We consider here the problem of computing the girth, \( g(G) \), of connected undirected graph \( G = (V, E) \), i.e., the minimum length of a cycle (contained) in a graph \( G \), or infinity if \( G \) has no cycle.

Throughout this notes, the term cycle refers to a simple closed walk and the term path refers to a simple non-closed walk. The usual BFS algorithm ignores vertices that have already been explored. On the other hand, if we reach an already-explored vertex at depth \( r \), then \( G \) must contain a cycle of size \( \leq r \); that is, the girth of \( G \) is at most \( r \). Conversely, if \( G \) contains an \( r \)-cycle \( C \) and we start the BFS algorithm at some vertex \( v \in V(C) \), then we are guaranteed to reach an already-explored vertex by the \( r \)th stage. Therefore, we can compute the girth \( g(G) \) of \( G \) as

\[
\min_{v \in V(G)} \left( \text{minimum depth at which a vertex appears for the second time, when we run a BFS starting at } v \right).
\]

We now present an algorithm to compute the girth of a graph. Our algorithm will use a BFS approach from each vertex of the graph.

When searching from vertex \( v \) - that is, constructing a rooted tree with root \( v \) - we need to keep track of the parent of each other vertex; otherwise we might mistakenly identify a closed walk as a cycle.

**Algorithm** \( \text{Girth}(G) \)

1. \( g(G) \leftarrow \infty \) \hfill \text{\scriptsize \( g(G) \) \rightarrow the size of the smallest cycle already found.}
2. for every \( v \in V(G) \) do
3. \( S \leftarrow \emptyset; R \leftarrow \{v\}; \text{Parent}(v) \leftarrow \text{NULL}; D(v) \leftarrow 0 \) \hfill \text{\scriptsize \( D(w) = d(v, w). \)}
4. while \( R \neq \emptyset \) do
5. \( \text{choose } x \in R \)
6. \( S \leftarrow S \cup \{x\}; R \leftarrow R \setminus \{x\} \)
7. for every \( y \in N(x) \setminus \{\text{Parent}(x)\} \) do \hfill \text{\scriptsize \( N(u) = \{w \in V \mid (u, w) \in E\}. \)}
8. if \( y \notin S \) then
9. \( \text{Parent}(y) \leftarrow x \)
10. \( D(y) \leftarrow D(x) + 1 \)
11. \( R \leftarrow R \cup \{y\} \)
12. else
13. \( g(G) \leftarrow \min \{g(G), \ D(x) + D(y) + 1\} \)
14. return \( g(G) \)

Clearly, the running time of this algorithm is \( O(V(V + E)) = O(VE) \).