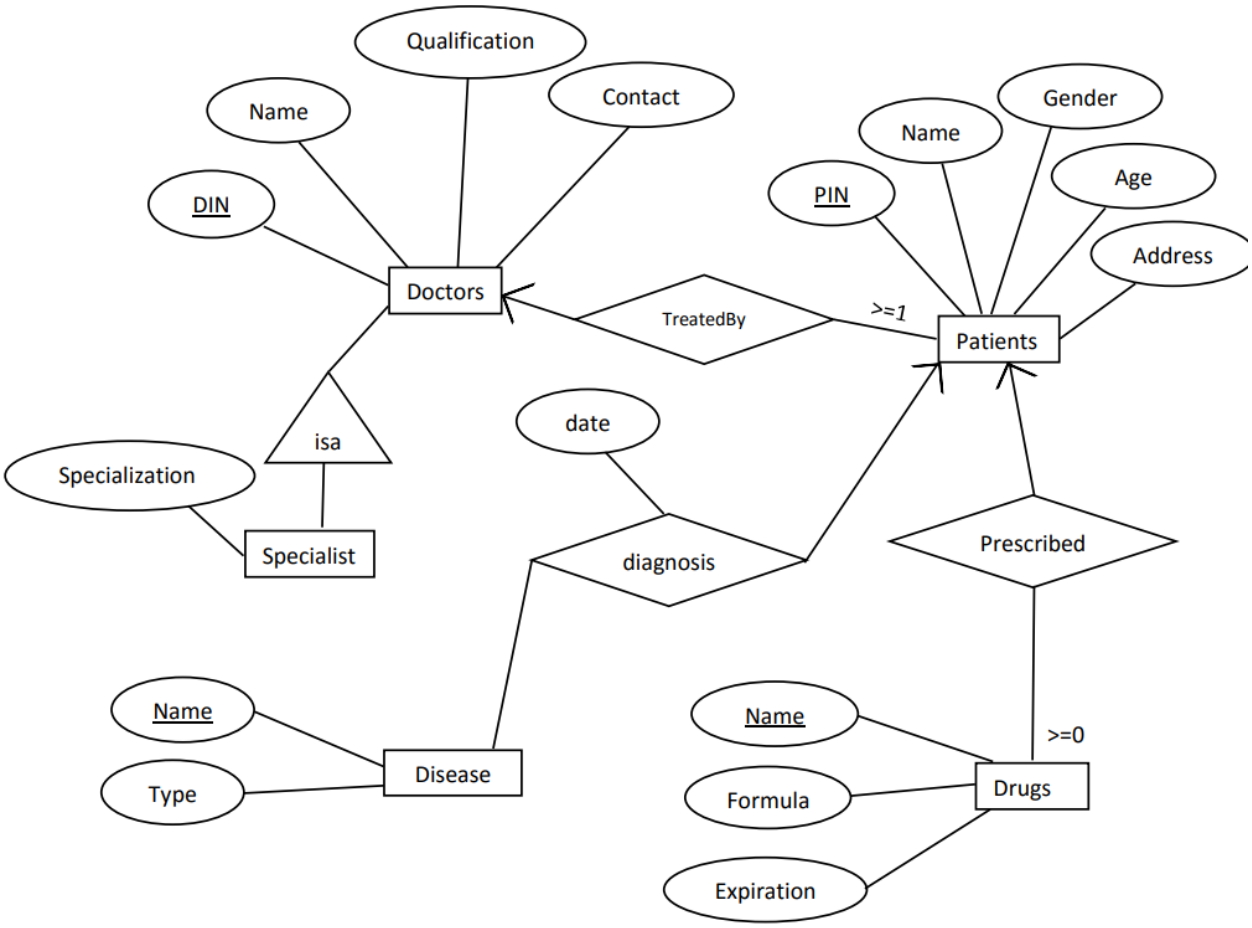
Databases

6 E/R Diagram Practice

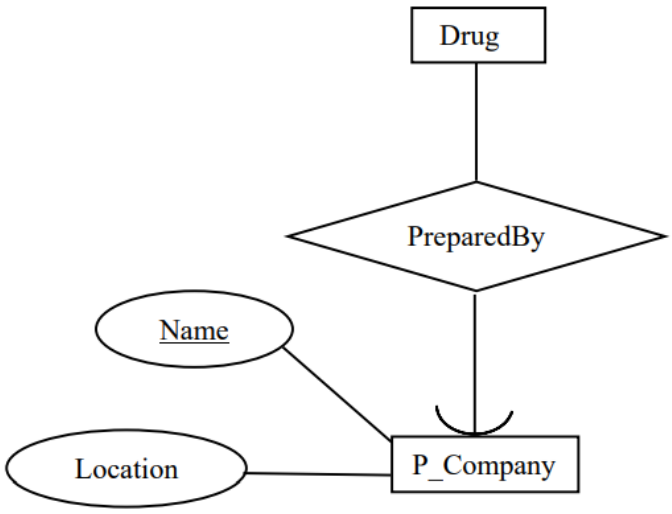
6.1 [3 points] Assume that you have been hired by Wexner Medical Center as a database consultant and you have to create and manage the patient record database system at the Center. Your first assigned task is to draw the database Entity Relationship (E/R) diagram based on the following information.

1. The medical center has doctors, each of whom is assigned a unique doctor identification number, 'DIN'. For each doctor, the name, qualification and contact information must be recorded. In addition, some of these doctors are specialists and you have to record their specialization. (You do not have to worry yet about what pieces of information goes into each of these recorded attributes, and may assume each is a string.) Each doctor treats one or more patients.
2. Every patient is assigned a unique patient identification number 'PIN', and must also have their name, gender, age and address information associated with their record. Each patient is treated by at most one doctor.
3. Each patient may have been diagnosed with one or more diseases, but not every patient has a diagnosed disease. If a patient does have a diagnosis, each diagnosis has a date associated with it. A disease is characterized by a unique name and a type.
4. Each patient may have been prescribed with drugs. Not every patient has to have prescriptions, and a patient may have been prescribed more than one drug. Each drug has a name, formula, and expiration date.

Is the following E/R diagram sufficient for illustrating the described database? Does it make any unstated assumptions? If so, please state them clearly. Are there any criteria it violates or doesn't meet? Would you make any modifications? If so, describe one.



6.2 [1.5 points; 0.5 point each] Let's assume that the ER diagram in Question 2 is extended to introduce a new entity set called Pharmaceutical Company (or 'P_Company' in short). The (relevant portion of this) new E/R diagram is shown below.

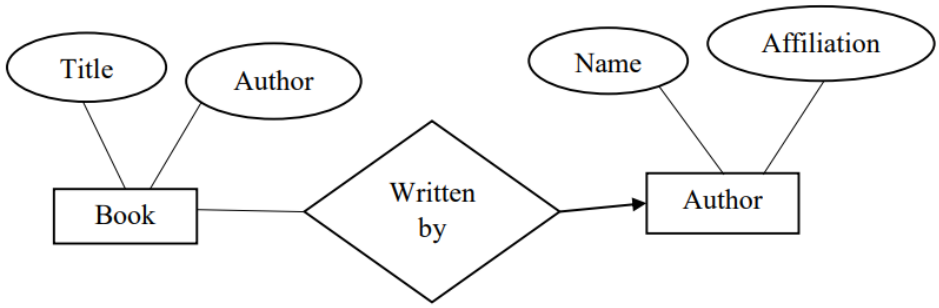


6.2.1 From the statements following the figure choose the best option in light of this E/R diagram.

a) Each drug is prepared by one company, and each company prepares one drug.
b) Each drug is prepared by several pharmaceutical companies.
c) Each drug is prepared by exactly one pharmaceutical company.
d) Each company prepares exactly one drug.

6.2.2 Referring back to 6.1, would "qualification" be a good, uniquely identifying key for doctors? Explain your answer briefly. Assume qualification is their role or specialty, e.g., "neurosurgeon"

6.2.3 Is the following a good or bad design? Explain briefly. (Assume that the designer was informed that a book will have at most one author.)



7 From Design to Code

7.1 [2 points] Write the SQL commands that would define each table below.

a) Professors
b) Students
c) Advise

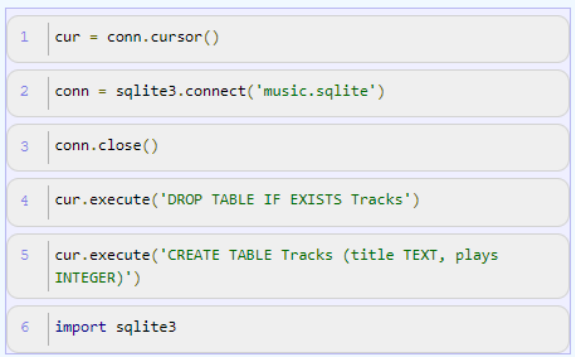
(Your definition must include appropriate data types for each field, as well as any key declarations. Do not worry about the lengths of various items of information; just choose some reasonable length, e.g., INT(20) for ints, FLOAT(3) for floats, VARCHAR(40) for strings)

7.2 [1 point] Delete the "phone-number" field in the "Professors" table. Then, in a separate command, add the "email-id" field, having a default value of "academic@berkeley.edu".

Hint: To delete the "phone-number", look into the ALTER TABLE and DROP keywords. To add the "email-id" field with default value, look into the ALTER TABLE, ADD, VARHAR(40), and DEFAULT keywords. This is one way to do it but not necessarily the only solution.

8 Mixed-Up Code Questions: Code is provided for each question, but it is mixed up! Provide the correct order of the code blocks (using their associated numbers). For some code blocks, you have to choose the correct option. For example, if the option is 1a or 1b, only one of those code blocks will be in the correct code submission. Note that, with respect to this example, the "1" does not signify that the code block is the first line of code.

8.1 [1 point] Create a database file "music.sqlite" and a table "Tracks" with two columns. Let's walk through the answer, to guide how to solve questions 8.2 through 8.7, inclusive. **YOU WILL STILL NEED TO INCLUDE THE ANSWER TO THIS QUESTION IN YOUR SUBMISSION FOR POINTS!!!**

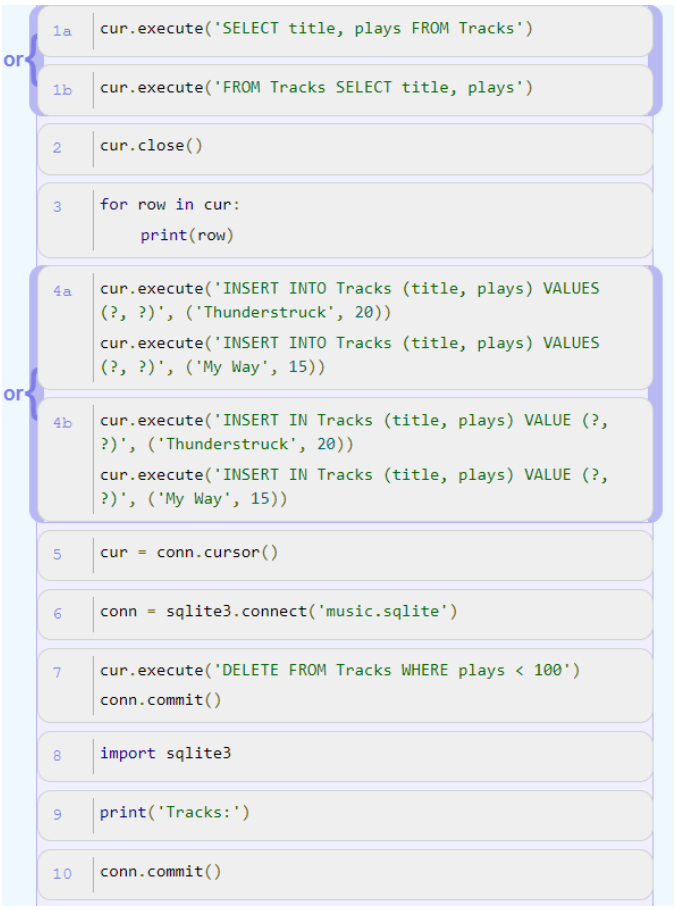


```
1  cur = conn.cursor()
2  conn = sqlite3.connect('music.sqlite')
3  conn.close()
4  cur.execute('DROP TABLE IF EXISTS Tracks')
5  cur.execute('CREATE TABLE Tracks (title TEXT, plays
   INTEGER)')
6  import sqlite3
```

Answer: 6, 2, 1, 4, 5, 3

- Line 6: We first import the sqlite3 library in our Python script
- Line 2: With sqlite3, we can establish connection with the "music.sqlite" file that contains data of interest
- Line 1: We set a cursor object, which implements functions such as "execute()" that we can pass a query into for application on our data base
- Line 4: Remove the table Tracks if it already exists
- Line 5: Create our Tracks table with the specified, two columns format
- Line 3: We close our connection as we don't plan to further modify the database

8.2 [1 point] Create a database file "music.sqlite". Then, insert 2 tracks into the table, commit that change, print the track data, delete a track if it meets a certain condition, and commit that change. Remember to make a connection to the database and create the cursor, first!



8.3 [1 point] Create a database file “spider.sqlite”. Then selects all of the rows in the table “Twitter”. Then loop through the rows and prints out each row. At the end it will print the total count of rows, before closing the cursor.

```
print(row)
count = count + 1
cur = conn.cursor()
conn = sqlite3.connect('spider.sqlite')
cur.close()
cur.execute('SELECT * FROM Twitter')
print(count, 'rows.')
or
print('rows.')
import sqlite3
for row in cur:
count = 1
or
count = 0
```

8.4 [1 point] Create a 'clothes.sqlite' database and select all of the rows in the table "Socks". Then, loop through the rows, print out each row and print out the total number of rows. Then, delete all green socks from the table.

```
import sqlite3
print(f"The table 'Socks' has {count} rows")
cur.close()
conn = sqlite3.connect('clothes.sqlite')
cur = conn.cursor()
for row in cur:
or
cur.execute('SELECT all_rows FROM Socks')
cur.execute('SELECT * FROM Socks')
count = 0
print(row)
count = count + 1
cur.execute('DELETE * FROM Socks WHERE color = "Green"')
conn.commit()
```

8.5 [1 point] Create a database "pets.sqlite". Join tables 'Dogs' and 'Cats', then select all rows on column 'name' for both tables, where the name is 'Spot'. At the end it will print the total count of rows, before closing the cursor.

```
1   import sqlite3
2   conn = sqlite3.connect('pets.sqlite')
    cur = conn.cursor()
3a  cur.execute('''SELECT * FROM Dogs JOIN Cats ON Dogs.name
    = Cats.name WHERE Dogs.name = "Spot"''')
or
3b  cur.execute('''SELECT * FROM Dogs, Cats ON Dogs.name ==
    Cats.name WHERE Dogs.name = "Spot"''')
4   cur.close()
```

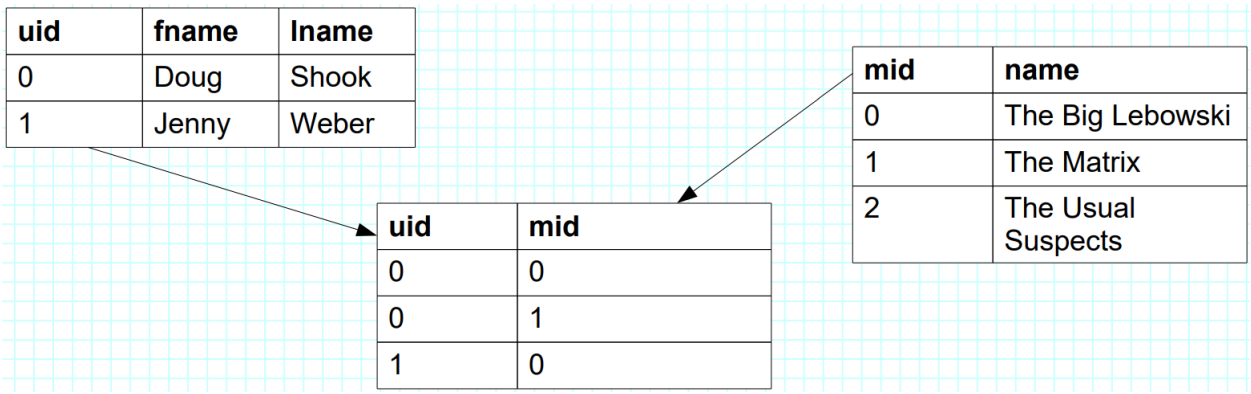
8.6 [1 point] Create a database “clothes.sqlite”. Simply open the database and join tables pants and shirts and select all pants and shirts with the same fabric, then print them.

```
1a  cur.execute('''SELECT * FROM Pants JOIN Shirts
    ON Pants.fabric == Shirts.fabric''')
or
1b  cur.execute('''SELECT * FROM Pants JOIN Shirts
            ON Pants.fabric = Shirts.fabric''')
2   conn = sqlite3.connect('clothes.sqlite')
3a  cur = conn.cursor
or
3b  cur = conn.cursor()
4   import sqlite3
5   cur.close()
6a  for row in conn:
or
6b  for row in cur:
7   print(row)
```

8.7 [1 point] Create a database ‘desserts.sqlite’. Simply open the database and select all of the Cupcakes and Cakes where the Cupcake icing and Cake frosting are chocolate. Then, print the rows.

```
1  print(row)

2  import sqlite3

3a cur.execute('''SELECT * FROM Cupcakes JOIN Cakes
                  ON Cupcakes.icing = Cakes.frosting
                  WHERE Cupcakes.icing = "chocolate"''')
or
3b cur.execute('''SELECT * FROM Cupcakes JOIN Cakes
                     ON icing.Cake = frosting.Cupcake
                  WHERE icing.Cupcakes = "chocolate"''')

4  conn = sqlite3.connect('desserts.sqlite')

5  cur.close()

6a cur = cursor()
or
6b cur = conn.cursor()

7  for row in cur:
```

9 SQL Coding Without Training Wheels: Given the following tables, write queries to answer the following questions. <u>You do not need to provide any additional code besides the queries themselves.</u>



uid	fname	lname
0	Doug	Shook
1	Jenny	Weber

mid	name
0	The Big Lebowski
1	The Matrix
2	The Usual Suspects

uid	mid
0	0
0	1
1	0

9.1 [1 point] What is the name of each user?

9.2 [1 point] How many movies have been watched by all users?

9.3 [1 point] How many movies has each user watched?

9.4 [2 points] How many times has each movie been watched?

10 SQL is Too Much Fun! Consider the following schema of a movie database which currently stores information about actors and their movies. Assume that the 'genre' attribute in movies can take the values 'Action', 'Thriller', 'Drama', 'Romance', 'Comedy'.

actors (<u>actorid</u>, name, gender, age, nationality)

actors2movies (<u>actorid</u>, <u>movieid</u>)

movies (<u>movieid</u>, title, year, genre, length, country)

10.1 [1.5 points; 0.5 points each] Single relation queries: write the query for each of the following:

10.1.1 Find the names of actors who were born in USA

10.1.2 Find the titles of action movies that were released in 2010

10.1.3 Find all movie titles that begin with "The"

10.2 [4 points; 1 point each] Queries involving more than one relation

10.2.1 Find the name of the oldest actor who worked in a thriller movie. (In the case of a tie, return the names of all the actors involved in the tie.)

10.2.2 Find the movie id, year and names of actors for the movie "Maze Runner: The Scorch Trials"

10.2.3 Write a SQL query that returns pairs of movies that have the same length. Make sure that your results should not return duplicate names for the same movies. You are required to return two columns only named as title 1 and title 2.

10.2.4 Find the names of the actors who have worked in more than 10 movies. Your results should return the names of such actors and number of movies each has acted in.

11 Constraints, Transactions, Schedules

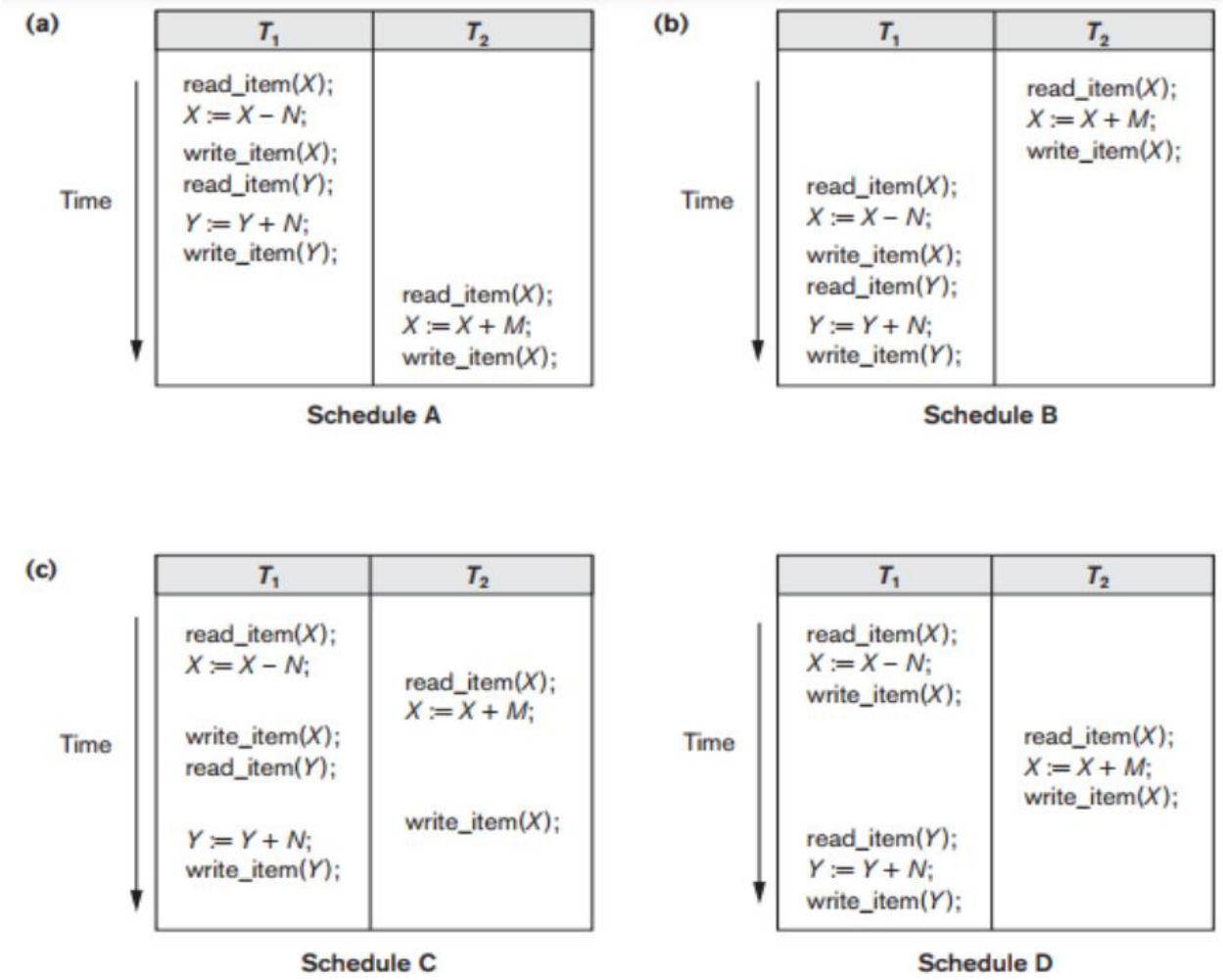
11.1 [2 BONUS points; 0.5 each] Constraints. How is it possible to violate the following constraints in the context of Insert, Delete, and/or Update modification operations?

- Domain constraints
- Key constraints
- Referential integrity
- Entity integrity

11.2 [1.5 points; 0.5 points each] Transactions. Why do we need transactions? What does a transaction contain? What are the four properties of a transaction?

11.3 [1.5 points; 0.5 points each] What is a schedule? What is a conflict? What does it mean for a schedule to be serializable?

11.4 [2 points; 0.5 points each] Run the conflict serializability test for each schedule below. Which schedules are serializable? Provide justification on how you reached your answer.



12 Concurrency

12.1 [1 point] Does strict two phase locking prevent deadlock? If yes, explain how. If no, provide an example of a schedule that obeys two phase locking but also creates deadlock.

12.2 [1 point] Will basic two phase locking lead to better performance than strict two phase locking (in general)?

12.3 [2 points; 1 point each] For the following schedules: what conflicts exist? Is the schedule serializable?

12.3.1 R1(X), R2(X), W1(Y), W1(Z), R2(Z), R1(Y), R2(Y)

12.3.2 R1(Z), W2(Y), R3(Y), W2(Z), R1(Y), W1(X), R3(Z), W1(Y), W3(Z)

13 Distributed Systems

13.1 [1 point] Name one advantage and one disadvantage of replication.

13.2 [1 point] Under what circumstances would it be preferable to use a distributed database as opposed to a centralized database?

13.3 [1 point] Describe (not just name) three different challenges in distributed data systems.

13.4 [1 point] When replication is used, the database must decide which replica to use for a given query. What factors will the database use to make this decision?

13.5 [1 point] Assuming that we are not using any replication, which partitioning scheme will take up more space? Vertical fragmentation or horizontal fragmentation? Why?

14 Database Security

14.1 [1 point] Discuss the types of privileges at the account level and those at the relation level.

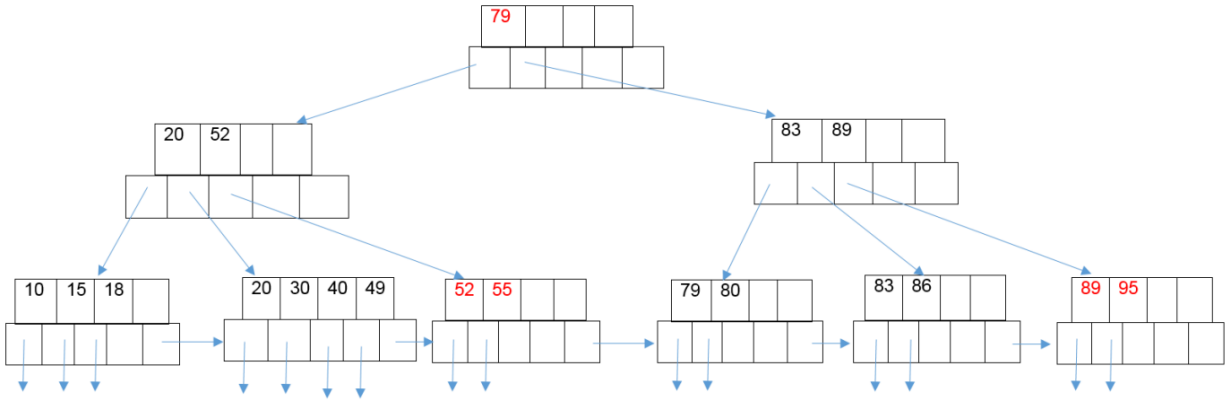
14.2 [1 point] What risks are associated with SQL injection attacks?

14.3 [1 point] What preventative measures are possible against SQL attacks?

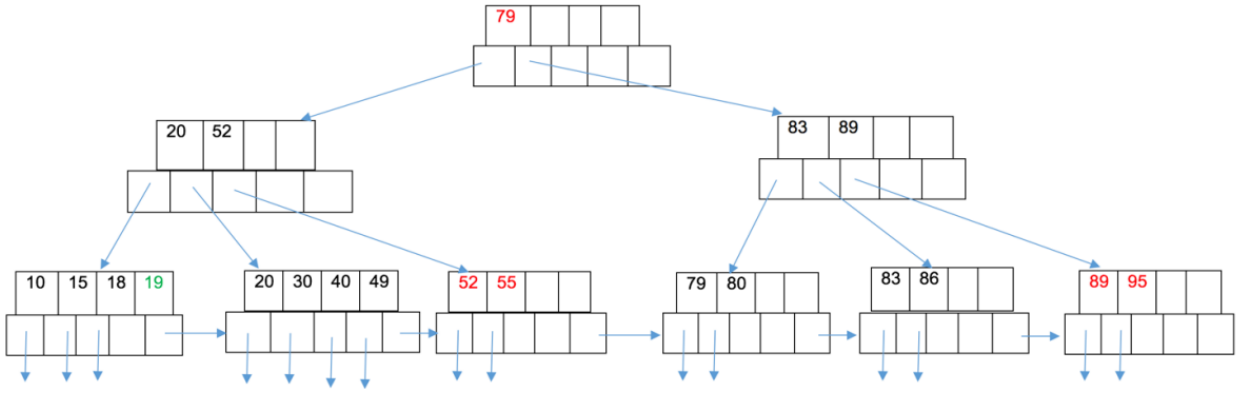
14.4 [1 point] What is the public key infrastructure scheme? How does it provide security?

14.5 [1 point] What are some of the sacrifices that we make for the sake of securing our data?

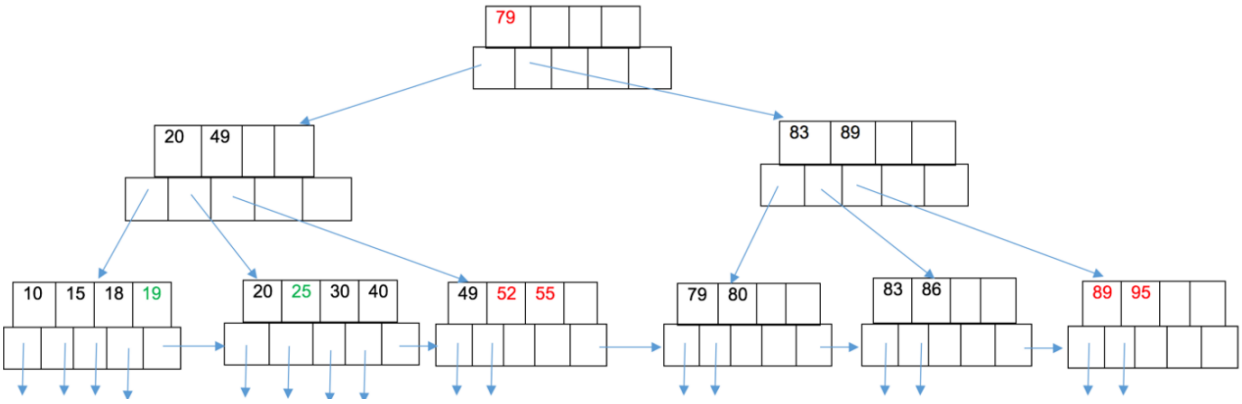
15 B+ Trees: Assume d = 2 and consider the state of the following B+ Tree.

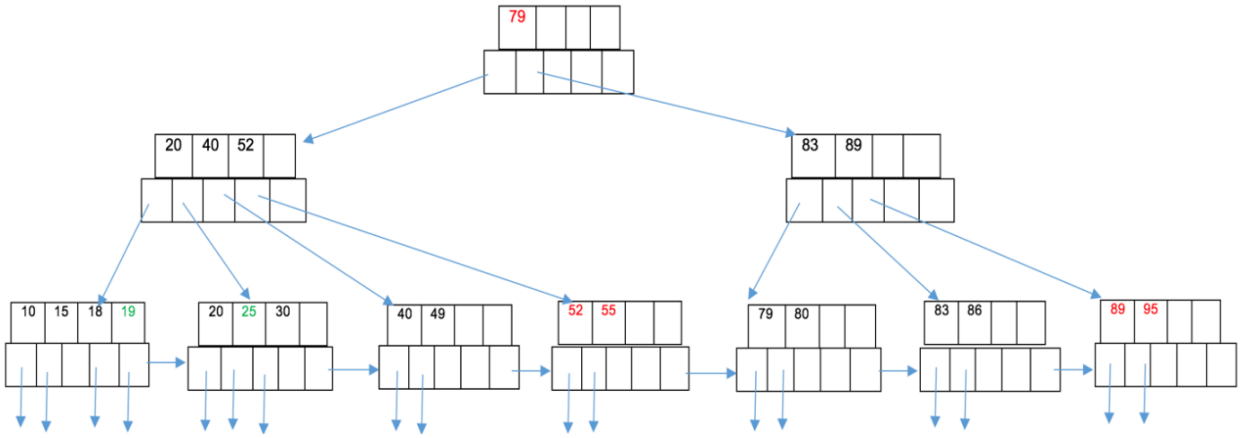


15.1 [Bonus 1 point] The tree will look like the following if we insert 19. Explain why.

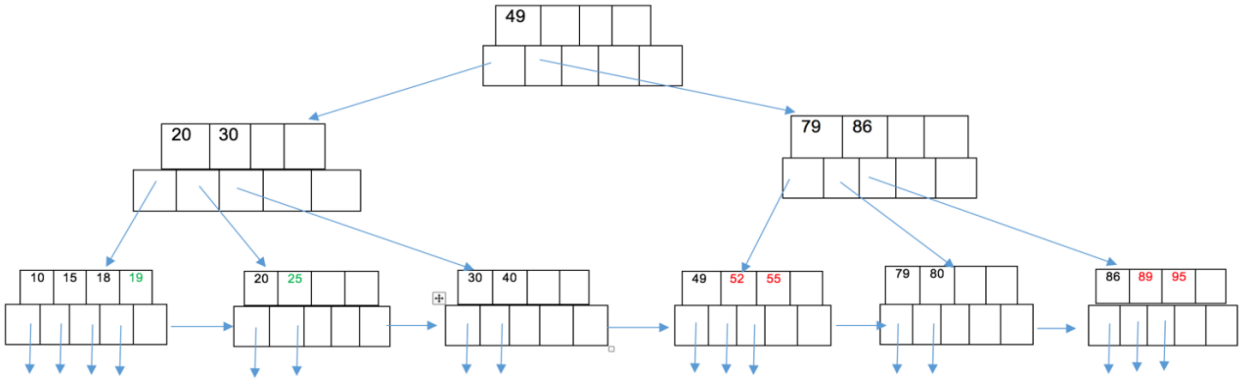


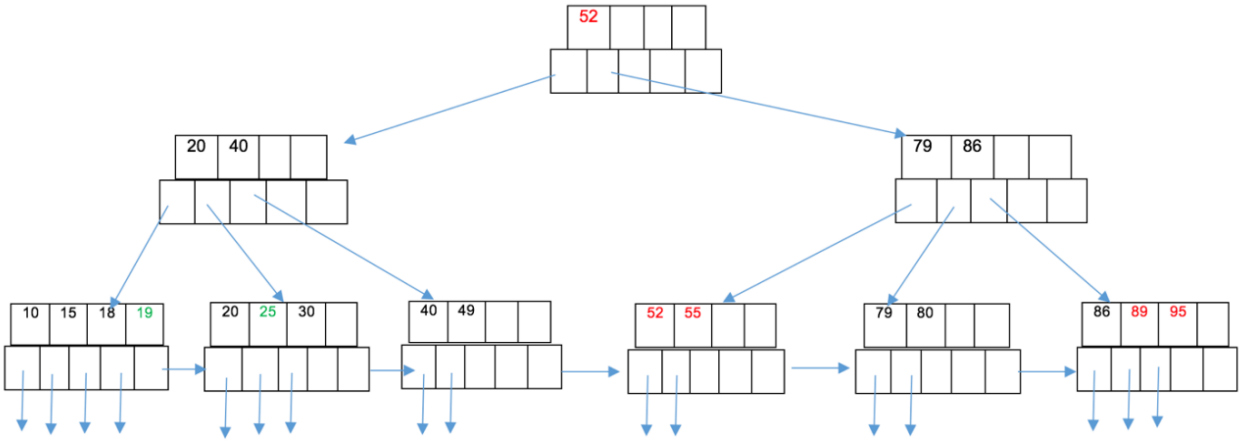
15.2 [Bonus 1 point] The tree could look like one of the following if we insert 25. Explain why for one of the possible outcomes.



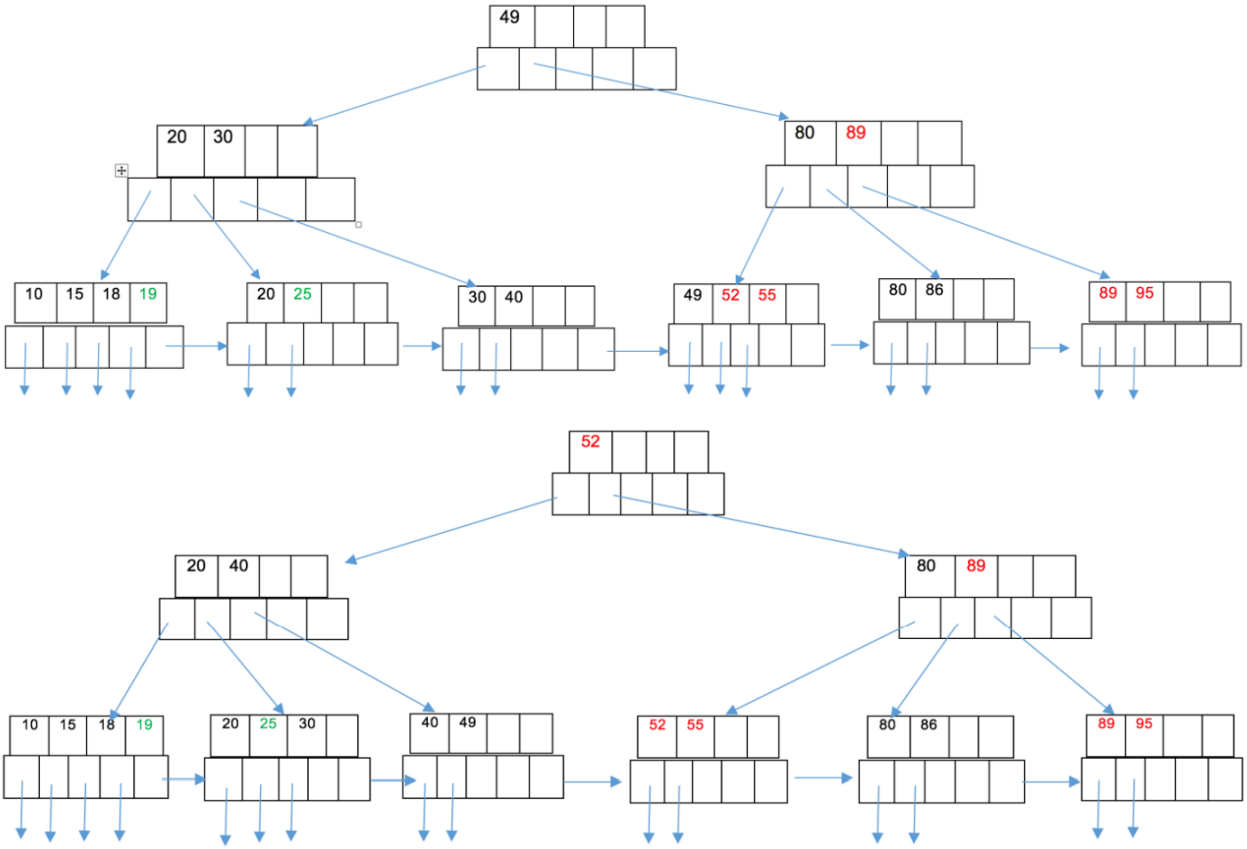


15.3 [Bonus 1 point] Why will the tree look like one of the following if we now delete 83?





15.4 [Bonus 1 point] Why will the tree look like one of the following if we now delete 79?



15.5 [Bonus 1 point] Suppose for a B+ tree the key size is 1 byte. The pointer size is 2 byte. The block size is 20 bytes. Find out the maximum possible value of d.

16. LRU Cache: Caching is a crucial technique in software engineering applications, including database systems. Caching improves performance by storing frequently accessed data in a faster, more easily accessible location. One popular caching strategy is the Least Recently Used (LRU) cache, which evicts the least recently used items when the cache reaches its capacity.

In database systems, we cache query results, or frequently accessed data, to reduce the storage load and improve response times. The starter code file, lru_cache.py, implements a basic LRU cache that meets the following requirements:

1. The cache has a fixed capacity.
2. It supports two operations:
    a. get(key): Get the value associated with the key if it exists in the cache, otherwise return -1.
    b. put(key, value): Insert a key-value pair into the cache. If the cache is at capacity, remove the least recently used item before inserting the new item.
3. Both get and put operations should have an average time complexity of O(1).

Note: For both of the following questions, a test_lru_cache.py file is provided in the starter code for more clarity on expected inputs and outputs.

16.1 [4 points] In real-world database systems, cached data often becomes stale and needs to be refreshed or removed after a certain period. Extend the basic LRU cache to include time-based expiration:

1. The cache should have a fixed capacity and a default expiration time for entries.
2. Modify the get and put operations to handle expiration:
    a. get(key): Return -1 if the key doesn't exist or has expired. Update the item's timestamp if it's accessed and not expired.
    b. put(key, value): Insert or update a key-value pair with the current timestamp.
3. Expired items should be removed when accessed or when space is needed for new items.
4. Maintain O(1) average time complexity for both operations.

16.2 [Bonus 3 points] In database systems, we need to perform certain actions when items are evicted from the cache, such as writing back to disk or updating statistics. Implement an LRU cache with a callback mechanism:

1. The cache should have a fixed capacity and an optional callback function for eviction events.
2. Modify the put operation to call the callback function (if provided) whenever an item is evicted from the cache.
3. The callback function should receive the key and value of the evicted item.
4. Maintain O(1) average time complexity for both get and put operations.

Graphs

17. Representing Graphs

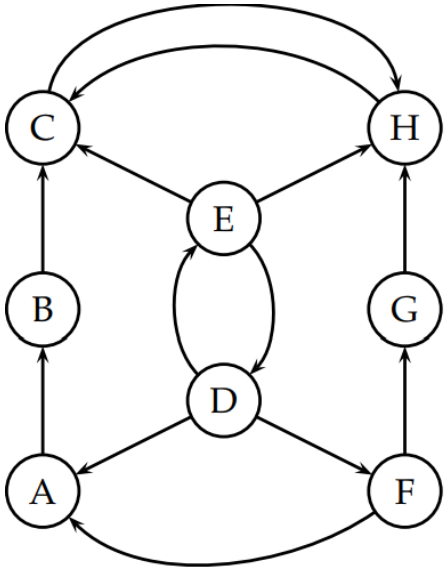
17.1 [1 point] Given an undirected graph $G = (V, E)$, below, answer whether it is connected. If not, how many connected components does it have?

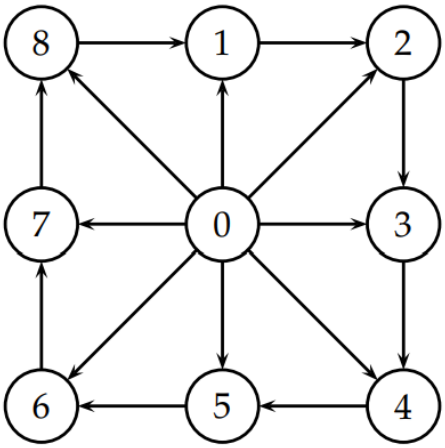
- $V = \{1, 2, 3, 4, 5, 6\}$
- $E = \{\{1, 2\}, \{1, 4\}, \{3, 2\}, \{4, 5\}, \{5, 1\}, \{5, 2\}\}$

17.2 [1 point] Given a directed graph $G = (V, E)$, below, answer whether it is strongly connected or weakly connected. Provide justification.

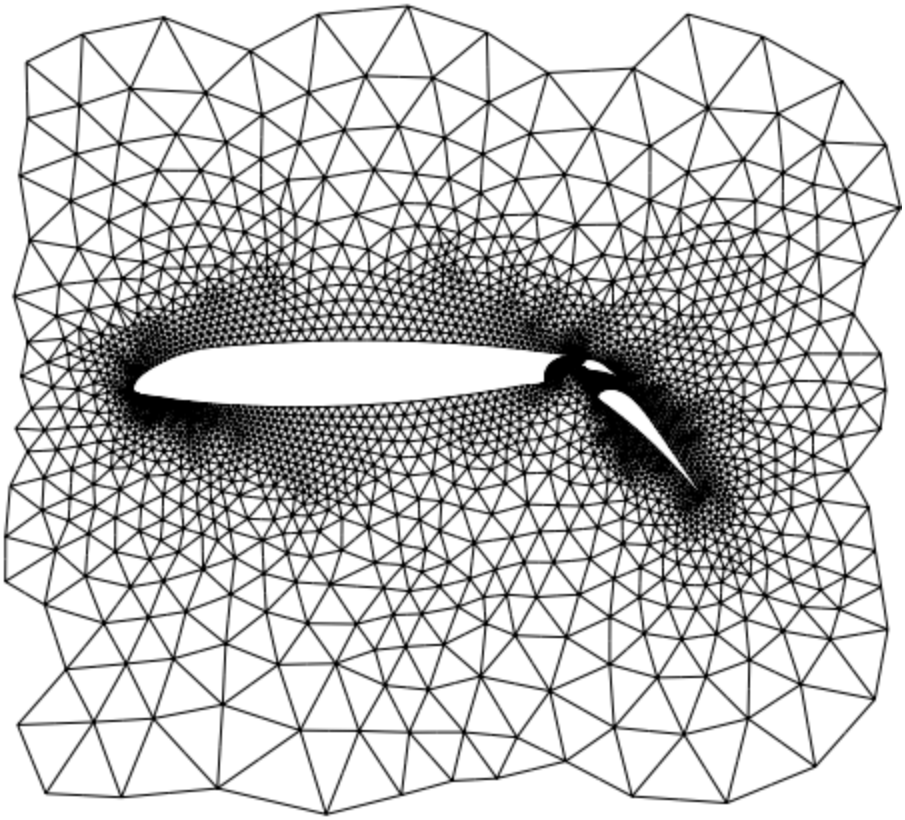
- $V = \{1, 2, 3, 4, 5\}$
- $E = \{(1, 2), (1, 4), (3, 2), (4, 5), (5, 1), (5, 2)\}$

17.3 [4 points] Show how the following directed graphs are represented using an adjacency matrix and adjacency lists. Count the indegree and the outdegree of each vertex. Say whether the graphs are strongly or weakly connected.

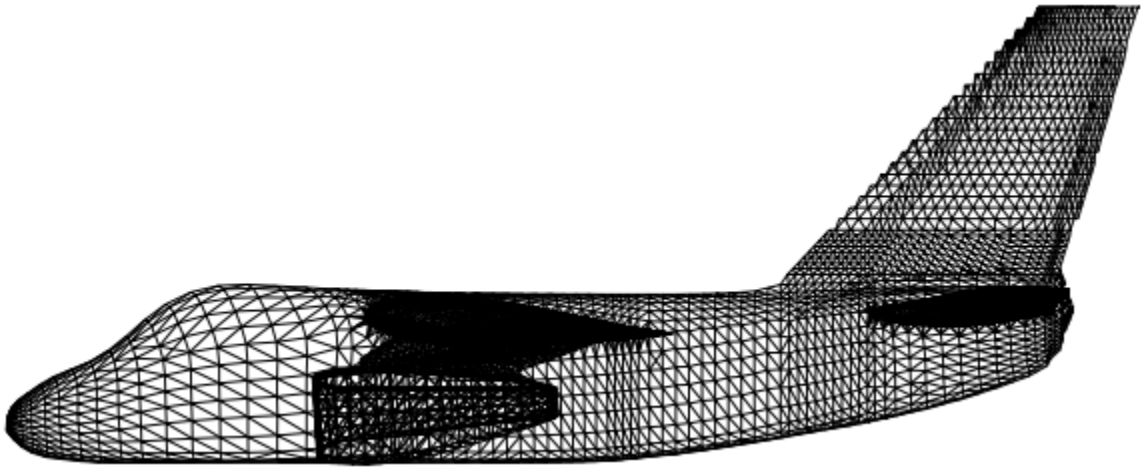




17.4 [2 points] The graph of the following figure has $n = 4253$ vertices and $m = 12289$ edges. Assume that in your computer each boolean is 1 byte, each integer is 4 bytes and each pointer also 4 bytes. Calculate the amount of memory that is necessary to store this graph using an adjacency matrix representation and an adjacency lists representation.

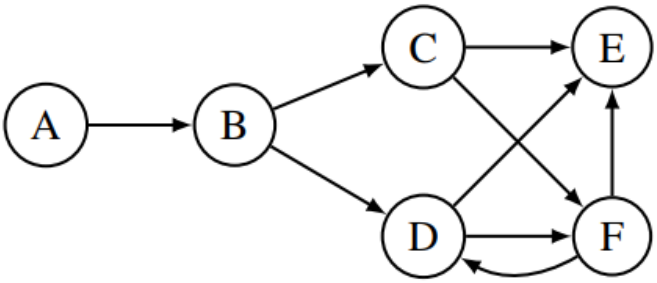


17.5 [1 point] Repeat the previous problem for the following graph, which has $n = 156317$ vertices and $m = 1059331$ edges



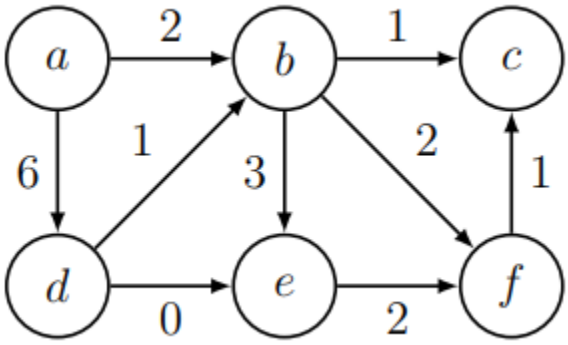
18 Graph Practice

18.1 [2 points] Run both BFS and DFS on the graph below, starting from node A. While performing each search, visit the outgoing neighbors of a vertex in alphabetical order. For each search, draw the resulting tree and list vertices in the order in which they were first visited.



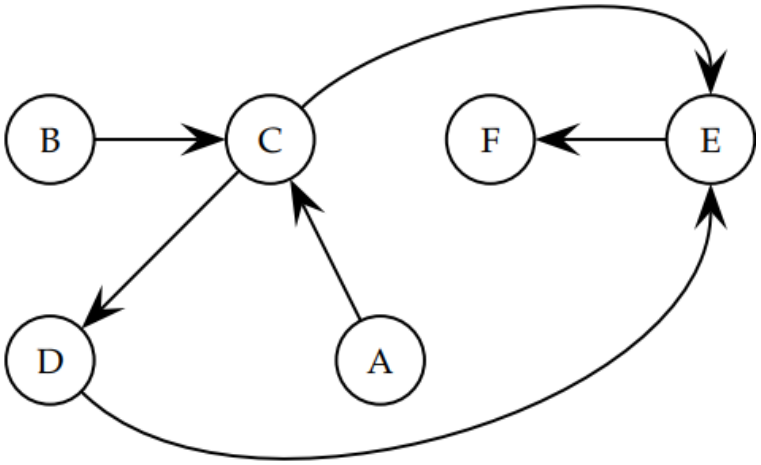
18.2 [2 points] It is possible to remove a single edge from the previous graph to make it a DAG. State every edge with this property, and for each, state a topological sort order of the resulting DAG.

18.3 [2 points] Pivoting to a new graph (below), which is acyclic and has nonnegative edge weights, run Dijkstra on the graph starting from vertex *a*. Write down a list of all edges in relaxation order: the order the algorithm tries to relax them. If there is ambiguity on which vertex or outgoing adjacency to process next, process them in alphabetical order.



18.4 [2 points] Find an interesting graph in the domain of your choosing. Is it directed or undirected? Sparse or dense? Weighted or unweighted? Would any of the graph algorithms we discussed be applicable to problems in this graph's domain?

18.5 [1 point] Topological Sorting. Given are, in lexicographical order, all possible topological orderings of the following directed acyclic graph (DAG).



Topological orderings:

1. A, B, C, D, E, F
2. A, B, C, E, D, F
3. A, B, C, E, F, D
4. B, A, C, D, E, F
5. B, A, C, E, D, F
6. B, A, C, E, F, D
7. B, C, A, D, E, F
8. B, C, A, E, D, F
9. B, C, A, E, F, D
10. B, C, D, A, E, F
11. B, C, D, E, A, F
12. B, C, D, E, F, A
13. B, C, E, A, D, F
14. B, C, E, A, F, D
15. B, C, E, D, A, F
16. B, C, E, D, F, A
17. B, C, E, F, A, D
18. B, C, E, F, D, A

How many topological orderings would the graph now have if we added an isolated vertex? Do not write all of the orderings out, just say how many and provide justification.

19. [Bonus] Knight's Tour: The Knight's Tour is a sequence of moves of a knight on a chessboard such that the knight visits every square exactly once. If the knight ends on a square that is one knight's move from the beginning square (so that it could tour the board again immediately, following the same path), the tour is closed; otherwise, it is open.

You are given an n x n chessboard and a knight piece. The knight can move in an L-shape: two squares vertically and one square horizontally, or two squares horizontally and one square vertically.

19.1 [Bonus 2 points] Explain why the general DFS or BFS algorithms are not suitable for solving the knight's tour problem. Also, are there better algorithms than both BFS and DFS for solving this problem (look it up!)? If so, name one and summarize how it works.

19.2 [Bonus 4 points] Implement a function that finds a sequence of moves for the knight such that it visits every square on the chessboard exactly once. Use starter code file knights_tour.py as the basis of your implementation. Note that, if your solution fails with an error in Gradescope, it is due to memory limits being exceeded for this question.

Input:

- n: The size of the chessboard (n x n)
- start_x: The starting x-coordinate of the knight (0-indexed)
- start_y: The starting y-coordinate of the knight (0-indexed)

Output:

- A list of tuples representing the sequence of positions the knight visits. Each tuple contains two integers (x, y) representing a position on the board.
- If no valid tour is found, return an empty list.

Constraints:

- 5 <= n <= 8
- 0 <= start_x, start_y < n

Example:

- Input: n = 5, start_x = 0, start_y = 0
- Output: [(0, 0), (2, 1), (4, 0), (3, 2), (1, 3), (0, 1), (2, 0), (4, 1), (3, 3), (1, 4), (0, 2), (2, 3), (4, 4), (3, 2), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (0, 4), (2, 2), (4, 1), (3, 3), (1, 2)]

Note: There may be multiple valid solutions. Any valid solution will be accepted.

20. [6 points; 0.75 points per question] Tearing Apart Missionaries & Cannibals: Consider the following problem: three missionaries and three cannibals come to a river and find a boat that holds two people. Everyone must get across the river to continue on the journey. However, if the cannibals ever outnumber the missionaries on either bank, the missionaries will be eaten. Find a series of crossings that will get everyone safely to the other side of the river.

A solution is already provided for you in pset2_starter_code/missionaries_and_cannibals.py. The main components:

1. The State class represents a state of the problem, including the number of missionaries and cannibals on each bank and the boat's position.
2. The get_successors function generates all valid next states from a given state.
3. The solve_missionaries_cannibals function implements the breadth-first search algorithm to find the solution.

4. The print_solution function displays the solution in a readable format.

An example solution produced by this code is:

```
Solution found!
Initial state:
  Left bank: 3M 3C
  Right bank: 0M 0C

Move 1:
  0 missionaries and 2 cannibals move to the right bank.
  Left bank: 3M 1C
  Right bank: 0M 2C

Move 2:
  0 missionaries and 1 cannibals move to the left bank.
  Left bank: 3M 2C
  Right bank: 0M 1C

Move 3:
  0 missionaries and 2 cannibals move to the right bank.
  Left bank: 3M 0C
  Right bank: 0M 3C

Move 4:
  0 missionaries and 1 cannibals move to the left bank.
  Left bank: 3M 1C
  Right bank: 0M 2C

Move 5:
  2 missionaries and 0 cannibals move to the right bank.
  Left bank: 1M 1C
  Right bank: 2M 2C

Move 6:
  1 missionaries and 1 cannibals move to the left bank.
  Left bank: 2M 2C
  Right bank: 1M 1C

Move 7:
  2 missionaries and 0 cannibals move to the right bank.
  Left bank: 0M 2C
  Right bank: 3M 1C

Move 8:
  0 missionaries and 1 cannibals move to the left bank.
  Left bank: 0M 3C
```

```
  Right bank: 3M 0C

Move 9:
  0 missionaries and 2 cannibals move to the right bank.
  Left bank: 0M 1C
  Right bank: 3M 2C

Move 10:
  1 missionaries and 0 cannibals move to the left bank.
  Left bank: 1M 1C
  Right bank: 2M 2C

Move 11:
  1 missionaries and 1 cannibals move to the right bank.
  Left bank: 0M 0C
  Right bank: 3M 3C
Goal state reached!
```

A great way to improve programming skills is to review code from other's projects or repositories. You may have to do this when onboarding onto a new team or organization's pre-existing code bases. The following questions test your ability to read and understand the mechanics of the solution code.

20.1 Why is the State class used in this program, and what information does it encapsulate?

20.2 What is the purpose of the is_valid() method in the State class?

20.3 How does the program represent and handle the boat movements?

20.4 Why does the program use a breadth-first search (BFS) algorithm? What advantage does this provide?

20.5 How does the program avoid exploring the same state multiple times?

20.6 What is the significance of the __eq__ and __hash__ methods in the State class?

20.7 How does the program reconstruct the solution path once the goal state is reached?

20.8 Why does the get_successors() function consider only specific move combinations (e.g., (1,0), (2,0), (0,1), (0,2), (1,1))?