This collection of problems is NOT a sample final. It does represent problems that will be similar in spirit to the "harder" problems on the final. The real final will also have some problems that are easier, and are a more straightforward application of material from class.

1 Graph Problems

1.1 Interacting Proteins

You work with biologists studying interacting proteins. They give you a graph G, where nodes are proteins, and there is an edge between two nodes if the proteins interact with each other. In addition, each protein is labeled with what part of the cell it resides in: Cell Wall (W), Cytoplasm (C), or Nucleus (N). Eg., given any node a, $PartOfCell(a) \in \{W, C, N\}$.

The biologists want you to find all proteins that interact with one protein in each part of the cell. Give an efficient algorithm to do this.

1.2 Kevin Bacon Game

Recall the Kevin Bacon Problem: you are given a graph, with a node for every actor, and an edge between two nodes if the corresponding actors were in a movie together. Then, if you are given a specific actor, you are supposed to find a path from that actor to Kevin Bacon in the graph. After watching "The Untouchables", your friend says that if it weren't for Sean Connery, then there are some people who no longer have any path to Kevin Bacon. Give an efficient algorithm that will prove whether or not your friend is right.

2 Greedy Algorithms

2.1 job scheduling

Chapter 4, Problem 7

3 Dynamic Programing

3.1 Knapsack

Given a knapsack with maximum weight $W = 17$, and the following four items, fill in the table to compute the optimal items, and state what items should be packed.

item #	value	weight
1	5	3
2	7	4
3	21	7
4	15	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																		
2																		
3																		
4																		

3.2 RNA fold

State what is wrong with this implementation of RNA fold, and state how you would fix it.

---

procedure SCORE($start, end$)
  if $end - start \leq 4$ **then return** 0
  else
    $s \leftarrow max_{start \leq t < end-4}(1 + \text{SCORE}(start, t-1) + \text{SCORE}(t+1, end-1))$
    $s \leftarrow max(s, \text{SCORE}(start, end-1))$
  end if
end procedure

3.3 Independent Set on Paths

Chapter 6, Problem 1

4 NP

4.1 Proving a problem is in NP

For problems 2 - 5 in chapter 8, prove that the given problem is in NP. You do NOT need to prove that they are NP-complete. (You will not be required to come up with any reductions on the test.)

4.2 Bad Reductions

Consider the decision problem $PATH$: Given a graph G, and two nodes s and t in the graph, is there a path in the graph from s to t.

Here are incorrect proofs that $PATH$ is NP-complete. State what is wrong in these proofs.

Proof. We will show that $PATH$ is NP-complete by showing $3SAT \leq_P PATH$.

Given a graph G, and nodes s, t, first check if there is a path from s to t (eg., with BFS). If there is a path, output the formula:

$$x_1$$

else, output the formula:

$$x_1 \wedge \neg x_1$$

Thus, if there is a path, we output an instance of $3SAT$ which is satisfiable, and if there is not a path, we output an instance of $3SAT$ which is not satisfiable. □

Proof. We will show that $PATH$ is NP-complete by showing $3SAT \leq_P PATH$.

Given an instance X of the $3SAT$ problem, solve it. If it satisfiable, output a graph with 2 nodes, s and t, with an edge between s and t. If it is not satisfiable, output a graph with

2 nodes, s, and t, with no edges. Thus, if there is a solution to X, we output a graph with a path, and if there is no solution to X, we output a graph without a path. □

4.3 Complexity Classes

Chapter 8, problem 1.