CSCI 3104 Spring 2024
Instructors: Profs. Divya Vernerey and Rachel Cox

Problem Set 13

Due Date	April 25, 2024
Name	**Mason Bott**
Student ID	**110435838**
Collaborators	**N/A**

Contents

Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeX can be found here on Canvas.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).

- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section hyperlinkHonorCodeHonor Code). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign the Honor Pledge)

Problem HC. On my honor, my submission reflects the following:

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

In the specified region below, clearly indicate that you have upheld the Honor Code. Then type your name.

Honor Code. I agree to the above, Mason Bott □

28 Standard 28 - Computational Complexity: Formulating Decision Problems

28.1 Problem 1

Problem 1. Recall the Maximum Flow problem:

- Input: A flow network, which consists of a directed graph $G = (V, E)$, two vertices $s, t \in V$, and a non-negative capacity function $c\colon E \to \mathbb{R}_{\geq 0}$.
- Decide: The maximum value of an $s - t$ flow in G

Formulate the decision variant of this problem using the Input/Decide format below.

Answer. Input/decide variant of the problem:

Input: A flow network, which consists of a directed graph $G = (V, E)$, two vertices $s, t \in V$, a non-negative capacity function $c\colon E \to \mathbb{R}_{\geq 0}$, and a natural (non-negative) number M.

Decide: Is there an $s - t$ flow in G of value at least M?

Since decision variant problems have to be answered with a yes or no, I added an additional input for representing a desired flow, M. The decision is whether or not the flow networks has a flow value of at least M. □

28.2 Problem 2

Problem 2. Recall the String Alignment problem:

- Input: Two strings s1 and s2, and four types of edits are provided (i.e., insert, delete, substitute, and no-op)
- Decide: The minimum number of edits required to convert texttts1 to s2

Formulate the decision variant of this problem using the Input/ textsfDecide format below.

Answer. Decision varaint of the problem:

Input: Two strings s1 and s2, four types of edits (insert, delete, substitute, no-op), and an (non-negative) integer K.

Decide: Can we align s1 and s2 in K edits or less.

Since we need to answer the question with a yes or no answer, I added a parameter K that represents the desired number of edits. As a result, we can make the decision whether or not the string can be aligned to the other string in K or less edits, which is a yes or no question. □

29 Standard 29 - Computational Complexity: Problems in P

29.1 Problem 1

Problem 1. Consider the decision variant of the Interval Scheduling problem.

- Input: Let $\mathcal{I} = \{[s_1, f_1], \ldots, [s_n, f_n]\}$ be our set of intervals, and let $k \in \mathbb{N}$.
- Decide: Does there exist a set $S \subseteq \mathcal{I}$ of at least k pairwise-disjoint intervals?

Show that the decision variant of the Interval Scheduling problem belongs to P. You are welcome and encouraged to cite algorithms we have previously covered in class, including known facts about their runtime. [textbfNote: To gauge the level of detail, we expect your solutions to this problem will be 2-4 sentences. We are neither asking you to come up with a new algorithm nor to analyze an algorithm in great detail.]

Answer. In this problem, we can use the greedy algorithm that always selects the interval with the earliest finish time. This algorithm guarantees that we will get the highest number of pairwise-disjoint intervals because an interval with the soonest start time cannot prevent us from choosing any other non-overlapping intervals that may come later, and thus it will be optimal for comparing to k. This algorithm runs in worst case $\Theta(n \log(n))$ (if we need to sort the intervals) which is polynomial, and will directly provide an answer to the decision since our answer will either be less or greater than or equal to k, and thus this problem belongs to the P class of problems. □

30 Standard 30 - Computational Complexity: Problems in NP

30.1 Problem 1

Problem 1. Consider the Vertex Cover problem. A *vertex cover* in a graph $G = (V, E)$ is a subset of vertices, $C \subseteq V$, such that every edge touches at least one vertex in C.

- Input: An undirected graph $G = (V, E)$ and a positive integer k.
- Decide: Does G have a vertex cover of $\leq k$ vertices?

Show that this problem belongs to NP.

Answer. In order for a problem to belong to NP, we need to show that a solution, or given value of k with a candidate vertex cover satisfying k can be verified in polynomial time.

In this problem, this is true because we have two cases:

1. Our value of k is greater than or equal to $|C|$.

   As a result, we know already that the answer is yes in $\Theta(1)$ time as the whole graph is naturally a vertex cover making the solution trivial.

2. Our value of k is less than or equal to $|C|$.

   In this case, we need to check every Edge to see if it has at least one endpoint in the candidate vertex cover C. This would involving examining each edge once, and then scanning the vertex cover C to see if one of the endpoints belong to C. Since the number of edges in the graph is bounded by —E—, this process would take polynomial time: $\Theta(|E|)$

Since both cases above are complete in polynomial time, this problem belongs to the NP class of problems. □

31 Standard 31 - Structure and Consequences of P vs. NP

31.1 Problem 1

Problem 1. Consider the Vertex Cover problem. A *vertex cover* in a graph $G = (V, E)$ is a subset of vertices, $C \subseteq V$, such that every edge touches at least one vertex in C. Do the following:

1. List the steps required to show that a problem belongs to NP- Complete
2. Show that *vertex cover* $\in$ NP-Complete (**Hint:** Perform a reduction from an Independent Set.)

Answer. To show that a problem is NP complete, we need to first show the problem is NP, and then show it is NP-hard.

The problem is NP:
To show a problem is NP, we need to show the answer can be verified in polynomial time, which we performed in the previous problem.

The problem is NP-hard:
To show that a problem is NP-hard, we need to perform a reduction from a known NP-complete problem to the vertex cover problem in polynomial time. If this reduction can be done in polynomial time, then we know our problem is not easier than the NP-complete problem, and thus is NP-hard.

The Independent Set problem is NP-complete:
Input: A graph $G = (V, E)$ and an integer k.
Decide: Does there exist a set of vertices $S \subseteq V$ such that $|S| \geq k$ and no two vertices in S are adjacent.

Convert the above instance to vertex cover:

1. Use the same graph as with the Independent Set problem. Further, let our vertex cover size be $|V| - k$, the inverse of the Independent Set problem.
2. We need to show that there exists an independent set of size $\geq k$ in G if and only if there exists a vertex cover of size $\leq |V| - k$ in G.
3. If there exists an independent set S of size k in G, then the complement of s is a vertex cover of size $|V| - k$. This is a result of the fact that an independent set has no adjacent vertices, meaning if one vertex on an edge is in the set, the other can't be. If we take the complement of this, we will end up getting the other vertex of the edge, so the edge will still be included in the vertex cover.

As a result, we can transform any Independent Set problem instance into a vertex-cover problem-instance since they are essentially inverse problems.

Then the Vertex Cover is NP-Complete:
Since an arbitrary instance of the Independent Set problem can be transformed in polynomial time to the vertex cover problem, then the vertex cover problem is also NP-hard and thus NP-complete. □

31.2 Problem 2

Problem 2. A student has a decision problem L which they know is in the class NP. This student wishes to show that L is NP-complete. They attempt to do so by constructing a polynomial time reduction from L to SAT, a known NP-complete problem. That is, the student attempts to show that $L \leq_p \mathsf{SAT}$. Determine if this student's approach is correct and throughly justify your answer.

Answer. The student's approach of showing that L is NP-complete is **incorrect**.

To prove that a problem L is NP-complete, we need to show two things:

1. $L \in \mathsf{NP}$ (which is given).
2. Every problem instance of SAT can be reduced to L in polynomial time.

The student was on the right track, but they actually showed the inverse of step 2, that every problem instance of L can be reduced to SAT in polynomial time. In doing this backwards, the student would show that L was not harder than SAT, which isn't useful since we already know SAT is NP-complete. Instead, the student should attempt to construct a polynomial time reduction from SAT to L which will allow them to claim that SAT is no harder than L. By claiming this, we can further justify that L is NP-complete. □