2D Challenge 50.004: Introduction to Algorithms Sept-Dec 2020 term

1 Preamble

In the 50.004 course, you are learning how to devise, describe, and analyze algorithms for given problems. You have learnt about SAT solvers in 50.001. For the 50.004 component of the 2D challenge you need to implement a 2-SAT solver. A 2-SAT problem restricts formulas to have clauses with at most 2 literals. For more information on 2-SAT, please refer to https://en.wikipedia.org/wiki/2-satisfiability.

2 Tasks

- **Necessary:** Design and implement in Java, C, C++, Python, etc or pseudo-code an algorithm that solves the 2-SAT problem in *polynomial time* in the number of variables and clauses. A suggestion is to use an algorithm based on analyzing the strongly connected components of the implication graph, as described in Wikipedia (by using DFS) in https://en.wikipedia.org/wiki/2-satisfiability. You can also use this method to find some satisfying solution. *Algorithms that run in exponential time like 2^n (brute force approaches) will not get credit.*

  1. Your program must accept an input file in the CNF format as specified in https://app.box.com/s/opgbiv7310szl97nzv5devu3gi7qzbe3 (See Section 2.1).
  2. Your program must print the result of the satisfiability analysis: print "SATISFIABLE" or "UNSATISFIABLE".
  3. If the formula is satisfiable, your program must also print a solution. For example, if you are evaluating the formula (x1 or x2) and (x2 or x3) and (x3 or x4) and (not x1 or not x3) and (not x2 or not x4), and you found a set of values for the four variables that leads to the whole formula being true to be x1=false, x2=true, x3=true, x4=false, then you produce the output: 0 1 1 0.

- **Bonus:** Implement a randomizing algorithm to check for satisfiability as outlined in https://people.seas.harvard.edu/~cs125/fall14/lec19.pdf (lec19_bonus.pdf is attached). The goal is to run both algorithms on the same input and compare their results in terms of efficiency.

3 Deliverables

1. A report on your 2-SAT algorithm implementation and performance analysis. In the report you *should mention how* the 2-SAT problem can be solved in polynomial time using essentially DFS. Moreover, *figures* might be helpful in your explanation. Why this algorithm is not working for 3-SAT but only for 2-SAT?

2. Submission of the code. We will not test your code, but is crucial to code for you understanding. We will check it briefly.

3. Bonus: In case you implemented the randomized algorithm, a short report (1-2 extra pages) comparing the performance of the two algorithms on similar inputs. Is the randomizing algorithm a practical substitute for the deterministic one?

Deadline: Friday November 6, 2020 (11:59pm) via edimension