The key here is that, while this kind of connectivity includes a notion of time, it can be converted into a graph connectivity problem of a more standard sort.

We construct the following directed graph \( G \). We scan through the ordered triples in the trace data, maintaining an array pointing to linked lists associated with each computer \( C_a \). (Each list is initialized to null.)

For each triple \((C_i, C_j, t_k)\) we see in our scan, we create nodes \((C_i, t_k)\) and \((C_j, t_k)\), and we create directed edges joining these two nodes in both directions. Also, we append these nodes to the lists for \( C_i \) and \( C_j \) respectively. If this is not the first triple involving \( C_i \), then we include a directed edge from \((C_i, t)\) to \((C_i, t_k)\), where \( t \) is the timestamp in the preceding element (the previously last one) in the list for \( C_i \). We do the analogous thing for \( C_j \). By explicitly maintaining these lists for each node, we are thus able to construct all these new nodes and edges in constant time per triple.

Now, given a collection of triples, we want to decide whether a virus introduced at computer \( C_a \) at time \( x \) could have infected computer \( C_b \) by time \( y \). We walk through the list for \( C_a \) until we get to the last node \((C_a, x')\) for which \( x' \leq x \). We now run directed BFS from \((C_a, x')\) to determine all nodes that are reachable from it. If a node of the form \((C_b, y')\) with \( y' \leq y \) is reachable, then we declare that \( C_b \) could have been infected by time \( y \); otherwise we declare it could not have.

Let’s argue first about the correctness of the algorithm, then its running time. First, we claim that if there is a path from \((C_a, x')\) to \((C_b, y')\) as in the previous paragraph, then \( C_b \) could have been infected by time \( y \). To see this, we simply have the virus move between computers \( C_i \) and \( C_j \) at time \( t_k \), whenever an edge from \((C_i, t_k)\) to \((C_j, t_k)\) is traversed by the BFS. This is a feasible sequence of virus transmissions that results in the virus first leaving \( C_a \) at time \( x \) or later (by the definition of \( x' \)) and arriving at \( C_b \) by time \( y \).

Conversely, suppose there were a sequence of virus transmissions that results in the virus first leaving \( C_a \) at time \( x \) or later and arriving at \( C_b \) by time \( y \). Then we can build a path in our graph as follows. We start at node \((C_a, x')\) and follow edges to \((C_a, x'')\), for the \( x'' \) when the virus first leaves \( C_a \). (Note that there are such edges since \( x' \leq x \leq x'' \); or else \( x' = x'' \).) In general, for each time that the virus moves from \( C_i \) to \( C_j \) at time \( t_k \), we add the edge from \((C_i, t_k)\) to \((C_j, t_k)\) to the path; if it next moves out of \( C_j \) at time \( t \geq t_k \), we add the the sequence of edges from \((C_j, t_k)\) to \((C_j, t)\). When the virus first arrives at node \( C_b \), we will have just added a node \((C_b, y'')\) to the path; since \( y'' \leq y \) and \( y' \) is the largest \( y \) involving \( C_b \) in the trace data with this property, there is a sequence of edges from \((C_b, y'')\) to \((C_b, y')\), completing the path.

Finally, we consider the running time. Each triple in the trace data causes us to add a constant number of nodes and edges to the graph, so the graph has \( O(m) \) nodes and edges, and since we build it in constant time per node and edge, this takes time \( O(m) \). Running BFS takes time linear in the size of the graph, so this too takes time \( O(m) \).

\[^1\text{ex207.316.912}\]