1 Problem 2: Search reduces to decision

Let

\[ VC = \{(G, k) \mid G \text{ has a vertex cover of size } \leq k\} . \]

If \( G \) is a graph, let \( V(G), E(G) \) represent its vertices, edges. Also, if \( S \subseteq V(G) \), let \( G - S = (V(G) - S, E(G) - \{ e \in E(G) \mid e \cap S \neq \emptyset \}) \). Also, if \( v \in V(G) \), let \( N_G(v) = \{ u \in V(G) \mid \{ u, v \} \in E(G) \} \) be the neighbors of \( v \).

Claim 1. The smallest vertex cover search problem poly-time Turing reduces to VC.

Proof. The following algorithm suffices.

1. \( A^{VC}(G_0) \)
2. for \( k \leftarrow 0, \ldots, |V(G_0)| \)
3. if \( (G_0, k) \in VC \), break
4. \( C \leftarrow \emptyset, G \leftarrow G_0 \)
5. while \( k > 0 \)
6. let \( v \in V(G) \)
7. if \( (G - \{ v \}, k - 1) \in VC \)
8. \( (C, k, G) \leftarrow (C \cup \{ v \}, k - 1, G - \{ v \}) \)
9. else
10. \( (C, k, G) \leftarrow (C \cup N_G(v), k - |N_G(v)|, G - N_G(v) - \{ v \}) \)
11. return \( C \)

Note that we could use binary search in place of lines 2-3 to find the size \( k \) of a smallest vertex cover, but for simplicity we don’t bother. To see the correctness of the algorithm, we claim that the while loop maintains the following invariant: \( k \) is the size of a smallest vertex cover of \( G \) and every smallest vertex cover of \( G \) when unioned with \( C \) is a smallest vertex cover of \( G_0 \).

Clearly the invariant holds initially. If \( (G - \{ v \}, k - 1) \in VC \), then \( v \) is an element of some smallest vertex cover of \( G \), and so the invariant holds. Otherwise, \( v \) is not an element of any smallest vertex cover of \( G \), and so \( N_G(v) \) is in every smallest vertex cover of \( G \), and again the invariant holds.

At each step, \( |V(G)| \) deceases by at least 1, and so the while loop iterates at most \( |V(G_0)| \) times, so the algorithm takes a polynomial amount of time. The loop invariant implies that when the while loop terminates the output is correct. \( \square \)