QUANTUM ALGORITHMS
HOMEWORK 2 SELECTED SOLUTIONS

PROF. MATTHEW MOORE

3.1. Prove that one can check the satisfiability of a 2-CNF (a conjunction of disjunctions, each containing two literals) in polynomial time.

Solution: We will show that 2-CNF satisfiability reduces to a graph problem. Consider an instance of the 2-CNF satisfiability problem,

$$S = \bigwedge_{i=1}^{n} s_i \vee t_i, \qquad\qquad s_i, t_i \in \big\{x_j, \neg x_j \mid 1 \leq j \leq m\big\}.$$

Construct a directed graph $\mathbb{G}$ with a vertex for each variable x_j and a vertex for the negation of each variable $\neg x_j$. For each clause $s_i \vee t_i$ add edges $\neg s_i \to t_i$ and $\neg t_i \to s_i$. This graph is called the *implication graph* of S. Observe that S is satisfiable if and only if there are no paths of the form

$$x_i \to \cdots \to \neg x_i \to \cdots \to x_i.$$

The existence of paths such as this can be detected by computing the strongly connected components of $\mathbb{G}$ and checking whether x_i and $\neg x_i$ are ever in the same component. This can be done in linear time.

3.3. Suppose we have an NP-oracle — a magic device that can immediately solve any instance of the SAT problem for us. In other words, for any propositional formula the oracle tells whether it is satisfiable or not. Prove that there is a polynomial-time algorithm that finds a satisfying assignment to a given formula by making a polynomial number of queries to the oracle. (A similar statement is true for the Hamiltonian cycle: finding a Hamiltonian cycle in a graph is at most polynomially harder than checking for its existence.)

Solution: Consider the algorithm below.

```
define solve_SAT(S):
  if not is_satisfiable(S):
    return False

  for each variable x_i in S:
    if is_satisfiable(S ∧ x_i):
      assign "True" to x_i
      let S = S ∧ x_i
    elif is_satisfiable(S ∧ (¬x_i)):
      assign "False" to x_i
      let S = S ∧ (¬x_i)
    else:   # should never get here!
      return False
```

Date: February 18, 2021.

```
    return the variable assignments
```

There will be a satisfying assignment for S with x_i true if and only if $S \wedge x_i$ is satisfiable.

A common error was to somehow include literals when modifying S, or to fail to update S once a value of x_i has been found. It is possible for x_i to be true in some assignment and x_j to be true in some assignment, but for there to not be an assignment where *both* are true:

$$\big(x_i \vee x_j\big) \wedge \big(\neg x_i \vee \neg x_j\big).$$