CSC 373 — Tutorial Solutions for Week 8 — Winter 2019

1. (a) variables: amounts to ship from each producer to each consumer:
   - $x1$ = number of shnupells to ship from Mexico to New York
   - $x2$ = number of shnupells to ship from Mexico to California
   - $x3$ = number of shnupells to ship from Kansas to New York
   - $x4$ = number of shnupells to ship from Kansas to California

   (b) objective function: minimize total shipping cost:

   minimize $4\, x1 + x2 + 2\, x3 + 3\, x4$

   (c) constraints:

   subject to
   - $x1 + x2 \le 8$ (production from Mexico)
   - $x3 + x4 \le 15$ (production from Kansas)
   - $x1 + x3 \ge 10$ (consumption in New York)
   - $x2 + x4 \ge 13$ (consumption in California)
   - $x1, x2, x3, x4 \ge 0$ (can't ship negative amounts)

2. - Use variable $x_{i,j}$ for each pair c_i, s_j close enough.

   - Maximize $\sum_{i,j} x_{i,j}$ (total number of clients connected) subject to:
     - A. $\sum_j x_{i,j} \le 1$ for all i
       (c_i connected to at most one base station)
     - B. $\sum_i x_{i,j} \le L$ for all j
       (s_j has at most L clients connected)
     - C. $x_{i,j}$ in $\{0,1\}$ for all $x_{i,j}$
       (either c_i connected to s_j or not)

   - Any set of connections yields a feasible solution by setting $x_{i,j} = 1$ if c_i is connected to s_j ($x_{i,j} = 0$ otherwise): constraints C automatically satisfied, constraints A satisfied since each client is connected to at most one station, and constraints B satisfied since each station connected to at most L clients.
     This implies the maximum objective value is at least as large as the maximum number of clients that can be connected.

   - Any feasible solution yields a set of connections by connecting c_i to s_j iff $x_{i,j} = 1$. By constraints A, each client is connected to at most one station and by constraints B, each station is connected to at most L clients. And since variables $x_{i,j}$ are either 0 or 1 (by constraints C) and are defined only for pairs c_i, s_j close enough, the range constraint is satisfied.
     This implies the maximum number of clients that can be connected is at least as large as the maximum objective value.

   - Hence, maximum objective value = maximum number of clients that can be connected, and any optimal solution to the linear program yields an optimal set of connections.

   - BUT, constraints C make this an instance of *Integer Programming* (IP), not general Linear Programming (LP). In general, IP problems cannot be solved in polytime!

   - If we relax constraints C to turn this back into LP,

C'. 0 <= x_{i,j} <= 1 for all x_{i,j}
then we can find an optimal solution in polytime. However, the solution is not guaranteed to be integer-valued! And if values x_{i,j} take on fractional values in [0,1], it's not clear what that means for assigning clients to base stations anymore... In this case, the problem *does* have an integer solution (from network flows, for example), but in general this does not have to be the case -- you will see examples later.

3. Show that the following UNARY-PRIMES decision problem belongs to P.
    Input: 1^n (i.e., a string of '1's of length n).
    Question: Is n prime?

The following algorithm decides UNARY-PRIMES.

```
On input 1^n:
    For k = 2,3,...,n-1:
        If k divides n, return False
    Return True if no value of k worked.
```

The algorithm returns True iff k is prime, by definition.
The division can be carried out by repeated subtraction, which takes time O(n^2) for each value of k, so the entire algorithm runs in time O(n^3).

NOTE: This works because n is the *size* of the input at the same time as the *value* of the input: for any other base, this would not work because the value m would be represented using n = log m many digits so the size would be proportional to n = log m and the running time would become exponential (as a function of n).

4. Show that the following TRIANGLE decision problem belongs to P.
    Input: An undirected graph G = (V,E).
    Question: Does G contain a "triangle", i.e., a subset of three vertices with all edges between them present in the graph?

The following algorithm decides TRIANGLE.

```
On input G:
    For each triplet of vertices (u,v,w) in G:
        Return True if G contains each edge (u,v), (v,w), (w,u).
    Return False if no triplet checked out.
```

By definition of TRIANGLE, the algorithm will return True iff G contains a triangle.
Let n = |V| (number of vertices) and m = |E| (number of edges) in G.
There are (n choose 3) = \Theta(n^3) many triplets of vertices in G, and it is possible to enumerate them one by one in time O(n^3). For each triplet, it takes time O(m) to verify the presence of the three edges (depending on how G is encoded, this could be reduced). So the algorithm runs in time O(m n^3).

5. Show that the following CLIQUE decision problem belongs to NP.
    Input: An undirected graph G = (V,E) and a positive integer k.
    Question: Does G contain a k-clique, i.e., a subset of k vertices with all edges between them present in the graph?

```
a---b    For example, the graph pictured on the left contains a
|\ /|    3-clique (there are sets of 3 vertices with all edges
| c |    between them, e.g., {a,b,c}), but it does not contain a
|/ \|    4-clique (every set of 4 vertices is missing at least one
d---e    edge, e.g., {a,b,c,d} is missing (b,d)).
```

Verifier for CLIQUE:
    On input <G,k,c>, where c is a subset of vertices:
        Return True if c contains k vertices and G contains edges between all pairs of vertices in c; return False otherwise.

Verifier runs in polytime (where n = |V|, m = |E|): checking all pairs of vertices in c takes time O(k^2 m) (O(k^2) pairs in c, time O(m) for each one).
If <G,k> (- CLIQUE, then verifier returns True when c = a k-clique of G; if verifier returns True for some c, then <G,k> (- CLIQUE (c is a k-clique).

CLIQUE (- P? Unknown (checking all possible subsets not polytime because k not fixed, part of input).

* Contrast CLIQUE with TRIANGLE: TRIANGLE (- NP (on input <G,c>, check c encodes a triangle in G), but TRIANGLE (- P as well.

  What’s the difference? Same algorithm to decide CLIQUE takes time O(n^{k+1}), except that k is part of the input (instead of being fixed) so this could be as bad as, e.g., O(n^{n/2}) -- not polytime.

6. Show that the following SUBSET-SUM decision problem belongs to NP.
    Input: A set of positive integers S and a positive integer t.
    Question: Is there some subset of S whose sum is exactly t?

For example, S = {4,22,10,8}, t = 14 belongs to SUBSET-SUM, but S = {4,22,10,8}, t = 13 does not belong to SUBSET-SUM.

Verivier for SUBSET-SUM:
    On input <S,t,c>, where c is a subset of S:
        Return True if \sum_{y (- c} y = t; return False otherwise.

Clearly runs in polytime (addition of numbers is polytime), and if <S,t> (- SUBSET-SUM, then there is some value of c such that verifier returns True (c = subset whose sum equals t); if verifier returns True for some c, then <S,t> (- SUBSET-SUM (c is subset with sum t).