

Problem Set 7

Harvard SEAS - Fall 2023

Due: Wed Nov. 15, 2023 (11:59pm)

Your name:**Collaborators:****No. of late days used on previous psets:****No. of late days used after including this pset:**

The purpose of this problem set is to develop skills in implementing graph algorithms, appreciate the impact of different kinds of worst-case exponential algorithms in practice, and practice reducing problems to SAT.

1. (Another coloring algorithm) In the [Github repository for PS7](#), we have given you basic data structures for graphs (in adjacency list representation) and colorings, an implementation of the coloring algorithm from ps5, and a variety of test cases (graphs) for coloring algorithms. For Windows users, use this [Google Colab](#) file to run your code.
 - (a) Implement the reduction from 3-coloring to SAT given in class in the function `sat_3_coloring`, producing an input that can be fed into the SAT Solver [Glucose](#), and verify its correctness by running `python3 -m ps7_tests 3`.
 - (b) Compare the efficiency of Exhaustive-Search 3-coloring, the $O(1.89^n)$ -time BFS-based algorithm for 3-coloring from problem set 5 (feel free to use the staff solution or your own implementations from problem set 5), and your implementation from Part 1a using `ps7_experiments`. In the experiments file, we've provided code to generate two types of graphs (lines of rings and clusters of independent sets) and some new hard graph instances. For each of those types of graphs, how many of the given instances, if any, can each algorithm solve within 10 seconds (same time limit as problem set 5)? You should fill out the table and briefly discuss and try to explain your findings.

Algorithm	Exhaustive	ISet BFS	SAT Color
# Solvable Ring Instances			
# Solvable Cluster Instances			
# Solvable Hard Graphs			

- (c) (optional¹) Find a graph G such that Glucose takes more than 1 second to solve the SAT instance to which the 3-colorability of G was reduced in part a, and n is as small as you can make it. Describe your approach to finding such a G .
2. (2SAT) Consider the following Boolean statements.
 - (a) $A :=$ "Alice's grandparents got married."

¹This problem is meant to be done based on your enjoyment/interest and only if you have time. It won't make a difference between N, L, R-, and R grades, and course staff will deprioritize questions about this problem at office hours and on Ed.

- (b) $B :=$ “Alice was born.”
- (c) $C :=$ “Alice created a time machine in order to travel to the past.”
- (d) $D :=$ “Alice landed the time machine at her grandparents’ first date and ruined it.”

Consider the CNF formula

$$(\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg B \vee C) \wedge (B \vee \neg C) \wedge (\neg C \vee D) \wedge (C \vee \neg D) \wedge (\neg D \vee \neg A) \wedge (D \vee A).$$

If you interpret each clause above as an implication, are the implications reasonable? Explain briefly for 3 clauses of your choice. Draw the implication graph (for the whole formula, not just your chosen 3 clauses) and explain graphically why the CNF is unsatisfiable.

3. (Reductions to SAT) Consider the following problem. From Harvard’s n CS concentrators (e.g. $n = 400$), we want to form a team of exactly k students (e.g. $k = 30$) to represent Harvard in a new programming competition. The programming competition problems may require expertise in any of m different programming languages (e.g. $m = 100$). But each of the CS concentrators only knows a few different programming languages, with a different set per person. So we want to try to find k Harvard CS concentrators such that between them, they know all m languages. Formally, we want to solve the following computational problem:

Input : A finite set $L = \{\ell_0, \dots, \ell_{m-1}\}$ of programming languages; a finite set $S = \{s_0, \dots, s_{n-1}\}$ of students; for each student $s \in S$, a set $K(s) \subseteq L$ of languages that student s knows; and a team size $k \in \mathbb{N}$

Output : A team $T \subseteq S$ of size k that collectively knows all of the programming languages in L (i.e. $\bigcup_{s \in T} K(s) = L$), if one exists

Computational Problem ProgrammingTeam

- (a) Show that ProgrammingTeam can be efficiently reduced to solving a SAT instance on kn variables and $m + O(kn^2)$ clauses. Prove the correctness of your reduction and analyze its runtime.
 - (b) (optional¹) Come up with a more efficient reduction that produces a SAT instance with $O(kn)$ variables and $m + O(kn)$ clauses (or even $m + O(n \log k)$ clauses). (Hint: something like $\psi_{n,k}$ or τ_ℓ formulas from the Section 7 problem on IndependentSet \leq SAT might be useful.)
4. Please fill out the Problem Set 7 Feedback Survey here: <https://forms.gle/bNfaHbXutBiXq97v6>. We appreciate your feedback!