Name: _____

UW NetID: _____

CSE 332 Summer 2019: Final Exam (pt. 2)

(closed book, closed notes, no calculators)

Instructions: Read the directions for each question carefully. We can only give partial credit based on the work you write down, so show your work. If you are running out of time, take note of which questions are weighted more and budget your time accordingly.

For questions where you are drawing pictures, please circle your final answer.

Good Luck!

Total: 62 points. Time: 60 minutes.

Question	**Max Points**
1 Concurrency	12
2 Parallel Methods	10
3 Shortest Path	12
4 MSTs	9
5 Graph Questions	8
6 P/NP	11
Total	62

1 Concurrency [12 points]

Summer is construction season and you have been tasked with writing code to manage contracts between several construction companies. These transactions can happen simultaneously across multiple threads. Consider this first bit of code from your Company class.

```
public class Company {
    private Dictionary<Contract> contracts;
    private String companyName;
    public Lock companyLock;

    public void removeContract(String key) {
        companyLock.lock();
        contracts.delete(key);
        companyLock.release();
    }
    public void addContract(String key, Contract c) {
        companyLock.lock();
        contracts.insert(key, c);
        companyLock.release();
    }

    \\other code further down
    \\...
}
```

1. Is it necessary for companyLock to be re-entrant? Why or why not? If you do not have enough information to answer, explain what you would need to know before you could answer.

   Solution: There is not enough information. To say one way or the other we would need to know if any other functions in the class use companyLock and call removeContract or addContract while holding the lock.

2. Assume the Dictionary class `contracts` uses is an AVLTree class Rob wrote for P2. He got a little sloppy so sometimes delete() will throw exceptions. What potential issue (beyond the exception itself) does this create in the removeContract function?

   Solution: It is possible to exit the function via an exception without having released the companyLock, thus permanently blocking any other threads waiting to acquire the lock.

As it turns out, all construction contract changes need to be registered with the Seattle Records Office. We'll now reveal another method in the same class that shows an example of this process.

```
public transferContractTo(String key, Company recordsOffice,
                          Company receiver) {
    //get our old contract
    Contract c = contracts.find(key);

    //let the records office know
    recordsOffice.companyLock.lock();
    recordsOffice.updateRecord(c, receiver); //don't worry what's in here
    recordsOffice.companyLock.release();

    //delete our contract and move it to the other company
    companyLock.lock();
    removeContract(key);
    receiver.companyLock.lock();
    receiver.addContract(c);
    receiver.companyLock.release();
    companyLock.release();
}
```

3. Are there now any potential data races in the class? Explain your answer.

Solution: Yes. We do not know the details of what's happening in .find, but without a lock around it there could be bad interleavings with delete or insert since these both open up the potential for data races.

4. Assuming there is only one records office, is there the potential for deadlocks in this function? If yes, give an example interleaving. If no, explain why.

Solution: Yes. In the case two companies are trying to transfer contracts to each other:
Thread 1: companyLock.lock();
Thread 2: companyLock.lock();
Thread 2: receiver.companyLock.lock(); (this blocks thread 2)
Thread 1: receiver.companyLock.lock(); (this blocks thread 1)

2 Parallel Methods [12 points]

Given an array of movie name strings with n total elements (which may include duplicates), describe a series of parallel actions to output a list of **unique** movie names that start with "The". Do not write pseudo-code for this. Instead, for each step in the process specify the input and output, which parallel algorithm you are using (some version of map,reduce,prefix,filter or sort), and the work and span.

As a separate example, if you were asked to find the sum total of the lengths of an array of strings your solution could look like:

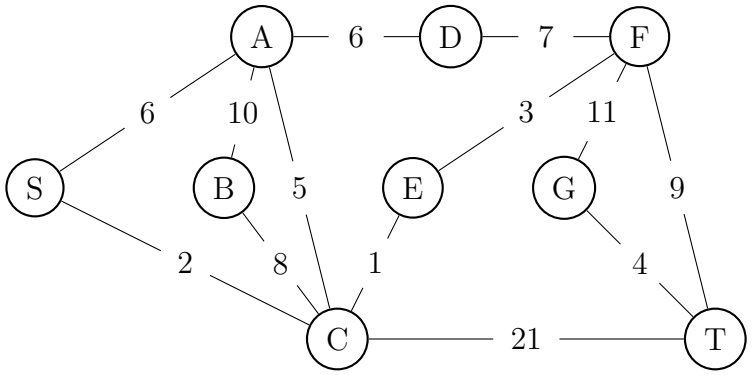
Step 1: Run a map on the array to convert strings to their respective lengths.
Work: _ Span: _
Step 2: Use a reduce on the output of step 1 to calculate the total sum of the array.
Work: _ Span: _

Solution:

- Step 1: Run a parallel filter on the movie list to filter out any movies whose names don't start with "The". Work $O(n)$ Span $O(\log n)$
- Step 2: Run a parallel quicksort on the output from step 1 to create a sorted list of "The" movies. Work $O(n \log n)$ Span $O(\log^2 n)$
- Step 3: Run a map on the output from step 2 that sets output[i] = input[i] if input[i-1] != input[i], or sets output[i] = "" if input[i] = input[i-1]. Work $O(n)$ Span $O(\log n)$
- Step 4: Run another filter on the output of step 3 to eliminate blank "" values. Work $O(n)$ Span $O(\log n)$

3 Shortest Path [12 points]

1. Perform Dijkstra's Algorithm on the graph below. Fill in the corresponding table as you go, crossing out old values instead of erasing them. Start from the S vertex. Halt the algorithm as soon as the T vertex is marked as processed. In the case of ties examine nodes in alphabetic order.



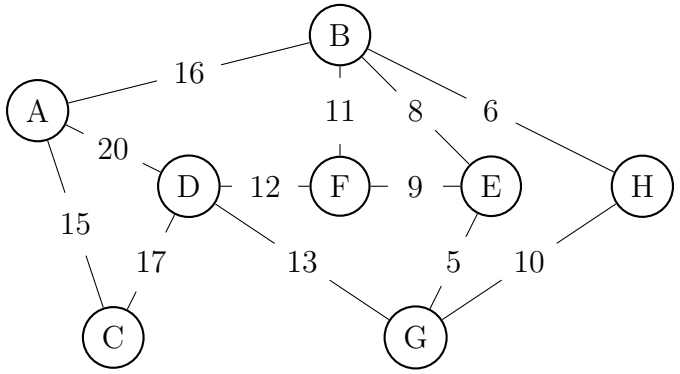
Vertex	Distance	Predecessor	Processed
S	0	N/A	✓
A	~~∞~~ 6	S	✓
B	~~∞~~ 10	C	✓
C	~~∞~~ 2	S	✓
D	~~∞~~ 12	A	✓
E	~~6~~ 3	C	✓
F	~~∞~~ 6	E	✓
G	~~∞~~ 17	F	
T	~~∞~~ ~~23~~ 15	~~F~~ F	✓

2. For the same graph, but ignoring edge weights, what order would a Breadth First Search add nodes onto the queue? In the case of ties, give nodes preference in alphabetical order. Also, what would be the derived shortest path from S to T?

Solution: Oder: S,A,C,B,D,E,T,F,G
Path: S,C,T

4 Minimum Spanning Trees [9 points]

1. Perform Prim's Algorithm on the graph below. Do not erase old values, instead cross them out so we can follow your process. Start your algorithm from the A vertex.



Vertex	Cost	Predecessor	Processed
A	0		✓
B	~~∞~~ 16	A	✓
C	~~∞~~ 15	A	✓
D	~~∞ 20 17 13~~ 12	~~A A G~~ F	✓
E	~~∞~~ 8	B	✓
F	~~∞~~ ~~11~~ 9	~~B~~ E	✓
G	~~∞~~ ~~16~~ 5	~~11~~ E	✓
H	~~∞~~ 6	B	✓

2. In at most a few sentences, describe the cut property of graphs that we used to explain how greedy algorithms like Prim's work for finding MTSs in linear time. Assume any graph in question has unique edge weights.

Solution: The cut property of graphs states that if we divide a graph's vertices into any two distinct groups, the smallest edge that connects these two groups will always be part of the MST.

5 Graph Short-Answer Questions [8 points]

In a sentence or two explain each of your answers for this section.

1. Will Prim's Algorithm find a minimum spanning tree if there is a negative cost cycle somewhere in the graph? Why or why not?

   Solution: Yes, Prim's only considers edges in order of cost and works just as well for negative edge valued graphs.

2. For a weakly-connected graph, that is not strongly connected, with n nodes what is the largest number of strongly connected components that can exist in such a graph? What is the smallest number that can exist?

   Solution: Largest: n Each node is its own SCC
   Smallest: 2 The entire graph cannot be an SCC and weakly-connected so 2 is the next smallest number.

3. Assuming the edge weights in a graph are not unique, will Prim's always find the same minimum spanning tree as Kruskal's? Why or why not?

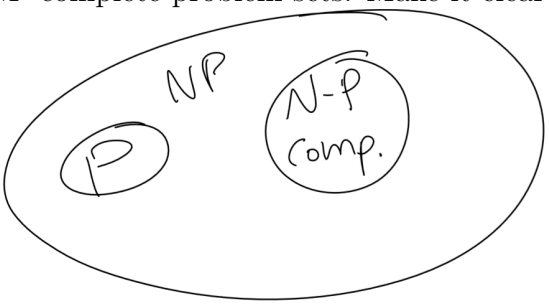
   Solution: If edges are not unique then there may be multiple possible minimum spanning trees. There is no guarantee that Prim's and Kruskal's return the same trees. It depends on how edges are ordered in the case of ties.

4. Imagine a directed, acyclic graph with n nodes and $n-1$ edges. One node has an in-degree of $n-1$, the other nodes each have an out-degree of 1. How many valid topological sort orderings are there for this graph?

   Solution: We can choose any order for the first $n-1$ nodes but the nth node must come last. Thus there are (n-1)! valid topological orderings.

6 P/NP [11 points]

1. Draw what we're pretty sure at this point is the relationship between the P, NP, and NP-complete problem sets. Make it clear where there is overlap.



2. Name one example problem that exists in each set. Do not name the same problem more than once.

   (a) P

   Solution: Shortest path in a positive-edged graph

   (b) NP

   Solution: Traveling Salesman problem

   (c) NP-Complete

   Solution: Hamilton circuits

   (d) None of the three above sets

   Solution: NxN Chess

3. True or False (circle your answer)

   (a) Solving (in polynomial time) a problem in NP but not P is sufficient to prove P=NP T / F

   (b) A problem is NP-complete if we can convert it in polynomial time to some known NP-complete problem T / F

(c) Any problem whose solution can be verified in polynomial time is in NP T / F

(d) Any problem whose solution can be verified in linear time is in P T / F

Some Useful Facts

When we're using the tree method to solve a recurrence, we usually use the following steps:

0. Draw a few levels of the tree.
1. Let the root node be at level 0. Give a formula for the size of the input at level i.
2. What is the number of nodes at level i?
3. What is the work done at the i^{th} recursive level?
4. What is the last level of the tree?
5. What is the work done at the base case?
6. Write an expression for the total work done.
7. Simplify until you have a "closed form" (i.e. no summations or recursion).

Geometric series identities:

$$\sum_{i=0}^{k} c^i = \frac{c^{k+1}-1}{c-1} \qquad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c} \text{ if } |c| < 1$$

Common Summations:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2} \qquad \sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \qquad \sum_{i=0}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

Log identities:

$$a^{\log_b(c)} = c^{\log_b(a)} \qquad \log_b(a) = \frac{\log_d(a)}{\log_d(b)}$$

Master Theorem:
Given a recurrence of the following form:

$$T(n) = \begin{cases} d & \text{if } n \leq \text{ some constant} \\ aT(n/b) + n^c & \text{otherwise} \end{cases}$$

with a, b, c are constants.
If $\log_b(a) < c$ then $T(n)$ is $\Theta(n^c)$
If $\log_b(a) = c$ then $T(n)$ is $\Theta(n^c \log n)$
If $\log_b(a) > c$ then $T(n)$ is $\Theta\left(n^{\log_b(a)}\right)$