Homework 22

Brian Knotten, Brett Schreiber, Brian Falkenstein

October 18, 2018

10

Define $HCD(G)$ to be the algorithm for the decision problem for if a Hamiltonian Cycle exists for a graph G, and $HCO(G)$ to be optimization problem for actually finding the Hamiltonian Cycle in graph G. That is $HCD(G)$ will output 1 if an HC exists in G, and a 0 if not, and $HCO(G)$ will actually output the edges that constitute a HC in G, or 0 if one doesn't exist. The claim is that $HCO(G) \leq_{poly} HCD(G)$, IE Hamiltonian Cycle is self reducible.
In order to prove this, we must show that we can use $HCD(G)$ to output a list of edges constituting a HC in G, in polynomial time.
First, assume graph G is defined as a list of vertices and a list of edges. Consider the following pseudo-code:

```
HCO(V, E):
      if HCD(V, E):                                  #initial check, make sure G has an HC
           HC = []                                   #initialize solution
           while E.hasNext:                         #continue until no edges left
                testEdge = E.pop                    #remove an edge from the graph
                if HCD(V, E):                       #check if HC exists in G minus one edge
                      HC.append(testEdge)          #if so, add the edge we removed to solution
           return isHC(HC, V, E)                     #function to determine if a path is a HC for a graph
     return 0
```

The general strategy of this algorithm is to look at an edge e in G, determine if we can still form an HC in G when we remove e. If so, we can safely add e to our solution. If not, we can exclude e. We repeat this until there are no edges left in G, and then we test if the cycle we've found is actually a HC. Note that *isHC* could be defined very simply by checking that:

- All edges in HC exist in E
- No vertex in V is visited more than once in HC
- HC spans all vertices in V

The number of times HCD will be called inside of HCO is at most n, where n is the number of edges in G, as each time it is called at least 1 edge is removed. Similarly, *isHC* will take at most n time, as if HC is a hamiltonian cycle, the max edges it could contain will be n. This results in a total run time of $n + n = O(n)$, a polynomial. Thus, we have proven that Hamiltonian Cycle is self reducible, and if we can determine whether a graph has an HC in polynomial time, we can find the HC in polynomial time.

12

Same idea as 10. To prove VC is self reducible, we must show that $VCO \leq_{poly} VCD$. Given the graph G and k:

```
VCO(V, E, k):
      if VCD(V, E, k):                        #initial check, make sure G has a VC of size k
            VC = []
            while V.hasNext:                   #continue until no V's left
                 testVertex = V.pop             #remove a vertex to check
                 if VCD(V, E, k -1):          #check if we still have a VC after removing the vertex
```

```
                VC.append(testVertex)   #if so, add it to the solution
                k -= 1
        return isVC(VC, V, E, k)
    return 0
```

The number of times VCD will be called inside VCO will be at most n, where n is the number of vertices in G. Further, $isVC$ will take at most nk time, as it will need to go through the k vertices in VC, and determine if all of the n edges are incident to the k vertices. This results in a runtime of $O(n + nk) = O(n)$. Thus, if VCD has a polynomial time algorithm, so does VCO, and Vertex Cover is self reducible.