Name(last, first): ___________________

U C L A Computer Science Department

CS 180 **Algorithms & Complexity** **ID : ____________**

Midterm **Total Time: 1.5 hours** **November 2, 2020**

Each problem has 20 points .

All algorithms should be described in bullet format (with justification/proof). You cannot quote any time complexity proofs we have done in class: you need to prove it yourself.

Problem 1: Describe the Breadth First Search algorithm in a DAG. Prove its correctness. Analyze its complexity.

Name(last, first): ____________________

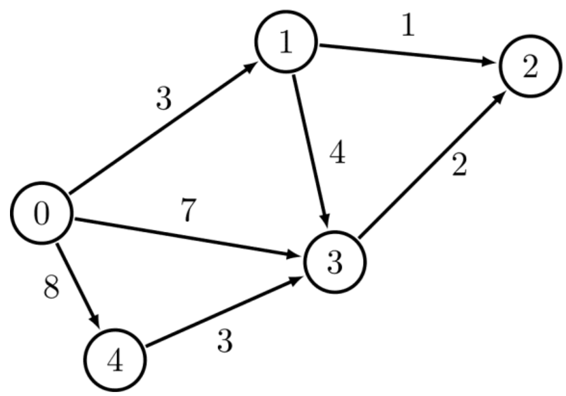
Problem 2: Consider a set of intervals/tasks. Each task has a start and an end time and each processor can handle one task at any given time. If tasks do not overlap, then we can use one processor to schedule them all. If they do overlap, we need more processors to schedule them. For example, in the figure below we need two processors to schedule all four intervals/tasks. A. Design an algorithm that finds the minimum number of processors needed to schedule all intervals/tasks. B. Analyze the time complexity of your algorithm. C. Prove the correctness of your algorithm.

```
Processor 1    ----------------------          ----------------------------
Processor 2             ------------------------------            ----------------------------
```

Problem 3: Design an algorithm that decides if a connected undirected graph is 2-colorable and finds a 2-coloring if it is indeed 2-colorable. Prove the correctness of your algorithms and analyzes its time complexity.

Name(last, first): ____________________

Problem 4: Apply the DFS algorithm to the graph shown below (step by step) starting from vertex zero (0), and show the final DFS Tree. (You can ignore the weight on the edges.)



Name(last, first): ____________________

Problem 5: It takes n-1 comparisons to find the minimum number in a given list of integers L = (x1, x2, x3, ….). Similarly, it takes n-1 comparisons to find the maximum. Therefore, it is trivial to design an algorithm that finds both the minimum and the maximum with about 2n-2 comparisons. Design an algorithm to find both the minimum and the maximum in a list using about 3n/2 comparisons.