CSCI 3104 Fall 2021
Instructors: Profs. Chen and Layer

Problem Set 10

Due Date ..... April 26
Name ..... **Chengming Li**
Student ID **109251991**
Collaborators ..... **List Your Collaborators Here**

Contents

1 Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- 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.

2 Standard 26 - Showing problems belong to P

Problem 1. Consider the Shortest Path problem that takes as input a graph $G = (V, E)$ and two vertices $v, t \in V$ and returns the shortest path from v to t. The shortest path decision problem takes as input a graph $G = (V, E)$, two a vertices $v, t \in V$, and a value k, and returns True if there is a path from v to t that is at most k edges and False otherwise. Show that the shortest path decision problem is in P. You are welcome and encouraged to cite algorithms we have previously covered in class, including known facts about their runtime. [**Note:** To gauge the level of detail, we expect your solutions to this problem will be 2-4 sentences. We are not asking you to analyze an algorithm in great detail.]

Answer.

- As given in the question, the decision problem is: is there a path from v to t that is at most k edges.
- We have seen Dijkstra's algorithm computes the shortest path from v to t in time $O(| V | + | E | log(| V |))$ with min-priority queue. And usually, the time complexity of Dijkstra's algorithm is $O(V^2)$
- Thus, we can apply Dijkstra's algorithm to G. And it will return a path that is computed in **polynomial** time. Then, we could compare the number of edges in this path to the value k. If the number of edges are less or equal to k, we could assert that there is path from v to t in the graph G at most k edges. So, this **Path** $\in$ **P** and Shortest Path decision problem $\in$ **P**

□

3 Standard 27 - Showing problems belong to NP

Problem 2. Consider the Simple Shortest Path decision problem that takes as input a directed graph $G = (V, E)$, a cost function $c(e) \in \mathbb{Z}$ for $e \in E$, and two vertices $v, t \in V$. The problem returns True if there is a simple path from v to t with edge weights that sum to at most k, and False otherwise. Show this problem is in NP.

Answer.

- Simple Path is the path from one vertex to another such that no vertex is visited more than once.
- Suppose graph G has such a simple **path** between v to t with edge weights that sum to at most k. There three things we need to check: is it a valid path, is it a simple path and is it length at most k.
- **Claim1: Simple Shortest Path decision problem has a valid path from v to t .**

In this claim, we need to check any edges in this **path** is $\in E$ A. In other words , any edges in this **path** is the edge in graph G. And we need to check the there is path from v to t. The time complexity to valid this claim is $O(n^2)$, where n is the number of edges in this **path**.

- **Claim2: Simple Shortest Path decision problem has a valid Simple path**.

In this claim2, we need to check every vertex in this path has been visited precisely once. The time complexity to check whether each vertex is included once is O(n), where n is the number of edges in this **path**.

The algorithm to check the whether each vertex has been visited precisely once is: define a **Visited** element for each vertex. And we mark each un-visited vertex to false at the beginning of the algorithm. Once the algorithm starts, we check each vertex in this **path** if it has been visited or not. if it has been visited before, then this **path** is not a simple path. if every vertex has been only visited only once, they **path** is a simple path. The time complexity is O(k), where k is number of vertex in the **path**.

- **Claim3: Simple Shortest Path decision problem has at most k edge weight**.

In this claim 3, we need to compare the **path** edge weights to k. if the edge weights are at most k, or less and equal to k, then we can assert this path has at most k edge weights. The time complexity of comparison is O(c), where c is a constant.

- **Conclusion:**.

Hence, Simple shortest path decision problem $\in$ NP if the **path** pass all three claims above in polynomial time.

Hence, we conclude that Simple Shortest Path decision problem $\in$ **NP**.

□

Problem 3. Indiana Jones is gathering n artifacts from a tomb, which is about to crumble and needs to fit them into 5 cases. Each case can carry up to W kilograms, where W is fixed. Suppose the weight of artifact i is the positive integer w_i. Indiana Jones needs to decide if he is able to pack all the artifacts. We formalize the Indiana Jones decision problem as follows.

- Instance: The weights of our n items, $w_1, \ldots, w_n > 0$.
- Decision: Is there a way to place the n items into different cases, such that each case is carrying weight at most W?

Show that Indiana Jones $\in$ NP.

Answer.

- Suppose we have a way to place n items into different cases, such that each case is carrying weight at most W.
- **Claim1:Indiana Jones problem has valid way to place n items into different cases**.

We add up all items from each case and we check if the total number of items is same as n. Second, we visit every item in each case and we check if every item has been only visited once.
We can assert that there is a valid way to place the n items into different cases if there are n items in total and every item has been visited precisely once.
The time complexity of checking n items is O(n)

- **Claim2:Each case has at most W weight**.

In this claim2, for each case, we add up the weight of every item w_i and we compare the total weight to W. If the total weights in each case is at most W kilograms, we can assert each case is carrying weight at most W.
The time complexity of this claim is O(n), because we just do summation of all those items in each case, and there is n item.

- **Conclusion:**.

Hence, the Indiana Jones $\in$ NP because claim1 and claim 2 can be solved in polynomial time. □

4 Standard 27 - NP-compelteness: Reduction

Problem 4. 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 justify your answer.

Answer.

- This student's approach is **incorrect**.
- Student's approach is correct if we flip the statement as $\mathsf{SAT} \leq_p \mathsf{L}$.
- The definition of NP complete is: A problem that is **NP - hard** and is a member of **NP**
- **Claim1: L is in the class NP**.

As given in the question statement, the decision problem L is in the class NP.

- **Claim2: L is in the class NP-hard**.

SAT problem is well-known as NP-complete problem. Thus, we know that $\mathsf{NP} \leq_p \mathsf{SAT}$, and then we could know SAT is at least as hard as any NP-problem. Hence we reduce SAT problem to L, which reduce an NP-complete problem to a NP problem. But , L must be NP-hard because SAT is at least as hard as any NP-problem.

So, L is in the class NP-hard class. And So L is NP-complete problem.

- **From the perspective of student's approach:**.

We only know L is in the class NP so that we couldn't find a way to reduce problem L to SAT. In other words, we don't have enough information to use reductions to reduce from a NP problem to a NP-complete problem. □

Problem 5. Consider the Simple Shortest Path decision problem that takes as input a directed graph $G = (V, E)$, a cost function $c(e) \in \mathbb{Z}$ for $e \in E$, and two vertices $v, t \in V$. The problem returns True if there is a simple path from v to t with edge weights that sum to at most k, and False otherwise. Show this problem is NP-compelte.

Answer. **We show that Simple Shortest Path decision problem is in NP, and that Simple Shortest Path decision problem is in NP-hard**

- **Claim1: Simple Shortest Path decision problem is in NP, Copy and Paste from Problem2**.

  Suppose graph G has such a simple **path** between v to t with edge weights that sum to at most k. There three things we need to check: is it a valid path, is it a simple path and is it length at most k.

  - **ClaimA: Simple Shortest Path decision problem has a valid path from v to t .**

  In this claim, we need to check any edges in this **path** is $\in E$ A. In other words , any edges in this **path** is the edge in graph G. And we need to check the there is path from v to t. The time complexity to valid this claim is O(n^2), where n is the number of edges in this **path**.

  - **ClaimB: Simple Shortest Path decision problem has a valid Simple path**.

  In this claim2, we need to check every vertex in this path has been visited precisely once. The time complexity to check whether each vertex is included once is O(n), where n is the number of edges in this **path**.

  The algorithm to check the whether each vertex has been visited precisely once is: define a **Visited** element for each vertex. And we mark each un-visited vertex to false at the beginning of the algorithm. Once the algorithm starts, we check each vertex in this **path** if it has been visited or not. if it has been visited before, then this **path** is not a simple path. if every vertex has been only visited only once, they **path** is a simple path. The time complexity is O(k), where k is number of vertex in the **path**.

  - **ClaimC: Simple Shortest Path decision problem has at most k edge weight**.

  In this claim 3, we need to compare the **path** edge weights to k. if the edge weights are at most k, or less and equal to k, then we can assert this path has at most k edge weights. The time complexity of comparison is O(c), where c is a constant.

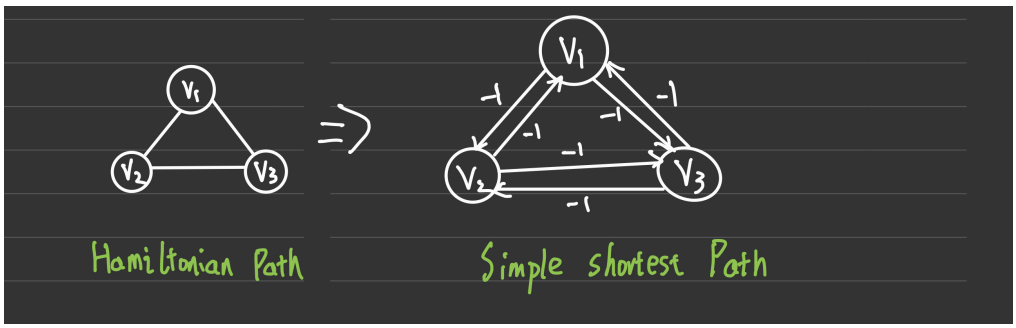
  - **Conclusion:**.

  Hence, Simple shortest path decision problem $\in$ NP if the **path** pass all three claims above in polynomial time.

  Hence, we conclude that Simple Shortest Path decision problem $\in$ **NP**.

- **Claim2: Simple Shortest Path decision problem is in NP-Hard**.

  We show Hamiltonian Path $\leq_P$ Simple Shortest Path.
  First, we reduce Hamiltonian Path problem to a Simple Shortest Path problem.



  - **Construction and Reduction**.

We replace $\{v1, v2\}$ in Hamiltonian Path to two edge $\{v1, v2\}\{v2, v1\}$.And define the cost for each new edge as -1 in simple Shortest Path. Last, replace the remaining edges in Hamiltonian Path with the same rules. The time complexity to convert the graph from Hamiltonian path to Simple Shortest Path is **O(cV)**, where c is constant and V is the number of vertex in graph G.

– **Examples**.

As seen in the graph, the shortest simple path between two vertices is actually the longest simple path due to negative edge cost. To find Simple Shortest Path from v to t, visiting any extra un-visited vertex will make the path shorter due to negative number.
For example, given v1 as v and v2 as t, the shortest path is v1 $->$ v2 $->$ v3 with cost -2. Rather than $v1 -> v2$ with cost -1.
Then, if the cost to the weighted graph is -1(V-1), where V is the number of the vertex in the graph G, we can convert the path to a path in the un-directed graph, which is a Hamiltonian path. This is because the path visits each vertex exactly once.

– **Conclusion**.

Hence, Simple Shortest Path decision problem is NP-hard since we reduce Hamiltonian Path Problem to it in Polynomial-time.

□