CS101 Algorithms and Data Structures
Fall 2023
Homework 12

Due date: 23:59, January 16th, 2024

1. Please write your solutions in English.

2. Submit your solutions to gradescope.com.

3. Set your FULL name to your Chinese name and your STUDENT ID correctly in Account Settings.

4. If you want to submit a handwritten version, scan it clearly. `CamScanner` is recommended.

5. When submitting, match your solutions to the problems correctly.

6. No late submission will be accepted.

7. Violations to any of the above may result in zero points.

8. You are recommended to finish this homework with LaTeX.

1. (0 points) Tutorial on how to prove that a particular problem is in NP-Complete
To prove problem A is in NP-Complete, your answer should include:

1. Prove that problem A is in NP by showing:

   (a) What your polynomial-size certificate is.
   (b) What your polynomial-time certifier is.

2. Choose a problem B in NP-Complete to reduce from.

3. Construct your polynomial-time many-one reduction f that maps instances of problem A to instances of problem B.

   (polynomial-time many-one reduction = polynomial transformation = Karp reduction, see presenter notes of page 7 & 61 in lecture slides (.pptx file) for more details.)

4. Prove the correctness of your reduction (i.e. Prove that your reduction f do map yes-instance of problem A to yes-instance of problem B and map no-instance of problem A to no-instance of problem B) by showing:

   (a) x is a yes-instance of problem $A \Rightarrow f(x)$ is a yes-instance of problem B.
   (b) x is a yes-instance of problem $A \Leftarrow f(x)$ is a yes-instance of problem B.

   (The statement above is the contrapositive of the statement "x is a no-instance of problem $A \Rightarrow f(x)$ is a no-instance of problem B.". These two statements are logically equivalent, but the one listed above would be much eaiser to prove.)

Proof Example

Prove that the decision version of Set-Cover is in NP-Complete. Recall that the yes-instances of the decision version of Set-Cover is:

$$\text{Set-Cover} = \left\{ \langle U, S_1, \ldots, S_n, k \rangle \;\middle|\; \begin{array}{l} n \in \mathbb{Z}^+, S_1, \ldots, S_n \subseteq U \text{ and there exist } k \text{ sets } S_{i_1}, \ldots, \\ S_{i_k} \text{that cover all of } U, \text{ i.e., } S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k} = U \end{array} \right\}$$

1. Our certificate and certifier for Set-Cover goes as follows:

   (a) A set of indices $\{i_1, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$, whose size is polynomial of input size .
   (b) Check whether $S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k} = U$, whose run-time is polynomial of input size.

2. We choose the decision version of Vertex-Cover to reduce from. Recall that the yes-instances of the decision version of Vertex-Cover is:

$$\text{Vertex-Cover} = \left\{ \langle G, k' \rangle \;\middle|\; \begin{array}{l} G \text{ is an undirected graph and there exists a set of} \\ k' \text{ vertices that touches all edges in } G. \end{array} \right\}$$

   (We use k' here because k has already appeared before.)

3. Given an undirected graph $G = (V, E)$ and an postive integer $k' \in \mathbb{Z}^+$, we construct $f(\langle G, k' \rangle) = \langle U, S_1, \ldots, S_n, k \rangle$ as follows:

   (a) $U = E$, which represents the edges from the graph.
   (b) Define $m = |V|$ and let $n = m$, which means the number of sets equals the number of vertices in G.

(c) Label elements in V as $V = \{v_1, v_2, \ldots, v_m\}$. For each $i \in \{1, \ldots, m\}$, the set S_i is defined as $S_i = \{e \in E \mid e = (v_i, u) \text{ for some } u \in V \setminus \{v\}\}$. In other word, S_i is the set of edges incident to v_i.

(d) $k = k'$.

Our reduction takes polynomial time because:

(a) Generating each S_i just takes polynomial time since each edge is visited twice (once for each endpoint).

(b) Generating U and k trivially takes polynomial time because they are copied directly from the input.

4. Then we prove the correctness of our reduction as follows:

(a) "$\Rightarrow$": Let $\langle G, k' \rangle$ be a yes-instance of Vertex-Cover and let $V^* = \left\{ v_{i_1}, \ldots, v_{i_{k'}} \right\}$ be the choice of k' vertices that form the vertex cover. Then for each $e \in E$, there is some v_{i_j} ($j \in [k']$) that is an endpoint of e, which directly translates to for each $e \in U$, there is some S_{i_j} ($j \in [k']$) containing e by our construction of f. Hence, we claim that the sets $S_{i_1}, \ldots, S_{i_{k'}}$ form a set cover of size $k = k'$ for U.

(b) "$\Leftarrow$": Let $\langle U, S_1, \ldots, S_m, k \rangle$ be a yes-instance of Set-Cover and let $\{S_{i_1}, \ldots, S_{i_k}\}$ be the choice of k sets that form the set cover. Then for each $e \in U$, there is some S_{i_j} ($j \in [k]$) that contains e, which directly translates to for each $e \in E$, there is some v_{i_j} ($j \in [k]$) that is an endpoint of e. Hence, we claim that the sets $\{v_{i_1}, \ldots, v_{i_k}\}$ form a vertex cover of size $k' = k$ for G.

Hence, the decision version of Set-Cover is in NP-Complete.

2. (18 points) Multiple / Single Choice(s)

This part consists of multiple choices questions ((a)-(d)) and single choice questions ((e)-(g)).

For each multiple choices question ((a)-(d)), there may be **one or more** correct choice(s). Select all the correct answer(s). For each such question, you will get 0 points if you select **any** wrong choice, but you will get 1 point if you select a non-empty subset of the correct choices.

For each single choice question ((e)-(g)), there is **exactly one** correct choice.

Write your answers in the following table.

2(a)	2(b)	2(c)	2(d)	2(e)	2(f)	2(g)
B	BC	AB	BD	D	A	B

(a) (3') A problem in NP is in NP-Complete if:

    A. It can be reduced to any other NP problem in polynomial time.

    B. Any other NP problem can be reduced to it in polynomial time.

    C. It can be reduced to another NP-Complete problem in polynomial time.

    D. There exists another NP-Complete problem which can be reduced to it in polynomial time.

(b) (3') Given two decision problems A and B such that there exists a polynomial-time many-one reduction from A to B. Which of the following statements must be true?

    A. $A \in P \implies B \in P$

    B. $A \in$ NP-Complete $\implies B \in$ NP-Complete.

    C. $B \in P \implies A \in P$.

    D. $B \in$ NP-Complete $\implies A \in$ NP-Complete.

(c) (3') Which of the following statements are true?

    A. The "N" in NP stands for "nondeterministic", instead of "not".

    B. According to Cook-Levin Theorem, any problem in NP can be reduced to Circuit-SAT in polynomial time.

    C. k-SAT $\in$ NP-Complete for any positive integer $k \geq 2$.

    D. Consider the optimization version of Knapsack problem with $n \in \mathbb{Z}^+$ items and $W \in \mathbb{Z}^+$ where the weight and value of each item is $w_i \in \mathbb{Z}^+$ and $v_i \in \mathbb{R}^+$ respectively. Since there is a dynamic programming algorithm for this problem that runs in $O(nW)$ time, we may deduce that we can solve this problem in polynomial time.

(d) (3') Which of the following statements are true?

    A. If you find a polynomial time algorithm for a problem in NP, then you have proved $P = NP$.

    B. If you prove that 4-Color can be reduced to 3-Color in polynomial time, then you've proved $P = NP$.

    C. $P \neq NP$ if and only if $P \cap$ NP-Complete $= \emptyset$.

    D. $P = NP$ if and only if $NP =$ NP-Complete.

Hint: In fact, any two problems in P reduces to each other in polynomial time. Intuitively, you may interpret this as the fact that all the questions in P share the same "hardness". A more formal explanation goes as follows:

For simplicity, we only consider those **decision** problem in P here. Let A and B be any two decision problems in P and we want to show that A can be reduced to B in polynomial time. Given an instance x of A, our reduction goes as follows:

1. Prepare two copies of data containing a yes-instance of B and a no-instance of B respectively, which can be done in polynomial time since $B \in P$.

2. Determine whether x is a yes-instance of A or not, which can be done in polynomial time since $A \in P$.

3. If the answer is yes, then we return the copy containing a yes-instance of B. Otherwise, we return the copy containing a no-instance of B.

More specifically, let's see how this idea works in the case where:

1. A is "Given a undirected weighted graph $G = (V, E, \langle w_e \,|\, e \in E \rangle)$ and $c \in \mathbb{R}$, does the minimum spanning tree of it have cost no more than c?"

2. B is "Given a undirected weighted graph $G' = (V', E', \langle w'_e \,|\, e \in E' \rangle)$ with no negative-cost cycles and $c' \in \mathbb{R}$, does the shortest path between any pairs of vertices of G' have cost no more than c'?"

Given a undirected weighted graph G (an instance of A):

1. Prepare two graphs G_1 and G_2 where G_1 satisfies the shortest path between any pairs of vertices of G_1 has cost no more than c' and G_2 satisfies there exists a pair of vertices of G_2 such that the shortest path between them has cost more than c'.

2. Find the minimum spanning tree of G with Kruskal's algorithm and compare the answer with c.

3. If the answer is no more than c, then we return G_1. Otherwise, we return G_2.

(e) (2') The statement "Minimum-Spanning-Tree is not in NP-Complete." is _____?

    A. True regardless of whether P equals to NP or not.

    B. False regardless of whether P equals to NP or not.

    C. True if and only if $P = NP$.

    D. True if and only if $P \neq NP$.

(f) (2') The statement "If problem B is in NP, then for any problem $A \in P$, A can be reduced to B in polynomial time." is _____?

    A. True regardless of whether P equals to NP or not.

    B. False regardless of whether P equals to NP or not.

    C. True if and only if $P = NP$.

    D. True if and only if $P \neq NP$.

(g) (2') The statement "There are problems in NP that cannot be solved in exponential time." is _____?

    A. True regardless of whether P equals to NP or not.

    B. False regardless of whether P equals to NP or not.

    C. True if and only if $P = NP$.

    D. True if and only if $P \neq NP$.

3. (9 points) Reductions

In this question, you are required to construct 3 **correct** direct polynomial-time many-one reduction that respectively:

1. maps instances of Independent-Set to instances of Clique.

2. maps instances of Subset-Sum to instances of Knapsack (decision version).

3. maps instances of 3-color to instances of k-color (k is a **given** positive integer and $k \geq 4$).

In this question, you are **not** required to prove the correctness of your reductions and it suffices to write your **reduction only**. However, make sure your reduction is correct to receive points.

Reminder: Don't forget state that your reduction takes polynomial time!

(a) (3') Consider the following problems:

1.Independet-Set: Given a undirected graph $G = (V, E)$ and a positive integer k, determine whether there exists a subset k (or more) vertices of V such that no two of them are adjacent (i.e. does G contains an independent set of size at least k).

The yes-instances of Independet-Set is:

$$\text{Independet-Set} = \left\{ \langle G, k \rangle \;\middle|\; \begin{array}{l} G = (V, E) \text{ is an undirected graph that contains} \\ k \text{ vertices with no edges between them.} \end{array} \right\}$$

2. Clique: Given a undirected graph $G' = (V', E')$ and a positive integer k', determine whether there exists a subset k' (or more) vertices of V' such that **any** two of them are adjacent (i.e. does G' contains a clique of size at least k').

The yes-instances of Clique is:

$$\text{Clique} = \left\{ \langle G', k' \rangle \;\middle|\; \begin{array}{l} G' = (V', E') \text{ is an undirected graph that contains } k' \\ \text{vertices such that they are connected to each other.} \end{array} \right\}$$

Construct a **correct** direct polynomial-time many-one reduction f_1 that maps instances of Independent-Set to instances of Clique.

Solution:

Given an undirected graph $G' = (V', E')$ and an postive integer $k' \in \mathbb{Z}^+$, we construct $f(\langle G, k \rangle)$ as follows:

1. Define $m = |V|$. Construct a complete graph G'' with m vertices.

2. Let $G = G'' - G'$, which means removing edges in G'.

3. Find Independent-Set in G. These sets will construct Clique in G'. Thus $k' = k$.

Our reduction takes polynomial time because:

1. Constructing a graph G'' costs polynomial time.

2. Removing edges take polynomial time since finding the vertex to remove costs polynomial time.

6

(b) (3') Consider the following problems:

3.Subset-Sum: Given an array $A = [a_1, a_2, ..., a_m]$ of positive integers and a positive integer k such that $k \leq \sum_{i \in [m]} a_i$, determine whether there exists a subset $S \subseteq [m]$ such that $\sum_{i \in S} a_i = k$ (i.e. determine whether there exists a subset of A such that the sum of its elements is k).

The yes-instances of Subset-Sum is:

$$\text{Subset-Sum} = \left\{ \langle a_1, a_2, \ldots, a_m, k \rangle \;\middle|\; \begin{array}{c} m \in \mathbb{Z}^+, a_1, \ldots, a_m, k \in \mathbb{Z}^+ \text{ and there} \\ \text{exists a subset of the } a_i\text{'s that sum up} \\ \text{to k, i.e. } \exists\, S \subseteq [m] : \sum_{i \in S} a_i = k. \end{array} \right\}$$

4.Knapsack: Given $n \in \mathbb{Z}^+$ items where the weight and value of each item is $w_i \in \mathbb{Z}^+$ and $v_i \in \mathbb{R}^+$ respectively as well as fixed $W \in \mathbb{Z}^+$ and $V \in \mathbb{R}^+$, determine whether there exists a subset $P \subseteq [n]$ such that $\sum_{i \in P} w_i \leq W$ and $\sum_{i \in P} v_i \geq V$ (i.e. determine whether there exists a subset of items such that the sum of their weights is no more than W while the sum of their values is no less than V).

The yes-instances of Knapsack is:

$$\text{Knapsack} = \left\{ \langle w_1, \ldots, w_n, v_1, \ldots, v_n, W, V \rangle \;\middle|\; \begin{array}{c} n, w_1, \ldots, w_n, v_1, \ldots, v_n, V, W \in \mathbb{Z}^+ \\ \text{and there exists a subset } P \subseteq [n] \\ \text{such that } \sum_{i \in P} w_i \leq W \text{ and } \sum_{i \in P} v_i \geq V. \end{array} \right\}$$

Construct a **correct** direct polynomial-time many-one reduction f_2 that maps instances of Subset-Sum to instances of Knapsack.

> **Solution:**

(c) (3') Consider the following problems:

5.3-Color: Given a undirected graph $G = (V, E)$, determine whether its vertices can be colored **within** 3 different colors such that no adjacent nodes have the same color?

The yes-instances of 3-Color is:

$$3\text{-Color} = \left\{ \langle G \rangle \,\middle|\, \begin{array}{l} G = (V, E) \text{ is an undirected graph such that its} \\ \text{vertices can be colored \textbf{within} 3 different colors} \\ \text{such that no adjacent nodes have the same color.} \end{array} \right\}$$

6.k-Color: For fixed positive integer $k \geq 4$, given a undirected graph $G' = (V', E')$, determine whether its vertices can be colored **within** k different colors such that no adjacent nodes have the same color?

The yes-instances of k-Color is:

$$k\text{-Color} = \left\{ \langle G' \rangle \,\middle|\, \begin{array}{l} G' = (V', E') \text{ is an undirected graph such that its} \\ \text{vertices can be colored \textbf{within} k different colors} \\ \text{such that no adjacent nodes have the same color.} \end{array} \right\}$$

Construct a **correct** direct polynomial-time many-one reduction f_3 that maps instances of 3-Color to instances of k-Color.

Reminder: You may **not** apply mathematical induction for k here. A direct reduction is required.

> **Solution:** Given three color graph G, we construct k color graph G' as follows:
> For each vertex in G, we add $|V| - 3$ vertices with different colors linked to it. Thus we construct G'.
> Our reduction takes polynomial time because:
> Every vertex will be visited once and add $|V| - 3$ vertices to it, thus it's polynomial time.

4. (8 points) Equivalent-Partition **is in** NP-Complete

In this question, we will prove that Equivalent-Partition is in NP-Complete.

Equivalent-Partition: Given an array $B = [b_1, b_2, ..., b_n]$ of non-negative integers, determine whether there exists a subset $T \subseteq [n]$ such that $\sum_{i \in T} b_i = \sum_{j \in [n] \setminus T} b_j$ (i.e. determine whether there is a way to partition B into two disjoint subsets such that the sum of the elements in each subset is equivalent).

The yes-instances of Equivalent-Partition is:

$$\text{Equivalent-Partition} = \left\{ \langle b_1, \ldots, b_n \rangle \; \middle| \; \begin{array}{l} n \in \mathbb{Z}^+, b_1, \ldots, b_n \in \mathbb{N} \text{ and there exists a} \\ \text{partition of the } b_i\text{'s to two parts whose sums} \\ \text{are equivalent, i.e. } \exists\, T \subseteq [n] : \sum_{i \in T} b_i = \sum_{j \in [n] \setminus T} b_j \end{array} \right\}$$

Based on the tutorial on page 2 and 3, our proof goes as follows:

(a) (2') Prove that Equivalent-Partition is in NP. (Show your certificate and certifier.)

> **Solution:**
> Our certificate and certifier for Set-Cover goes as follows:
>
> 1. A subset $T \subseteq [n]$ and its complementary set, whose size is less than n, thus is polynomial.
>
> 2. Check whether $\sum_{i \in T} b_i = \sum_{j \in [n] \setminus T} b_j$, whose run time is polynomial.

(b) (0') We choose Subset-Sum to reduce from. Recall that the yes-instance of Subset-Sum is:

$$\text{Subset-Sum} = \left\{ \langle a_1, a_2, \ldots, a_m, k \rangle \; \middle| \; \begin{array}{l} m \in \mathbb{Z}^+, a_1, \ldots, a_m, k \in \mathbb{Z}^+ \text{ and there} \\ \text{exists a subset of the } a_i\text{'s that sum up} \\ \text{to k, i.e. } \exists\, S \subseteq [m] : \sum_{i \in S} a_i = k. \end{array} \right\}$$

(c) Construct your polynomial-time many-one reduction f that maps instances of Subset-Sum to instances of Equivalent-Partition.

   i. (0') Vixbob proposed a reduction as follows:
   Let $n = m$ and $b_i = a_i$ for $\forall\, i \in [m]$. Finally set $k = \frac{1}{2} \sum_{i \in [m]} a_i$. In this way, $\langle a_1, a_2, \ldots, a_m, k \rangle$ is a yes-instance of Subset-Sum if and only if $\langle b_1, \ldots, b_n \rangle = \langle a_1, a_2, \ldots, a_m \rangle$ is a yes-instance of Equivalent-Partition.
   However, this reduction is **wrong**. Why?

   > **Solution:**
   > Here k is given (fixed) since it's part of the instance of the problem that we want to reduce from (i.e. it's part of the input of your reduction). Thus, you **can't** arbitrarily modify the value of k.

ii. (0') GKxx proposed another reduction as follows:
Define $X = \sum_{i \in [m]} a_i$ and let $n = m + 2$. Then we define our reduction as:

$$\langle b_1, \ldots, b_n \rangle = f(\langle a_1, a_2, \ldots, a_m, k \rangle) \overset{\Delta}{=} \langle a_1, a_2, \ldots, a_m, k, X - k \rangle$$

In this way, we may deduce that $\langle a_1, a_2, \ldots, a_m, k \rangle$ is a *yes*-instance of Subset-Sum if and only if $\langle b_1, \ldots, b_n \rangle = \langle a_1, a_2, \ldots, a_m, k, X - k \rangle$ is a *yes*-instance of Equivalent-Partition because a subset with sum k can be paired with $X - k$ and the remaining subset with sum $X - k$ can be paired with k, resulting in an equivalent partition. However, this reduction is **wrong** again. Why?

> **Solution:**
> This reduction is not a valid one since the sequence $\langle a_1, a_2, \ldots, a_m, k, X - k \rangle$ **always** has a trivial equivalent partition $\langle a_1, a_2, \ldots, a_m \rangle$ versus $\langle k, X - k \rangle$, which indicates that $f(\langle a_1, a_2, \ldots, a_m, k \rangle)$ is **always** a *yes*-instance of Equivalent-Partition regardless of whether $\langle a_1, a_2, \ldots, a_m, k \rangle$ is a *yes*-instance of Subset-Sum or not.

iii. (3') What's your **correct** polynomial-time many-one reduction f that maps instances of Subset-Sum to instances of Equivalent-Partition?
Hint: GKxx's reduction is really close to a correct one. Maybe you can modify it a little bit to make it correct?

> **Solution:**
> Define $X = \sum_{i \in [m]} a_i$ and let $n = m + 1$. Then we define our reduction as:
>
> $$\langle b_1, \ldots, b_n \rangle = f(\langle a_1, a_2, \ldots, a_m, k \rangle) \overset{\Delta}{=} \langle a_1, a_2, \ldots, a_m, 2k - X \rangle$$
>
> In this way, we may deduce that $\langle a_1, a_2, \ldots, a_m, k \rangle$ is a *yes*-instance of Subset-Sum if and only if $\langle b_1, \ldots, b_n \rangle = \langle a_1, a_2, \ldots, a_m, 2k - X \rangle$ is a *yes*-instance of Equivalent-Partition because a subset has sum k and the remaining subset also has sum k, resulting in an equivalent partition.

(d) Prove the correctness of your reduction by showing:

   i. (1') x is a yes-instance of Subset-Sum $\Rightarrow$ $f(x)$ is a yes-instance of Equivalent-Partition.

> **Solution:**

   ii. (2') x is a yes-instance of Subset-Sum $\Leftarrow$ $f(x)$ is a yes-instance of Equivalent-Partition.

> **Solution:**