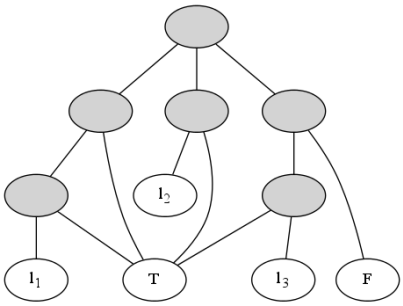
CPSC 320 Random Practice Problems

April 4, 2017

1 Graph Staining

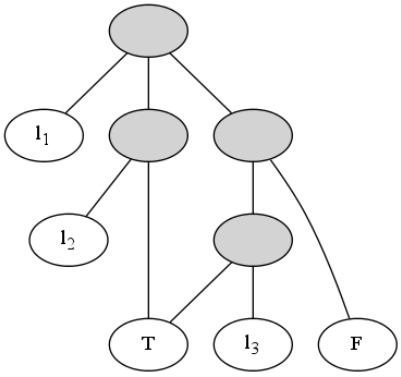
When reducing 3-SAT to graph 3-colouring (3-GC), we first establish one colour as "true" coloured, one as "false" coloured, and one as "base" coloured (neither true nor false) and ensure that each literal has a corresponding node that must take on either the true or false colour, while its negation takes on the opposite of these two colours.

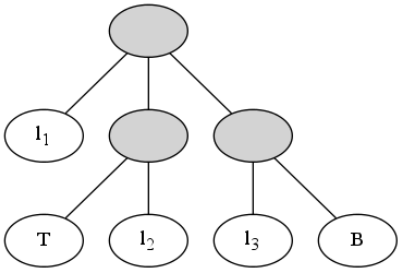
Then, to encode a clause, we connect the clause's three literals in this pattern:



The central goal of this pattern is to ensure that if the three literal nodes l_1, l_2, and l_3 are all false-coloured, then the second-to-highest row of nodes must be false-, base-, and true-coloured, respectively. Since the top node then has one neighbour of each colour, it cannot be coloured.

Given this goal, consider the following two simplified versions of the gadget. For each, either prove that it does work or give an instance of 3-SAT and its corresponding 3-GC instance on which it fails.





This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.

2 Taking Stock

You work for an algorithmic trading firm that is setting bounds on the maximum gains achievable over a particular historical period, for ML training purposes. You have second-by-second prices for seven stocks of interest. Under the constraint that you must always have your entire investment in **one** stock and that selling out of one stock and buying another costs 0.1% of the stock value, you want to find the series of second-by-second investments that maximizes profits. (When you sell a stock and buy another, you sell your entire stake in the first stock, paying 0.1% of the resulting value and then buy exactly the remaining value in the other stock. Both transactions use that second's price.)

Design an algorithm to solve this problem. Make reasonable assumptions where this specification is vague. (Do **NOT** skip our design steps like creating trivial and small instances, or things will go terribly wrong!)

3 Binary Besmirched

You are given a "sorted" array of n numbers and a target number t to search for. However, as many as $\lfloor \lg n \rfloor - 1$ entries in the array have become corrupted, meaning they may have any value. Give and analyse an efficient algorithm to find the target. If the target is not present **or** has been corrupted, you should simply indicate that the target is not in the array.

What if you were told instead that k entries had become corrupted, where $0 \leq k \leq n$?

This work is licensed under a Creative Commons Attribution 4.0 International License.
For license purposes, the author is the University of British Columbia.