Exam Name________________________________

1. ***a*** A *Hamilton path* of a graph is a path that goes through every vertex of the graph. A *Hamiltonian cycle* of a graph is a cycle that goes through every vertex of the graph. Show that the problem of deciding if a graph has a Hamilton path is polynomially reducible to the problem of deciding if a graph has a Hamiltonian cycle.

1. ***b*** Is the problem of finding a longest path in a graph polynomially reducible to the problem of finding a shortest path? Justify your answer.

2. Consider the problem which accepts as input a binary array (all values of *A* are 0 or 1) and integer *k* and decides whether the array contains at least *k* entries of 1 such that no pair of these *k* 1's appear on the same row or on the same column. For example, if the array is

$$\begin{matrix} 0 & \mathbf{1} & 1 & 0 \\ \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 1 & \mathbf{1} \end{matrix}$$

and *k*=3, the answer is true because of the bold entries (1,2), (2,1) and (3,4). Is this problem NP-complete? Justify your answer.

SOLUTIONS

1. ***a*** Take any instance $G=(V,E)$ of the spanning path problem. Construct a new graph $G^*=(V\cup\{w\},E^*)$ where E^* is *E* plus an edge between *w* and every $v\in V$. *G* has a spanning path if and only if G^* has a Hamilton cycle, and the spanning path of *G* is obtained from the Hamilton cycle of G^* by removing *w* and both edges incident with it.

b The problem of finding a shortest path belongs to P - it admits a polynomial time algorithm to solve it. A graph $G=(V,E)$ has a path of length $|V|$ if and only if it has a Hamilton Path, and the problem of testing if a graph has a Hamilton Path problem is NP-complete. If the problem of finding a longest path in a graph were polynomially reducible to the problem of finding a shortest path, this would imply that the problem of finding a shortest path is NP-complete, which it almost certainly is not.

2. The problem is almost certainly not NP-complete because it is equivalent to finding a maximum matching in a bipartite graph, which can be solved using a polynomial time algorithm to find a maximal flow in a network. To show this reduction, we label the rows of array *A* to be *X* and the columns *Y*. We add a source *s* and a terminus *t*. The edges of our network are $\{(x,y)\mid A[x,y]=1\}\cup\{(s,x)\mid x\in X\}\cup\{(y,t)\mid y\in Y\}$. All edges have capacity 1. *A* admits *k* entries of 1 if and only if the network admits a flow of value at least *k*, which can be determined in polynomial time.