Marcus Anderson

Homework 7

CS 6515: Introduction to Graduate Algorithms

A)

Algorithm:

In order find if our algorithm returns an assignment of variables so that each clause contains literals that are either all *true* or *false*, we will leverage the 2SAT algorithm. Using the CNF boolean formula input, *f*, we first need to split the literals in each clause to match the input of 2SAT. For each clause, we pair each literal with the opposite assignment of the literal directly to its right, pairing the last literal in the clause with the first one. This will be called *f′*. Next, we pass *f′* through the 2SAT algorithm and run the algorithm as a black box. Finally, we analyze the truth assignments for each clause in *f′*, and if all outputs of the 2SAT algorithm are satisfiable, then we return TRUE, else FALSE.

Correctness:

Since *f* is made up of three literals, we need to modify them to make a valid input for 2SAT. We do this by paring up each literal with its complement, making each clause a size of 2. If the literals in each clause within *f′* have opposite assignments, it is a satisfiable clause in 2SAT. If they have the same assignments, the clause would be unsatisfiable since the literal of *x* and its inversion would be a part of the same SCC. If 2SAT gives us a satisfiable outcome for each clause in *f′*, we know the results of every literal in each clause within *f* evaluate to either all *true* or *false*. If 2SAT gives us an unsatisfiable outcome for even one of the clauses in *f′*, then we know there is at least one literal result within a clause in *f* that causes the clause to not fulfill the all *true* or *false* requirement.

Runtime:

Modifying the CNF boolean formula takes O(nm) time. Running 2SAT as a black box takes O(n + m) time. Making the overall runtime O(nm).

B)

1.) This problem is within an NP.

We can traverse through each clause within the CNF boolean formula input to verify satisfiability in O(m) time, which is also in polynomial time. Thus, the 3-at most-3-SAT problem is within an NP.

2.) Reduction (3SAT - > 3-at most-3-SAT)

We can show a reduction from the 3SAT problem to the 3-at most-3-SAT problem for the solution. Given the CNF boolean input, *f*, we check if any *x* literals appear in more than three clauses. If so, we replace the *k* appearances of *x* with a new variable and add the new clauses to *f*, we'll call the new boolean formula *f'*. This is done in O(nm) time. Next we pass *f'* into the 3-at most-3-SAT algorithm. Once solution, *S*, is returned from the 3-at most-3-SAT algorithm, no further transformation is needed, and we can check *S* to see if the boolean formula is satisfiable or not. Returning the boolean formula or NO takes O(1) time. Finally, this means that *f* is a satisfiable boolean formula IFF *f'* is a satisfiable boolean formula.

Collaborators:

- Lilley, Zachary J: zlilley3@gatech.edu
- Bertrand, James M: jbertrand9@gatech.edu
- Ramasamy, Veerajothi: vramasamy9@gatech.edu
- Acker, Joshua R: jacker7@gatech.edu
- Shah, Jeet Hemant: jshah328@gatech.edu