CPSC 320 2018W1: Assignment 5

November 19, 2018

Please submit this assignment via GradeScope at `https://gradescope.com`. Be sure to identify everyone in your group if you're making a group submission. (Reminder: groups can include a maximum of three students; we strongly encourage groups of two.)

Submit by the deadline **Thursday November 29, 2018 at 10PM**. For credit, your group must make a **single** submission via one group member's account, marking all other group members in that submission **using GradeScope's interface**. Your group's submission **must**:

- Be on time.
- Consist of a single, clearly legible file uploadable to GradeScope with clearly indicated solutions to the problems. (PDFs produced via LaTeX, Word, Google Docs, or other editing software work well. Scanned documents will likely work well. **High-quality** photographs are OK if we agree they're legible.)
- Include prominent numbering that corresponds to the numbering used in this assignment handout (not the individual quizzes). Put these **in order** starting each problem on a new page, ideally. If not, **very clearly** and prominently indicate which problem is answered where! When uploading assignments to gradescope, marks will be docked if pages are not properly assigned to each question.
- Include at the start of the document the **ugrad.cs.ubc.ca e-mail addresses** of each member of your team. (Please do **NOT** include your name on the assignment, however.[1])
- Include at the start of the document the statement: "All group members have read and followed the guidelines for academic conduct in CPSC 320. As part of those rules, when collaborating with anyone outside my group, (1) I and my collaborators took no record but names (and GradeScope information) away, and (2) after a suitable break, my group created the assignment I am submitting without help from anyone other than the course staff." (Go read those guidelines!)
- Include at the start of the document your outside-group collaborators' ugrad.cs.ubc.ca IDs, but **not** their names. (Be sure to get those IDs when you collaborate!)

Before we begin, a few notes on pseudocode throughout CPSC 320: Your pseudocode need not compile in any language, but it must communicate your algorithm clearly, concisely, correctly, and without irrelevant detail. Reasonable use of plain English is fine in such pseudocode. You should envision your audience as a capable CPSC 320 student unfamiliar with the problem you are solving. If you choose to use actual code, note that you may **neither** include what we consider to be irrelevant detail **nor** assume that we understand the particular language you chose. (So, for example, do not write `#include <iostream>` at the start of your pseudocode, and avoid idiosyncratic features of your language like Java's ternary (question-mark-colon) operator.)

[1] If you don't mind private information being stored outside Canada and want an extra double-check on your identity, include your student number rather than your name.

1 NP true or false

Let X and X' be problems in NP. State whether you think each of the following statements is true, false, or an open question. Give a short explanation of your answer.

1. The problem $X \cup X'$ is always in NP.
2. If $X \leq_p X'$ and X is in P, then X' must be in P.
3. If both $X \leq_p X'$ and $X' \leq_p X$, then both X and X' must be NP-complete.

2 Scheduling classes

A college wishes to offer evening courses and has n possible courses that can be offered. There are k students interested in taking a course and each student is interested in some subset of the n possible courses. To manage costs, the college has a bound b on the number of courses it is willing to actually offer. Is there a way to choose b courses out of the n possibilities so that every student can take a course that interests them? Formally, an instance of the CS problem consists of

- a set C of n possible courses, numbered 1 through n,
- subsets $S_1, S_2, \ldots, S_k$ of $\{1, 2, \ldots, n\}$, (the courses of interest to each of the students), and
- a bound b (the maximum number of classes actually offered).

The problem is to determine whether there is a subset B of $\{1, 2, \ldots, n\}$, where the size of B is $\leq b$, such that $B \cap S_i$ is not empty for all i, $1 \leq i \leq k$.

1. Describe an efficient reduction from the Vertex Cover problem to the Class Scheduling (CS) problem.
2. Your reduction maps instances (G, p) of Vertex Cover to instances $(C, S_1, \ldots, S_k, b)$ of CS. Show that G has a vertex cover of size at most p if and only if in the mapped instance $(C, S_1, \ldots, S_k, b)$, there is a set of courses of size at most b such that every student can take a course that interests them. (Remember that you need to two parts here, one for "if" and one for "only if".)
3. Explain why your reduction runs in polynomial time.
4. Explain why CS is *NP*-complete.

3 Connected dominating set

In quiz 5 we described the problem of finding a *connected dominating set* with at most K vertices in a graph $G = (V, E)$ (a connected dominating set is a dominating set W where the elements of W, together with any edge of G that connects two of them, form a connected subgraph of G).

In the solutions we described the following reduction from the vertex cover problem to the connected dominating set problem.

> Given an instance (G, K) of the vertex cover problem, construct a graph G' by first creating a copy of G. Next, add a new vertex v^*, and add the edges $\{v^*, v\}$ for each $v \in V$. Then, add one vertex v_e for each edge $e \in E$, and connecting this vertex to the endpoints of e. That is, if $e = \{x, y\}$ then add the edges $\{v_e, x\}$ and $\{v_e, y\}$. Finally, add two new vertices z and z' and add the two edges $\{v^*, z\}$ and $\{v^*, z'\}$. Map instance (G, K) to instance $(G', K + 1)$ of the connected dominating set problem.

1. Explain why this reduction is correct. That is, prove that the instance of the vertex cover problem has a Yes answer is and only if the instance of the connected dominating set problem produced by the reduction also has a Yes answer.

2. Using the fact you proved in part (1), prove that the maximum leaf spanning tree problem is NP-complete.