CSE 332: Data Structures and Parallelism

Section 09: Concurrency, Graphs, & P/NP

0. User Profile

You are designing a new social-networking site to take over the world. To handle all the volume you expect, you want to support multiple threads with a fine-grained locking strategy in which each user's profile is protected with a different lock. At the core of your system is this simple class definition:

```
class UserProfile {
    static int id_counter;
    int id; // unique for each account
    int[] friends = new int[9999]; // horrible style
    int numFriends;
    Image[] embarrassingPhotos = new Image[9999];

    UserProfile() { // constructor for new profiles
        id = id_counter++;
        numFriends = 0;
    }

    synchronized void makeFriends(UserProfile newFriend) {
        synchronized(newFriend) {
            if(numFriends == friends.length
            || newFriend.numFriends == newFriend.friends.length)
                throw new TooManyFriendsException();
                friends[numFriends++] = newFriend.id;
                newFriend.friends[newFriend.numFriends++] = id;
        }
    }

    synchronized void removeFriend(UserProfile frenemy) {
        ...
    }
}
```

a) The constructor has a concurrency error. What is it and how would you fix it? A short English answer is enough - no code or details required.

b) The `makeFriends` method has a concurrency error. What is it and how would you fix it? A short English answer is enough no code or details required.

1. Bubble Tea

The `BubbleTea` class manages a bubble tea order assembled by multiple workers. Multiple threads could be accessing the same `BubbleTea` object. Assume the Stack objects are thread-safe, have enough space, and operations on them will not throw an exception.

```java
public class BubbleTea {
    private Stack<String> drink = new Stack<String>();
    private Stack<String> toppings = new Stack<String>();
    private final int maxDrinkAmount = 8;

    // Checks if drink has capacity
    public boolean hasCapacity() {
        return drink.size() < maxDrinkAmount;
    }

    // Adds liquid to drink
    public void addLiquid(String liquid) {
        if (hasCapacity()) {
            if (liquid.equals("Milk")) {
                while (hasCapacity()) {
                    drink.push("Milk");
                }
            } else {
                drink.push(liquid);
            }
        }
    }

    // Adds newTop to list of toppings to add to drink
    public void addTopping(String newTop) {
        if (newTop.equals("Boba") || newTop.equals("Tapioca")) {
             toppings.push("Bubbles");
        } else {
            toppings.push(newTop);
        }
    }
}
```

a) Does the `BubbleTea` class above have (circle all that apply):

a race condition   potential for deadlock   a data race   none of these

If there are any problems, give an example of when those problems could occur. Be specific!

b) Suppose we made the addTopping method synchronized, and changed nothing else in the code. Does this modified BubbleTea class above have (circle all that apply):

a race condition   potential for deadlock   a data race   none of these

If there are any FIXED problems, describe why they are FIXED. If there are any NEW problems, give an example of when those problems could occur. Be specific!

2. Phone Monitor

The PhoneMonitor class tries to help manage how much you use your cell phone each day. Multiple threads can access the same PhoneMonitor object. Remember that synchronized gives you reentrancy.

```java
public class PhoneMonitor {
    private int numMinutes = 0;
    private int numAccesses = 0;
    private int maxMinutes = 200;
    private int maxAccesses = 10;
    private boolean phoneOn = true;
    private Object accessesLock = new Object();
    private Object minutesLock = new Object();

    public void accessPhone(int minutes) {
        if (phoneOn) {
            synchronized (accessesLock) {
                synchronized (minutesLock) {
                    numAccesses++;
                    numMinutes += minutes;
                    checkLimits();
                }
            }
        }
    }

    private void checkLimits() {
        synchronized (minutesLock) {
            synchronized (accessesLock) {
                if (numAccesses >= maxAccesses
                  || numMinutes >= maxMinutes) {
                    phoneOn = false;
                }
            }
        }
    }
}
```

a) Does the `PhoneMonitor` class as shown above have (circle all that apply):

a race condition   potential for deadlock   a data race   none of these

If there are any problems, give an example of when those problems could occur. Be specific!

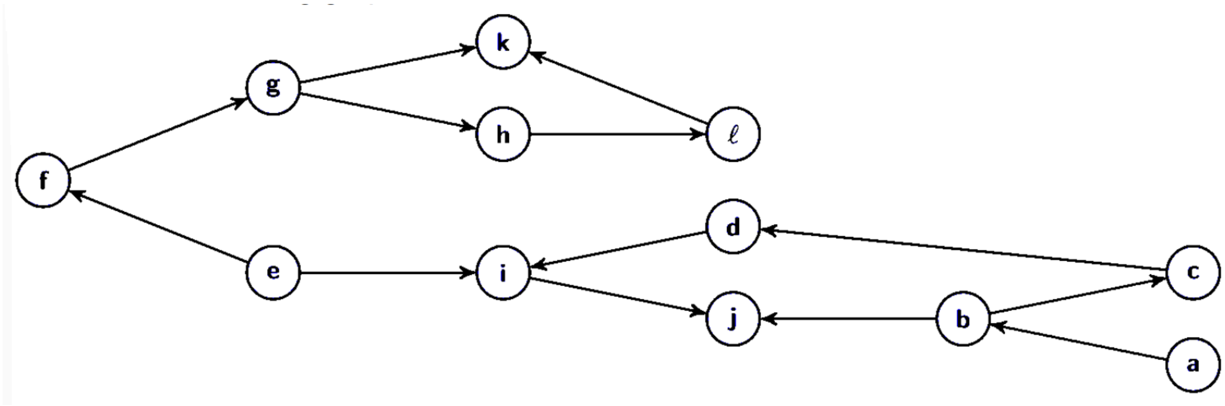
b) Suppose we made the `checkLimits` method public, and changed nothing else in the code. Does this modified `PhoneMonitor` class have (circle all that apply):

a race condition   potential for deadlock   a data race   none of these

If there are any FIXED problems, describe why they are FIXED. If there are any NEW problems, give an example of when those problems could occur. Be specific!

3. It Rhymes with Flopological Sort

Consider the following graph:



a) Does this graph have a topological sort? Explain why or why not. If you answered that it does not, remove the MINIMUM number of edges from the graph necessary for there to be a topological sort and carefully mark the edge(s) you are removing. Otherwise, just move on to the next part.

For the remaining parts, work with this (potentially) new version of the graph.

b) Find a topological sort of the graph. Do not bother showing intermediary work.

Snow Day

After 4 snow days last year, UW has decided to improve its snow response plan. Instead of doing "late start" days, they want an "extended passing period" plan. The goal is to clear enough sidewalks that everyone can get from every classroom to every other \textbf{eventually} but not necessarily very quickly.

Unfortunately, UW has access to only one snowplow. Your goal is to determine which sidewalks to plow and whether it can be done in time for the first 8:30 AM lectures.

You have a map of campus, with each sidewalk labeled with the time it will take to plow to clear it.

a) What will the vertices of your graph be?

b) What will the edges be? You should at least say whether your edges are directed or not and whether they're weighted or not.

c) What algorithm will you run on your graph?

d) How will you interpret the output of your algorithm? (i.e. which sidewalks to plow “in the real world" instead of just in graph terms).

e) Briefly (2-4 sentences) explain why your model works. You should at least address why you ran the algorithm you did (e.g., why are you looking for a shortest path/MST/topological ordering/etc.) and how you are ensuring your algorithm will be able to produce an "extended passing period" plowing plan.