CSCE 629: Analysis of Algorithms
Home Work - 6

Shabarish Kumar Rajendra Prasad
UIN - 228000166

December 2019

1 Question 1

Case (i): If VC-O is solvable in polynomial time, VC-D is also solvable in polynomial time

Consider that We have an algorithm, that solves the VC-O problem in polynomial time. Then we can also solve the VC-D problem in polynomial time as follows

---

Algorithm: VCD(G=V, E, k)

```
vertex_covers = VCO(G)

if k >= len(vertex_covers):
    return True
else:
    return False
```

This algorithm runs the solution VC-O algorithm to retrieve the vertex covers, finds the length of the array and makes a decision based on that. Finding the length of the vertex covers is bound by cardinality of the vertices in the graph, thus the time complexity of the entire problem is bounded by the time complexity of VC-O algorithm since it is greater. Thus if the time complexity of the VC-O algorithm is bound by a polynomial, the time complexity of the above algorithm is also bound by a polynomial . This proves that VC-D problem is solvable in polynomial time if VC-O is solvable in polynomial time.

Case (ii): If VC-D is solvable in polynomial time, VC-O is also solvable in polynomial time

Consider that we have an algorithm that solves the VC-D problem in polynomial time. Then we can also solve VC-O problem in polynomial time as follows:

Algorithm: VCO(G=V, E)

```
for i=0 to i=|V|:
    if VCD(G,i):
        k = i
        break
    else:
        continue

vertex_covers = set()
```

```
for v in V:
    G1 = G - v
    if VCD(G1,k-1):
        vertex_covers.add(v)

return vertex_covers
```

The algorithm is simple. We check and select the minimum value of k for which the VC-D algorithm returns $True$. Then for every vertex in the graph, we remove the vertex and form a new graph $G1$ and then we check if $VCD(G1, k-1)$ is true. If it is true, we add that vertex to the set of vertex covers and return after iterating through all the vertices.

Aside from running the VCD routine, the complexity of the program can be described as $O(n^2)$, which is polynomial in itself. If VC-D also runs in polynomial time, the complexity of the program would be polynomial times polynomial which also yields a polynomial. Thus if VC-D can be solved in polynomial time, VC-O can also be solved in polynomial time.

2 Question 2

We consider a problem is in $\mathcal{NP}$, if given a 'certificate'(a solution) we can verify whether it is correct in polynomial time. A 'certificate' for the VC-D problem is y, which is a subset of vertices in the Graph. The algorithm to verify the solution is given below:

Algorithm: CheckVCD(G={V,E},y,k)

```
count = 0
for v in y:
    remove all edges adjacent to v from E
    count += 1
if (count==k) and (E is empty):
    return True
else:
    return False
```

It is easier to see that this is verifiable in polynomial time. Thus VC-D are in $\mathcal{NP}$.

3 Question 3

If we can show that IS $\leq \mathscr{P}_m$ Clique, it would prove that Clique is also an $\mathcal{NP}$-complete problem, since Independent sets problem is $\mathcal{NP}$-complete. Since we know that Clique is just the complimentary graph of the Independent Set graph, we shall write the IS to clique reduction algorithm as follows:

Algorithm: IS_to_Clique(G)

```
allEdges = set of all possible edges in G
G1 = new empty Graph

for (u,v) in allEdges:
    if (u,v) is not an edge in G:
        G1.add(edge(u,v))

return G1
```

Constructing the set of all possible vertices in G is an ${}^nC_2$ time operation, which is polynomial, where $n = |V|$. Thus the time complexity of the algorithm can be described as a polynomial. This proves that IS $\leq \mathscr{P}_m$ Clique. Thus the Clique problem is also $\mathcal{NP}$-complete.

4 Question 4

In Question-2, we proved that VC-D is in $\mathcal{NP}$. And from Question-1, we also showed that VC-O $= \mathscr{P}_m$ VC-D. Thus it follows that, if VC-O is solvable in polynomial time, then VC-D is also solvable in polynomial time. We know that $\mathcal{P} \subseteq \mathcal{NP}$. Thus, to prove $\mathcal{P} = \mathcal{NP}$, we just have to show that $\mathcal{NP} \subseteq \mathcal{P}$ .i.e., every problem in $\mathcal{NP}$ is also in $\mathcal{P}$.

Now consider any problem $\mathcal{F}$ in $\mathcal{NP}$. Since we know VC-D is also in $\mathcal{NP}$, $\mathcal{F} \leq \mathscr{P}_m$ VC-D. Consider an instance X of VC-D and an instance Y of $\mathcal{F}$. Because of the above mentioned relationship between $\mathcal{F}$ and VC-D, Y can be obtained from X using a polynomial time algorithm and VC-D can also be solved in a polynomial time. Since both these operations are solvable in polynomial time, it can also be said that $\mathcal{F}$ is solvable in polynomial time. Thus every problem in $\mathcal{NP}$ is solvable in polynomial time.

Thus $\mathcal{P} = \mathcal{NP}$.