Kushagr,Gupta & Rayan,Dasoriya
kgupta7,rdasori@ncsu.edu

1. The *min-ones satisfiability problem (M1-Sat)* is defined as follows.

   Given a formula F in conjunctive normal form and an integer k, does there exist a satisfying assignment A for F such that the number of true variables in A is $\leq k$?

   Prove that M1-Sat is NP-complete. Be sure to include and clearly mark all parts of the proof, even those that may seem trivial.

(b) Let p be the number of variables that have at least one *positive* occurrence – a variable x has a positive occurrence if some clause contains x as a literal (as opposed to $\overline{x} = \neg x$). Give a $O(n)$ algorithm for solving M1-Sat when p is constant, where n is the total number of literals in F.

(a) To prove that M1-Sat is NP complete, we have to prove two things:

i. M1-SAT is NP
Proof:
M1 SAT is in NP because any combination of assignment of Boolean values to the variables which claims to satisfy the validity of the given expression or which acts as a certificate can be checked in a deterministic polynomial time. If we are given a certificate, we can directly plug in the values of those variables in the formula F and can check thee validity of the assignment.

ii. There exists an NP Complete problem which can be reduced to M1-SAT problem in a deterministic polynomial time.
Proof:
We know that SAT (Satisfiability of Boolean formulas) is NP Complete problem. So, we just need to prove that any instance of a SAT problem can be reduced to M1-SAT. The first step is to convert each clause to CNF form. We start by creating a truth table and evaluating all the possible assignments to the variables. Using this, we will build a DNF and negate that formula to get a CNF form. At this point, we have converted each clause in the formula to CNF. Now, since we are given that M1-SAT is in CNF form and there doesn't exist any restriction on converting SAT to M1-SAT(unlike the 3SAT problems where you should have 3 literals in every clause), we can easily reduce a SAT problem to M1-SAT in deterministic polynomial time. The SAT problem will give us a solution or the satisfying assignment to the formula F. We just need a linear time operation to count the number of true variables in the satisfying argument and if the count is less than k, then we will return accept. The whole process completes in polynomial time. Hence, we can state that M1-SAT is NP Complete.

(b) The algorithm for M1 SAT in deterministic mode is described below. It runs in linear time. Since the algorithm is checking the state of each literal in the formula, it will run in $O(n)$ time.

```
DeterministicM1SAT(F,n,k,p):
    // F: CNF expression; n: number of variables;
    Declare an array A of size n and initialize the value as false
    Create a set M with variables having at least 1 +ve occurrence
    Set initial pattern of M to false
    //There can be 2^p possible pattern for M
    //So we will check it for all the possible patterns
    for each possible pattern M:
        Set all values in A to False
        For each variable m in M:
            if m is True in this pattern:
                Set m to True in A
        if count of True in A > k
            return Reject
        if F is satisfied:
            return Accept
    return Reject
```

2. Do Problem 34-4, parts (a) and (b) on page 1104. The problem statement is identical to that of Problem 1 of Homework 3, but there is no restriction on the t_j's. Part (a) asks you to formulate the problem as a decision problem; part (b) asks for an NP-completeness proof. You already did parts (c) and (d) in Homework 3.

   (a) Following is the statement for decision problem:
   Suppose that we have one machine and a set of n tasks $a_1, a_2, ...a_n$, each of which requires time on the machine. Each task a_j requires t_j time units on the machine (its processing time), yields a profit of p_j , and has a deadline d_j. The machine can process only one task at a time, and task a_j must run without interruption for t_j consecutive time units.If we complete task a_j by its deadline d_j,we receive a profit p_j , but if we complete it after its deadline, we receive no profit. As an decision problem, we are given the processing times, profits, and deadlines for a set of n tasks, and we find if for a constant P, if there exists an order of jobs which when executed in that order, yields a total profit of at least P.

   (b) To show that the above decision problem is in NP Complete, we need to prove two parts:

      i. The decision problem is in NP
      Proof:
      The problem is in NP because if we find a valid order of jobs or a certificate which can be done in the deadline, then we can check if the profit yielded by the job is at least P or not to check the validity of the claim in polynomial time. This validator will run in O(n^2) time.

      ii. There exists an NP Complete problem which can be reduced to this decision problem in deterministic polynomial time
      Proof:
      In Subset sum, we are given a set of integers and a target sum. We are supposed to find if their exists a subset of integers from that set whose sum is equal to the target sum or not.
      We can reduce subset sum to prove that the job scheduling algorithm is in NP Complete.
      Let M = (m[1..n],t) be an instance. We will create an instance of N = (m[1..n], d[1..n], m[1..n], t) in which for all i, d[i]=t . This construction will take O(n) time. Let us assume that there exists a subset of M which sums up to t, then there exists an order in N as well. We can also say that, if there is an order in N, then there

exists a subset N' of tasks in N where the last task in N' will meet the deadline, i.e., $\sum_{i \epsilon N'} m[i] \leq t$ (Since d[i] = t)and the restriction on profit as well, i.e, $\sum_{i \epsilon N'} m[i] \geq t$. In this way, we have satisfied the two constraints of deadline and profit. This gives us $\sum_{i \epsilon N'} m[i] = t$. Hence, the solution of M will be True.
Since we know that subset sum is a NP complete problem and we have reduced that into job scheduling problem, we can conclude that the above-mentioned problem is NP Complete.