CS41 Lab 10: more verifiers and reductions

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

1. Give polynomial-time verifiers for the following problems, none of which are known to have polynomial-time algorithms.

   (a) Independent-Set.

   (b) Vertex-Cover.

   (c) Sat.

   (d) Factoring. Given numbers n, k written in binary, output YES iff n is divisible by d for some $1 < d \leqslant k$.

   (e) Not-Factoring. Given numbers n, k written in binary, output YES iff n is **NOT** divisible by d for any $1 < d \leqslant k$.

   Hint: The following problem **is solvable**[1] in polynomial time:

   Primes: Given a number n written in binary, output YES iff n is a prime number.

2. Multiple-Interval-Scheduling (K&T 8.14) In this problem, there is a machine that is available to run jobs over some period of time, say 9AM to 5PM.

   People submit jobs to run on the processor; the processor can only wok on one job at any simgle point in time. However, in this problem, each job requires a **set of intervals** of time during which it needs to use the machine. Thus, for example, one job could require the processor from 10AM to 11AM and again from 2PM to 3PM. If you accept this job, it ties up your machine during these two hours, but you could still accept jobs that need any other time periods (including the hours from 11AM to 2PM).

   Now, you're given an integer k and a set of n jobs, each specified by a tset of time intervals, and you want to answer the following question: is it possible to accept at least k of the jobs so that no two of the accepted jobs have any overlap in time?

   In this problem, you are to show that Multiple-Interval-Scheduling $\in$ NP-complete. To assist you, we've broken down this problem into smaller parts:

   (a) First, show that Multiple-Interval-Scheduling $\in$ NP.

   (b) In the remaining two parts, you will reduce

   $$\text{Independent-Set} \leqslant_P \text{Multiple-Interval-Scheduling} .$$

[1] This actually wasn't known until 2002, when Agrawal, Kayal, and Saxena created the AKS primality test. Kayal and Saxena were undergraduates at IIT Kanpur at the time; Agrawal was their advisor.

Given input $(G = (V, E), k)$ for Independent-Set, create a valid input for Multiple-Interval-Scheduling. First, divide the processor time window into m distinct and disjoint *intervals* $i_1, \ldots, i_m$. Associate each interval i_j with an edge e_j. Next, create a different job J_v for each vertex $v \in V$. What set of time intervals should you pick for job J_v?

(c) Finally, run the Multiple-Interval-Scheduling algorithm on the input you create, and output yes iff the Multiple-Interval-Scheduling algorithm outputs yes. Argue that the answer to Multiple-Interval-Scheduling gives you a correct answer to Independent-Set.