# CSci 423 Homework 11

Due: 1:00 pm, Wednesday, 11/28

Eric Shih

1. (10 points) Problem 5.17 on page 212. (Hint: Give a simple algorithm) If the alphabet is unary, the dominoes only differ in the number of 1$s$ that each has on the top and bottom. This case is solved by:

   M=Given a collection of dominoes

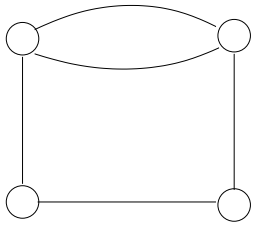
   (a) If some domino has the same number of 1$s$ on top and bottom, there is a trivial match, so accept.

   (b) If all the dominoes have more 1$s$ on top than on bottom, there is no possibility of a match, so reject. Likewise, if all the dominoes have less 1s on top than on bottom, reject.

   (c) Find one domino with more 1$s$ on top than on bottom (say a difference of a 1$s$), and one domino with more 1$s$ on bottom than on top (say a difference of b 1$s$). Choosing b of the first domino and a of the second should make an an equal number of 1$s$ on both top and bottom, and hence a match, thus accept.

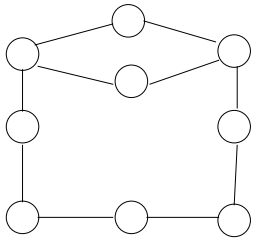
2. (10 points) This problem considers an attempt at a polynomial reduction from one problem to another that does not work. Your task is to find the flaw. A bipartite graph is an undirected graph in which every cycle has even length. We attempt to show that the Hamiltonian cycle (a cycle that passes through each node exactly once) problem polynomially reduces to the Hamiltonian cycle problem in bipartite graphs. We need a function $T$: $\{\text{graphs}\}\rightarrow\{\text{bipartite graphs}\}$ such that T can be computed in polynomial time and for any graph $G$, $G$ has Hamiltonian cycle iff $T(G)$ has a Hamiltonian cycle. Let $T(G)$ be the bipartite graph obtained by inserting a new vertex on every edge. What is wrong with this transformation?

   This transformation has a condition that for any graph G, G has a Hamiltonian cycle iff T(T) has a Hamiltonian cycle. A counter-example will be used to show the error. If G is the graph shown here:



   Graph G

   G does have a Hamiltonian cycle, but there is a T(G) that does not have a Hamiltonian path.



Graph T(G)

From here, it is obvious that there is no path that visits each vertex once while also finishing at the starting vertex. Thus it does not have a Hamiltonian cycle showing what is wrong with the transformation.

3. (10 points) The SET INTERSECTION (SI) problem is defined as follows:
INSTANCE: Finite sets $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$.
QUESTION: Is there a set $T$ such that $|T \cap A_i| \geq 1$ for $i = 1, \ldots, m$ and $|T \cap B_j| \leq 1$ for $j = 1, \ldots, n$?
Show that SAT polynomially reduces to SI.
*reference research.cs.queensu.ca/ cisc365/2010F/365%20SetIntersectionNPC.pdf*

For a CNF expression, $T$ can be transformed as follows:

(a) For each clause in the CNF expression, it is turned to set $A_i$. Each literal in $A_i$ becomes an element of the set. The unnegated literal $x_i$ will become element $T_i$. Negated literal $\bar{x_i}$ will become element $F_i$. $m$ in the set intersection will equal the number of clauses in the CNF expression

(b) For each boolean variable, set $B_j$ can be created with elements $T_j$ and $F_j$. $n$ from the set intersection will be the number of boolean variables.

**Reading Summary 5**

The $P$ versus $NP$ discussion is a problem that is unlikely to be solved in the near future, but its research has driven numerous topics in varying computational problems. This article looks at how people have tried to solve the $P$ versus $NP$ problem and how it has shaped the research of computer science. It also shows how to handle $NP$-complete problems, a new proof strategy, the possibility that quantum computing can help to solve $NP$-complete problems, and finally an idea to "separate $P$ from $NP$ using using algebraic-geometric techniques."

The $P$ versus $NP$ problem is one that compares whether or not $P$, algorithms that run in polynomial time, is equal to $NP$, nondeterministic polynomial time. If $P = NP$, then "for every problem that has an efficiently verifiable solution, we can find that solution efficiently as well." This would mean that there would be a simple efficient algorithm to solve any $NP$-complete problem, which would in turn allow us to apply these solutions to improve real world issues. Many researchers believe that it is impossible for this to happen, so many strategies such as diagonalization, circuit complexity, and proof complexity have been used in an attempt to prove that $P \neq NP$.

Dealing with $NP$-complete problems will undoubtedly be very difficult because of the $P$ versus $NP$ issue. To deal with any real world $NP$-complete problems one can utilize a host of strategies. Brute force methods, parameterized complexity, approximation, and Heuristics and Average-Case Complexity are some available strategies to solve these problems. Usually one will have to use a combination of these techniques in order to fully solve an $NP$-complete problem.

Interactive proof models is one strategy being used to find a solution. PCPs fall under this category and can be used to "to show the limitations of approximation for a large number of optimization questions." Quantum computing is also an option that is being explored. The author is highly doubtful that these machines, if even capable of being built, will be able to solve the problem. It is cited that Shor's quantum computing algorithm "cannot be applied to generic "black-box" search problems so any algorithm would have to use some special structure of NP-complete problems that we don't know about." The most hopeful method being researched is through the use of algebro-geometric polygons to define group representations. And even if this method does work, they predict that it will take at least 100 years to complete.

The $P$ versus $NP$ problem is one that we have only begun to truly uncover. It has promoted many different research areas and will remain one of the top problems to solve in the field of Computer Science.