Nondeterministic Polynomial Quals Problems

Sijian Tan

1 Indep-Set to Indep-Set-Even (HW4)

Given a simple undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, a subset $I \subseteq V$ in G is an independent set if no two vertices in I are connected by an edge in E.

The INDEPENDENT-SET problem is: for a given graph G, does there exist an independent set with size at least k?

Here we have a variant of the INDEPENDENT-SET problem, which we call the INDEPENDENT-SET-EVEN problem. In the INDEPENDENT-SET-EVEN problem, all vertices in G have even degrees, and we ask whether there exists an independent set for such a graph with at least l vertices.

Prove that INDEPENDENT-SET-EVEN is NP-complete. You can use the fact the INDEPENDENT-SET is NP-complete.

Hint: given an instance of INDEPENDENT-SET (G, k), you can construct a graph G' for the INDEPENDENT-SET-EVEN instance by adding extra vertices and edges to G and make sure that all vertices in G' have even degrees.

1.1 Prove Indep-Set-Even is in NP

Given a graph $G' = (V', E')$, where all vertices have even degrees and a candidate solution I', which is a subset of V', the verification includes two checks:

1. **Independence Check:** Iterate over each pair of vertices in I' and check if any pair is connected by an edge in E'. If the candidate solution I' contains m vertices, this process will take $O(m^2)$ time.

2. **Size Check:** Verifying that $|I'|$ is at least l is straightforward and could be done in constant time by comparing two integers.

1.2 Prove Indep-Set is polynomially reducible to Indep-Set-Even

1.2.1 Construction and Time

For any given Indep-Set instance $I_1 = (G, k)$, we construct an Indep-Set-Even instance I_2 as follows. First, note that if there are vertices with odd degrees in G, the number of such vertices is even. This is because that the total degree in the graph is even: $\sum_{v \in V} \text{degree}(v) = 2|E|$. The new graph G' is constructed by adding a new vertex

x, and edges between x and every vertex that has an odd degree. Because previously there are even number of odd degree vertices, so in G' every node has an even degree. This construction process takes polynomial time with respect to the number of vertices in G.

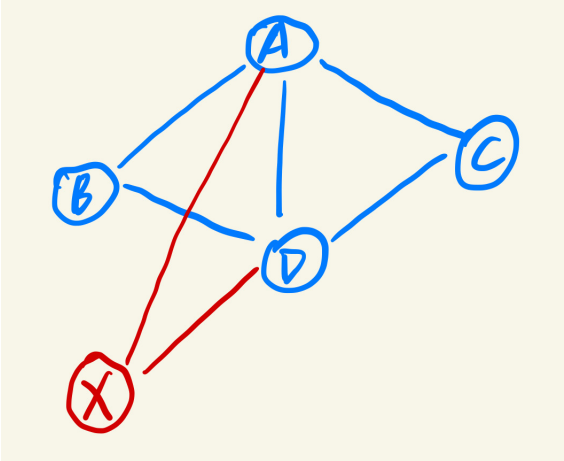

Figure 1: Indep-Set-Even Construction

1.2.2 If and Only If Proof

Suppose there is an independent set I of size k in G. Since adding v_0 and connecting it to vertices of odd degree does not affect I (we are not adding edges within I, so the vertices in I remain independent), so I is also an independent set in G'

Suppose there is an independent set I' of size k in G'. If v_0 is not in I', then the independent set will be the same in original graph G (during the construction, v_0 does not affect I), so G has an independent size of k. If v_0 is in I', then we can remove it and still have an independent set of size $k-1$ from the original graph G.

1.3 Conclusion

Since Indep-Set-Even is in NP, Indep-Set is polynomially reducible to Indep-Set-Even and Indep-Set is NP-Complete, so Indep-Set-Even is NP-Complete.

2 SAT to DOUBLE-SAT (HW4)

Recall that the **SATISFIABILITY (SAT)** problem is: given a Boolean formula ϕ of n variables $x_1, x_2, \ldots, x_n$, determine whether there exist at least one assignment of the n variables that evaluate ϕ to be TRUE. SAT is known to be an NP-complete problem.

The **DOUBLE-SAT** problem is: given a Boolean formula ϕ of n variables $x_1, x_2, \ldots, x_n$, determine whether there exist at least two assignments of the n variables that evaluate ϕ to be TRUE. Prove that DOUBLE-SAT is NP-complete using the fact that SAT is NP-complete.

For this problem, you can assume that ϕ is in the Conjunctive Normal Form (CNF) for both SAT and DOUBLE-SAT (no points will be taken off if you make this assumption).

Please remember to include all steps of the NP-completeness proof.

2.1 Prove DOUBLE-SAT is in NP

A potential solution could be two different assignment for the variable used in ϕ. We can simply iterate through each literal in ϕ, which takes linear time in terms of the number of literals in ϕ.

2.2 Prove SAT is polynomially reducible to DOUBLE-SAT

2.2.1 Construction and Time

For any given SAT instance $I_1 = \phi$ with variables $x_1, x_2, ..., x_n$, we construct a DOUBLE-SAT instance I_2 as follows. We introduce a new variable x_{n+1}, and construct $\phi' = \phi \wedge (x_{n+1} \vee \neg x_{n+1})$. This process takes polynomial time in terms of the number of literals.

2.2.2 If and Only If Proof

If ϕ has an assignment that evaluates ϕ to be true, the same assignment for variables $x_1, x_2, ..., x_n$ together with $x_{n+1} =$ TRUE or $x_{n+1} =$ FALSE can both satisfy ϕ'. Therefore ϕ' has two assignments that evaluate ϕ' to be TRUE.

If ϕ' has two assignments that evaluate ϕ' to be TRUE, we can take the values of $x_1, x_2, ..., x_n$ in any one of the assignments, and the same assignment will evaluate ϕ to be true. This is because, in order to satisfy ϕ', both ϕ and $(x_{n+1} \vee \neg x_{n+1})$ need to be satisfied as $\phi' = \phi \wedge (x_{n+1} \vee \neg x_{n+1})$. Therefore, there is at least one assignment that satisfies ϕ.

2.3 Conclusion

Since DOUBLE-SAT is in NP, SAT is polynomially reducible to DOUBLE-SAT and SAT is NP-Complete, so DOUBLE-SAT is NP-Complete.

3 Vertex Cover to Efficient Recruiting (HW5)

Suppose you are helping to organize a summer sports camp, and the following problem comes up. The camp is supposed to have at least one counselor who is skilled at each of the n sports covered by the camp (baseball, volleyball, and so on). They have received job applications from m potential counselors. For each of the n sports, there is some subset of the m applicants qualified in that sport. The question is: For a given number $k < m$, is it possible to hire at most k of the counselors and have at least one counselor qualified in each of the n sports? We'll call this the Efficient Recruiting Problem.

Prove that the Efficient Recruiting Problem is NP-complete using the fact that the Vertex Cover problem is NP-complete. Remember to follow all steps of the NP-completeness proof.

3.1 Prove Efficient Recruiting is in NP

Given a set of k counselors, we can easily check whether they cover all n sports by verifying that each sport has at least one qualified counselor. Therefore, Efficient Recruiting Problem is in NP.

3.2 Prove Vertex Cover is polynomially reducible to Efficient Recruiting

3.2.1 Construction and Time

Given an instance of the Vertex Cover problem, consisting of a graph $G = (V, E)$ and an integer k, we construct an instance of the Efficient Recruiting Problem as follows:

- For each vertex v in G, we create a counselor C_v qualified in a unique sport, denoted as S_v
- For each edge e in G, we create a sport S_e
- A counselor C_v is qualified in sport S_e if and only if e has an endpoint equal to v

This transformation can be done in polynomial time since it requires creating counselors and sports for each vertex and edge.

3.2.2 If and Only If Proof

Suppose there exists a vertex cover C of size at most k in G. Since each edge e must have at least one endpoint in the vertex cover set, and every sport S_e has at least one qualified counselor C_v of the corresponding Efficient Recruiting instance. Therefore, each vertex cover set in G corresponds to a set of counselors in the ER.

Suppose there exists a solution to the ER instance with at most k counselors. Each counselor in this solution corresponds to a vertex in the vertex cover of G. Since each sport is covered by at least one counselor, so every edge in G will have at least one endpoint in the vertex cover set. Therefore, the vertex cover has size at most k.

3.3 Conclusion

Since ER is in NP, Vertex Cover is polynomially reducible to ER and Vertex Cover is NP-Complete, so ER is NP-Complete.

4 Vertex Cover to EXAMDESIGN (Final)

At the end of the semester, your professor needs to solve the following **EXAMDESIGN** problem. She has a list of problems, and she knows for each problem which students will really enjoy that problem. She needs to choose a subset of problems for the exam such that for each student in the class, the exam includes at least one question that student will really enjoy. On the other hand, the professor does not want to give the students too many questions for the exam, so the exam must also contain as few questions as possible.

Formalize the decision problem of **EXAMDESIGN** with appropriate notations (2 pts) and prove that the decision problem of **EXAMDESIGN** is NP-complete by using the fact that **VERTEXCOVER** is NP-complete (8 pts). Remember to include all the steps of the "recipe".

4.1 Decision Problem

Assume the list of problems as U, is there a subset of questions (exam) $U' \subseteq U$ of size at most k such that for each student in the class, the exam includes at least one question that student will really enjoy?

4.2 Prove EXAMDESIGN is in NP

Given a set of k questions, we can easily check in polynomial time that for every student s_i, there is at least one problem that s_i will enjoy. Therefore, EXAMDESIGN is in NP.

4.3 Prove Vertex Cover is polynomially reducible to EXAMDESIGN

4.3.1 Construction and Time

Given an instance of the Vertex Cover problem, consisting of a graph $G = (V, E)$ and an integer k, we construct an instance of the Efficient Recruiting Problem as follows:

- For each vertex v in G, we create a question node, denoted as S_v
- For each edge e in G, we create a student S_e
- A student S_e enjoys a question S_v if and only if e has an endpoint equal to v

This transformation can be done in polynomial time since it requires creating students and questions for each vertex and edge.

4.3.2 If and Only If Proof

Suppose there exists a vertex cover C of size at most k in G. Since each edge e must have at least one endpoint in the vertex cover set, and every studnet S_e has at least one question S_v to enjoy in the corresponding EXAMDESIGN instance. Therefore, each vertex cover set in G corresponds to a set in the EXAMDESIGN.

Suppose there exists a solution to the EXAMDESIGN instance with at most k questions. Each question in this solution corresponds to a vertex in the vertex cover of G. Since each question is enjoyed by at least one student, so every edge in G will have at least one endpoint in the vertex cover set. Therefore, the vertex cover has size at most k.

4.4 Conclusion

Since EXAMDESIGN is in NP, Vertex Cover is polynomially reducible to EXAMDESIGN and Vertex Cover is NP-Complete, so EXAMDESIGN is NP-Complete.

5 GRAPHVALUE (Spring 2024)

The GraphValue Problem is defined as follows:

- **Input:** An undirected graph G with n vertices, a list L of n nonnegative integers, and a bound b.
- **Question:** Is there a way to assign the numbers in L to the vertices of G such that (1) each number is assigned to exactly one vertex, and (2) the graph's value is not greater than b? The value of the graph is the sum of all the edge values; each edge value is the product of the numbers assigned to its endpoint vertices.

Prove that GraphValue is NP-complete by using the fact the VertexCover is NP-complete. VertexCover(G, k) is the problem of deciding whether in graph G there exists a subset of vertices S of size at most k such that all edges in G have at least one endpoint in the selected vertex set S. Remember to include all the steps of the NP-completeness proof.

Hint: when constructing a GraphValue instance from a VertexCover instance, consider assigning some vertices value 0 and setting b to 0.

5.1 Prove GRAPHVALUE is in NP

Given an assignment of numbers in L to the vertices of G, we could traverse all the vertices to verify that each number is assigned to exactly one vertex, and traverse all the edges to get the graph's value, both of which will take polynomial time.

5.2 Prove Vertex Cover is polynomially reducible to GRAPHVALUE

5.2.1 Construction and Time

Given a Vertex Cover instance with:

- A graph G'
- An integer k

We construct the instance of GRAPHVALUE by:

- Map all the nodes and edges in G' to form $G = (V, E)$
- Create a list L where each vertex $v \in V$ is assigned a number.
  - Assign the number 0 to vertices that are supposed to be in the vertex cover set.
  - Assign the number 1 to vertices that are not in the vertex cover set.
  - Set $b = 0$, so the total value of the graph is 0 when each edge in E has at least one endpoint with the assigned number 0.

This process will take polynomial time.

5.2.2 If and Only If Proof

Suppose there exists a vertex cover set S in G os size at most k, then every edge $e = (u, v) \in E$ has at least one endpoint in S. Based on the construction, all the values in this set are assigned to be 0. Therefore, every edge has value as 0, so the graph's value of G will be 0.

Suppose there is an assignment of values in G to make the graph's value become 0, then each edge must have at least one endpoint assigned the value 0, otherwise the graph's value will not be 0. Collect all of the vertices that were assigned the value 0 into a set S. By construction, every edge in E has at least one endpoint in S, so S is a vertex cover of G.

5.3 Conclusion

Because GRAPHVALUE is in NP, Vertex Cover is polynomially reducible to GRAPHVALUE, Vertex Cover is NP-Complete, so GRAPHVALUE is NP-Complete.

6 Degree-Constrained-ST (Spring 2023)

The Degree-Constrained-ST problem is defined as follows: given an undirected, unweighted graph $G = (V, E)$, does there exist a spanning tree of this graph where each node in the spanning tree has a degree of at most k?

Prove that this problem is NP-Complete using the statement that the HamPath problem is NP-Complete.

In the HamPath problem, we are given a connected graph G, and are asked whether it contains a simple path that visits all vertices of the graph (the path should visit each vertex exactly once).

Please remember to include all steps of the NP-completeness proof.

6.1 Prove Degree-Constrained-ST is NP

Given a spanning tree, we could traverse every node in the tree to verify that each node has a degree of at most k. This will be done by polynomial time.

6.2 Prove Hampath is polynomially reducible to Degree-Constrained-ST

6.2.1 Construction and Time

Given a Hampath instance as a connected graph G, we construct the Degree-Constrained-ST instance by:

- Map all the nodes and edges in G to get G'
- Set $k = 2$: this means that each vertex in the spanning tree is allowed to have a degree of at most 2.

This process will take polynomial time.

6.2.2 If and Only If Proof

If there exists a spanning tree in G' where each vertex has a degree of at most 2, then the spanning tree must be a simple path because any vertex with degree 2 can only be connected to two other vertices. This path is just the Hamiltonian path in G because it visit each vertex exactly once.

If the Hampath problem has a solution, then this path corresponds to a spanning tree of G' where each vertex has a degree of at most 2.

6.3 Conclusion

Since Degree-Constrained-ST is in NP, Hampath is polynomially reducible to Degree-Constrained-ST, and Hampath is NP-Complete, so Degree-Constrained-ST is NP-Complete.

7 Frenemies (Spring 2022)

Prove that the following decision problem is NP-complete. Given n students and a set of pairs of students who are enemies, is it possible to arrange a dinner around a round table so that two enemies do not sit side by side? Remember to include all the steps of the NP-completeness proof.

7.1 Prove Frenemies Problem is in NP

Given an arrangement of students, we can check in polynomial time whether any two adjacent student in this arrangement are enemies. If no such pair exists, then the arrangement is a valid solution.

7.2 Prove HamCycle is polynomially reducible to Frenemies

7.2.1 Construction and Time

We construct the instance of Frenemies with a graph G:

- Create a node for each student, this will be V

- For every pair of students who are not enemies, create an edge between them, this will be E

This process will take polynomial time, which is dependent on the number of the students.

7.2.2 If and Only If Proof

Suppose there is a Hamcycle in the graph G we constructed, that means there is one cycle that visits each vertex exactly once. Based on the construction, the edges exist only between students who are not enemies, so that two enemies will not sit side by side in the round table.

Suppose there is an arrangement that two enemies do not sit side by side around a round table, that means there is a cycle existing in the graph G that visits each vertex exactly once and returns to the starting vertex. This is just the HamCycle of the graph.

7.3 Conclusion

Because Frenemies is in NP, HamCycle is polynomially reducible to Frenemies, and HamCycle is NP-Complete, so Frenemies is NP-Complete.

8 Travel Salesperson (Spring 2015)

The classical Traveling Salesperson Problem requires to input a weighted undirected graph and find a minimum-weight cycle that includes all vertices of the graph. We consider the modified version of the problem, which requires to find a minimum-weight *path* through all vertices; that is, the salesperson has to visit all vertices without returning to the initial vertex. The classical Traveling Salesperson Problem is known to be NP-complete. Use this fact to prove that the modified problem is also NP-complete.

8.1 The Modified TSP is in NP

To show that the modified TSP is in NP, we must demonstrate that a proposed solution (a Hamiltonian path) can be verified in polynomial time.

Given an instance of the modified TSP and a proposed solution, which is a sequence of vertices $\langle v_1, v_2, \ldots, v_n \rangle$, we can check the following in polynomial time:

- Verify that each vertex in V' appears exactly once in the sequence (this takes $O(n)$ time).
- Verify that the sequence forms a valid path by checking the existence of edges between consecutive vertices in the sequence (this takes $O(n)$ time).
- Calculate the total weight of the path and verify that it is indeed the minimum (this takes $O(n)$ time).

Since all these steps can be performed in polynomial time, the modified TSP is in NP.

8.2 Reduction from Classical TSP to Modified TSP

We will reduce an instance of the classical TSP to an instance of the modified TSP. Consider an arbitrary instance of the classical TSP defined on a graph $G = (V, E)$ with $|V| = n$ vertices.

Constructing the Modified Instance:

- **Graph Construction:**
  - Let $G' = (V', E')$ be a graph where $V' = V \cup \{v_0\}$ (i.e., the original vertices plus a new vertex v_0).
  - For each vertex $v_i \in V$, add an edge (v_0, v_i) with weight 0.
- **Objective:** In the modified TSP, we need to find a minimum-weight Hamiltonian path in G' starting from v_0 and visiting all vertices in V' exactly once.

Mapping Solutions Between Problems:

- **From TSP to Modified TSP:**
  - If we solve the modified TSP on G' and find a minimum-weight Hamiltonian path, say $P = \langle v_0, v_1, v_2, \ldots, v_n \rangle$, the subpath $\langle v_1, v_2, \ldots, v_n \rangle$ corresponds to a Hamiltonian cycle in the original graph G.
  - The weight of this path in G' will be equal to the weight of the corresponding cycle in G since the edges involving v_0 have zero weight and do not contribute to the total weight.
- **From Modified TSP to TSP:**
  - Conversely, given a solution to the classical TSP (a minimum-weight Hamiltonian cycle in G), removing one edge from the cycle produces a Hamiltonian path in G' (starting from any vertex and not returning to the start).
  - We can prepend the vertex v_0 to the path, forming a valid solution to the modified TSP with the same total weight.

Step 3: The Modified TSP is NP-Complete

Given that the classical TSP is NP-complete, and we have shown how to reduce the classical TSP to the modified TSP in polynomial time, it follows that the modified TSP is at least as hard as the classical TSP. Since the classical TSP is NP-complete and the modified TSP is in NP, the modified TSP is also NP-complete.