ShanghaiTech University

CS240 Algorithm Design and Analysis

Fall 2023

Problem Set 3

Due: 23:59, Dec. 12, 2023

1. Submit your solutions to Gradescope (www.gradescope.com).
2. In "Account Settings" of Gradescope, set your FULL NAME to your Chinese name and enter your STUDENT ID correctly.
3. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
4. When submitting your homework, match each of your solution to the corresponding problem number.

Problem 1:

NAE-4-SAT: A clause in an instance of 4-SAT is a disjunction with four literals i.e. $(x_1 \vee x_2 \vee x_3 \vee x_4)$. A satisfied assignment for a collection of clauses is called *not all equal* (NAE) if the literals in each clause are not all equal to each other. In other words, at least one false, and at least one true. Prove NAE-4-SAT is NP-complete, using a reduction from 3-SAT.

Solution:

In order to prove NAE-3-SAT is NP Hard, reduce a known NP Hard problem to this problem, in our case 3-SAT to NAE-4-SAT.

The 3-SAT formula f is transformed by adding a variable v to every clause of the formula f.

For the pedantic's sake, we first have a polynomial reduction $3SAT \leq_q NAE-4-SAT$

Since the formula f now has four literals in each of the clauses, it becomes a 4-SAT formula. Now, the following propositions hold: If 3-SAT is true, this implies that at least one of the literals in every clause must be true. We may assign the newly added variable v, a value of 0. NAE-4-SAT also holds, since the literals that were true in formula f also hold in NAE formula and the variable v is assigned a false boolean value, that is at least one is true and another is false. If the 4CNF formula is satisfiable, then this particular assignment of variables must have at least one true literal and at-least one negative literal. If the additional variable v, has the boolean value false, then we can remove it easily from each clause, and then the 3-SAT formula will still be satisfiable. If the additional variable v has a boolean true value, then we can remove any of the literal from each clause, since the variable v will make the 3CNF formula satisfiable. Consider the example: f=$(x1 \vee x2' \vee x3) \wedge (x1' \vee x2' \wedge x3)$

NAE formula=$(x1 \vee x2' \vee x3' \vee v) \wedge (x1' \vee x2 \vee x3 \vee v)$

Let us assume one of the valid assignment for 4CNF formula to be (x1=0, x2=0, x3=0, v=1)

The 4-NAE formula is true because v=1 and x1=0 in every clause. Now if we negate all values, to remove v=1 later, even then the formula is satisfied by setting x1=1 in the first clause and x2=1 in the second clause. Therefore, the 3-SAT formula is also satisfied. Hence NAE 4-SAT problem is NP-Complete.

Problem 2:

NAE-3-SAT: Prove NAE-3-SAT is NP-complete.

Solution:

In order to prove NAE-3-SAT is NP Hard, reduce a known NP Hard problem to this problem, in our case 3-SAT to NAE-3-SAT.

The 3-SAT formula f is transformed by adding a variable v to every clause of the formula f.

For the pedantic's sake, we first have a polynomial reduction $3SAT \leq_q NAE-4-SAT$

As a last step we reduce $NAE-4-SAT \leq_q NAE-3-SAT$. Take an instance in NAE−4SAT and map the clauses as follows:

$(a \vee b \vee c \vee d) \mapsto (s \vee a \vee b) \wedge (\neg s \vee c \vee d)$ All the same as before, concatenate the result once more. Notice here that if the true and false variable's (one of each must exist) are mapped to the same clause, s can be choose appropriately. If they are mapped to different clauses, s can be choose opposite to the respective variable value in each pair.

To summarize: $3SAT \leq_p NAT-4-SAT \leq_p NAE-3-SAT$.

Problem 3:

4-SAT: Prove 4-SAT is NP-complete.

Solution:

Obviously, the 4SAT problem is an NP problem, an exponential order of all possible assignments in total. Now reduce the 3SAT to the 4SAT to prove that the 4SAT is NP complete problem:

Given an instance of $3SAT$ and converts any clause in I $(a1 \vee a2 \vee a3)$ to $(a1 \vee a2 \vee a3 \vee y) \vee (a1 \vee a2 \vee a3 \vee \neg y)$, we note that the resulting 4SAT instance is I', and the conversion from I to I' is obviously polynomial-time.

Next, it is proved that I and I' are equivalent:

$$I \quad \rightarrow \quad I'$$

If $(a1 \vee a2 \vee a3)$ satisfies, then $(a1 \vee a2 \vee a3 \vee y)$and $(a1 \vee a2 \vee a3 \vee \neg y)$ are true, i.e., $(a1 \vee a2 \vee a3 \vee y) \vee (a1 \vee a2 \vee a3 \vee \neg y)$.

$$I' \quad \rightarrow \quad I$$

Conversely, if $(A1 \vee A2 \vee A3 \vee Y) \vee (A1 \vee A2 \vee A3 \vee \neg Y)$ satisfies, i.e., $(A1 \vee A2 \vee A3 \vee Y)$ and $A1 \vee A2 \vee A3 \vee \neg Y)$(are both true, if Y is true then $(A1 \vee A2 \vee A3)$(must be true to guarantee that $(A1 \vee A2 \vee A3 \vee \neg Y)$ is true, and in the same way, if Y is false, then $(A1 \vee A2 \vee A3)$ must be true to guarantee that $(A1 \vee A2 \vee A3 \vee \neg Y)$ is true, so $(A1 \vee A2 \vee A3)$ is satisfied.

finally!

It can be seen that I and I' are equivalent, i.e. any instance of the 3SAT can be converted into an equivalent instance of the 4SAT. Since the 3SAT is a NP perfect question, the 4SAT is also NP perfect question.

Problem 4:

Feedback Vertex Set problem Let $G = (V, E)$ be a directed graph. A set $F \subseteq V$ is a *feedback vertex set* if every cycle of G contains at least one vertex from F. The *Feedback Vertex Set problem* asks whether G has a feedback vertex set with at most K vertices. Show that Feedback Vertex Set is NP-complete. (Hint: use Vertex Cover problem for reduction.)

Vertex Cover Problem

Given an undirected graph $G' = (V', E')$ and a number K', the Vertex Cover problem asks if there is a vertex cover of size at most K', i.e., a subset of vertices $V'' \subseteq V'$ such that every edge in E' is incident to at least one vertex in V''.

Solution:

Obviously, the Feedback Vertex Set problem is an NP problem, an exponential order of all possible assignments in total.

To demonstrate that the Feedback Vertex Set (FVS) problem is NP-complete, we can establish its NP-hardness by reducing the Vertex Cover problem to it. Here's the reduction:

Given an instance of the Vertex Cover problem with graph $G' = (V', E')$ and a positive integer K', construct an instance of the Feedback Vertex Set problem as follows:

1. Create a directed graph $G = (V, E)$ from the undirected graph G' by transforming each edge $(u, v) \in E'$ into a directed cycle (u, v) and (v, u). This transformation generates a cycle for each edge.

2. Set $K = K'$.

The transformation takes polynomial time relative to the size of the input.

Now, let's prove that the reduction satisfies the requirements for a valid reduction:

1. If G' has a vertex cover of size at most K': - Select the vertices corresponding to the vertices in the vertex cover as the feedback vertex set in G. Since each edge in G' is covered by at least one vertex in the vertex cover, every cycle in G will contain at least one vertex from the feedback vertex set. Thus, G has a feedback vertex set with at most K.

2. If G has a feedback vertex set with at most K: - Let F be the feedback vertex set in G. Consider F', which consists of the corresponding vertices in G'. Since every cycle in G contains at least one vertex from F, each cycle in G' must have at least one vertex from F'. Therefore, F' is a vertex cover of size at most K' in G'.

Thus, the reduction demonstrates that if we can solve the Feedback Vertex Set problem in polynomial time, we can solve the Vertex Cover problem in polynomial time as well. As the Vertex Cover problem is NP-complete, this implies that the Feedback Vertex Set problem is NP-hard. Since Feedback Vertex Set is also in NP, it is NP-complete.

Problem 5:

STINGY SAT is the following problem: given a set of clauses (each a disjunction of literals) and an integer k, find a satisfying assignment in which at most k variables a true, if such an assignment exists. Prove that STINGY SAT is NP-complete.

Solution:

To prove that STINGY SAT is an NP-complete (NPC) problem, we need to show that it is in NP and that SAT can be reduced to STINGY SAT.

1. Proving STINGY SAT is in NP:

STINGY SAT is an extension of the SAT problem. Let F be a SAT instance with k variables, and (F, k) is an instance of STINGY SAT. Since we can verify in polynomial time whether a given assignment satisfies F, we can also verify in polynomial time whether an assignment satisfies (F, k). Therefore, STINGY SAT is in NP.

2. Reducing SAT to STINGY SAT:

To reduce SAT to STINGY SAT, we need to show that SAT is satisfied if and only if STINGY SAT is satisfied. In the instance described in the previous section, we need to prove that an assignment x is a solution for F if and only if it is a solution for (F, k).

Assume x is a solution for F. Since F has k variables, there can be at most k variables set to true in x. Therefore, (F, k) is also satisfied because x satisfies both F and has fewer than or equal to k variables set to true.

Conversely, if x is a solution for (F, k), it means x makes F true, and x has fewer than or equal to k variables set to true. Therefore, x is a solution for F as well.

Since SAT is an NP-complete problem, we have shown that STINGY SAT is also NP-complete by reducing SAT to STINGY SAT.

Problem 6:

Given a tree T, a set of terminal vertices, and an integer k, is there a set of at most k edges which, when removed from T, separates every pair of terminal vertices? Prove that this problem is NP-complete.

Solution:

Given a tree $T = (V, E)$, a set S of terminal vertices where $S \subseteq V$, and an integer k, determine whether there exists a set $E' \subseteq E$ with $|E'| \leq k$ such that after removing edges in E' from T, every pair of terminal vertices in S is disconnected.

Proof of NP-Completeness:

To prove that this problem is NP-complete, we'll reduce the Hamiltonian Path problem to it.

Reduction:

Given an instance of the Hamiltonian Path problem with a graph $G = (V_G, E_G)$, we construct a tree T and a set of terminal vertices S as follows:

1. Construct a path-like tree T where each vertex $v \in V_G$ corresponds to a terminal vertex in S. 2. Connect these terminal vertices in the order that they appear in the Hamiltonian Path instance. 3. Assign an arbitrarily large value to k.

Claim:

The Hamiltonian Path exists in graph G if and only if there exists a set $E' \subseteq E(T)$ with $|E'| \leq k$ such that after removing edges in E' from T, every pair of terminal vertices in S is disconnected.

Proof:

- **If there exists a Hamiltonian Path in G:**

If a Hamiltonian Path exists in G, then the corresponding set of edges in tree T connecting the terminal vertices forms a path in T. Removing any edges from this path will disconnect at least one pair of terminal vertices, as the path is contiguous. Therefore, no set of at most k edges can disconnect every pair of terminal vertices in S.

- **If there does not exist a Hamiltonian Path in G:**

If there's no Hamiltonian Path in G, then there's no contiguous path among terminal vertices in T. In this case, removing edges in T to disconnect every pair of terminal vertices is equivalent to removing edges to break the non-existent path. Thus, a set of at most k edges can disconnect every pair of terminal vertices in S.

Therefore, since the Hamiltonian Path problem is NP-complete and we've shown that an instance of this problem reduces to our problem in polynomial time, the problem of determining whether a set of at most k edges separates every pair of terminal vertices in a given tree is NP-complete.