(1) Monotone SAT (10 points).

A monotone SAT formula is a SAT formula with no negated variables. So, for example,

$$\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_4)$$

is a monotone formula. A monotone formula is easy to satisfy, we can simply set all variables to be *true*. In the MONOTONE SAT problem we are given a monotone formula and an integer k, and ask if there is a way to satisfy formula Φ with setting at most k variables true (and the rest of the variables false).

Show that the MONOTONE SAT problem is NP-complete.

Solution 1. MONOTONE SAT is clearly in NP: all we need to do is to show the k variables that satisfy all the clauses, and check that those variables are satisfied. This certificate is $O(k)$ and checking the clauses is linear in the number of terms, and we will only output a "yes" answer for a correct

To prove that the problem is NP-complete, we show that VERTEX-COVER $\leq$ MONOTONE SAT. Vertex-Cover is given by an undirected graph G and integer k. To do so, we need to show how to solve the vertex cover problem using a MONOTONE SAT subroutine. Consider an input to VERTEX-COVER: a graph $G = (V, E)$ with n nodes and an integer $k \leq n$. Assume G has ℓ isolated vertices, that is vertices not adjacent to edges. Clearly the answer to VERTEX-COVER is "yes" if $k \geq n - \ell$ as we can select all vertices adjacent to edges. What remains to solve is the case when $k \leq n - \ell$. We will transform such a VERTEX-COVER problem to an equivalent input to monotone SAT. To do this, we use a variable x_v for all nodes v, and add a clause $x_v \vee x_w$ for all edges (v, w), so let $\Phi = \wedge_{(v,w)\in E} x_v \vee x_w$.

- Given a set of k nodes S in V that form a vertex cover, setting the corresponding variables x_v to true makes the formula satisfiable.
- If the formula Φ is satisfiable by setting k variables true, then the nodes corresponding to these true variables form a vertex cover.

Solution 2. An alternate way to prove that the problem is NP-complete, we show that SAT $\leq$ MONOTONE SAT. Consider a SAT formula Φ with n variables $x_1, \ldots, x_n$. This formula has negated variables. Add n new variables $y_1, \ldots, y_n$, and replace all occurrences of $\bar{x}_i$ in the formula with y_i for each i. This gives us a monotone formula Φ'. Now set $k = n$.

If we can ensure that exactly one of x_i or y_i is selected for each i in this monotone formula, we will have a solution to out SAT problem. To do this, add n new clause to Φ' with $(x_i \vee y_i)$ for all i. Let the resulting formula Φ''.

- Given a truth assignment for Φ, we can make Φ'' true by setting $y_i = \bar{x}_i$.
- If the formula Φ'' is satisfiable by setting $k = n$ variables true, then due to the n clauses of the form $(x_i \vee y_i)$ we must have exactly one of x_i or y_i to be true, that is, we must have that $y_i = \bar{x}_i$. And so x makes the formula Φ true.

(2) Weighted Multiple Interval Scheduling (10 points). Recall the weighted interval scheduling problem, where we had n possible jobs, each job i had value v_i and required a resource for a fixed interval of time $[s_i, f_i]$. The problem was to find a set of jobs I that request disjoint intervals and have maximum total value $\sum_{i \in I} v_i$. A decision version of the problem would have an additional input value v and ask if there is a set I requesting disjoint intervals with total value at least v, that is $\sum_{i \in I} v_i \geq v$.

In this problem, jobs may require a few disjoint intervals. For example, a single job i may require the resource for [9am,9:10am] and then again [2pm, 2:22pm] and you can only accept the job i if it is possible to assign the resource to job i for all the requested intervals of time.

The input to the problem consist of a target value v, and a set of n jobs, each with a value v_i and a set of intervals of time called R_i, where R_i is the set of times job i needs the resource (may consist of multiple intervals).[1] The problem is to decide if there is a subset of jobs I with $\sum_{i \in I} v_i \geq v$, such that all requests of all the jobs included in I are disjoint, i.e. $\cup_{i \in I} R_i$ consists of disjoint intervals. We call this the Weighted Multiple Interval Scheduling problem.

For example if the minimum total value is $v = 2$ and job 1 has $R_1 = \{[1, 3], [4, 6]\}, v_1 = 1$, job 2 has $R_2\{[7, 8]\}$, $v_2 = 1$, and job 3 has $R_3 = \{[2, 4], [7, 8]\}$, $v_3 = 1$, then choosing $I = \{1, 2\}$ would satisfy the requirements. However $I = \{1, 3\}$ would not since $[2, 4]$ overlaps with $[1, 3]$.

Show that the Weighted Multiple Interval Scheduling problem is NP-complete.

Solution. The problem is clearly in NP: given a set of k requests, one can check that each pair are indeed requesting disjoint intervals. This can be done in time at most quadratic in the input size (due to checking all pairs).

To prove that its NP-complete, we reduce from Independent Set. We need to show that we can solve an Independent Set using a subroutine to Weighted Multiple Interval Scheduling.

Input to the independent set problem is a graph $G = (V, E)$ and an integer k. Have each edge $e \in E$ correspond to a disjoint interval of time (an hour). Nodes correspond to jobs, with node $i \in V$ requesting the intervals that correspond to the edges adjacent to node i, and each job has value 1 with target value $v = k$.

- This construction is clearly done in polynomial time
- If G has an independent set I of size k, and the requests corresponding to the k nodes in I form k requests with disjoint intervals. So the answer to the Weighted Multiple Interval Scheduling we constructed is also "yes".
- if the constructed Weighted Multiple Interval Scheduling problem has k requests with disjoint intervals, then the nodes corresponding to these requests must form an independent set.

[1] You may assume that all numbers involved (values and beginning and end of intervals) are integers, but can be arbitrary large numbers.