1. **Ans.** First perform topological sort on \( G \) and re-label the vertices by their rank in the topological sort order. Consequently all edges must be from small label to large label, as \( G \) is acyclic.

Then, construct an array \( A[1..|V|] \) so that \( A[i] \) will eventually store the number of different paths from \( s \) to \( t \).

Suppose vertex \( s \) is labeled by the rank \( i' \). It follows that for all \( j < i' \), \( A[j] \) should be 0. Also, \( A[i'] \), which is the number of paths from \( s \) to \( s \), is equal to 1.

Next, compute \( A[k] \) for all \( k = i' + 1 \), \( i' + 2 \), \ldots sequentially by dynamic program. It is easy to check that the number of paths from \( s \) to \( k \) is equal to the sum of the number of paths from \( s \) to \( j \), for all \( j \) such that \( (j, k) \) is an edge in \( G \). That is,

\[
A[k] = \sum_{j | (j,k) \in E} A[j].
\]

In total, topological sort runs in \( O(|V| + |E|) \) time. The subsequent dynamic program computes \( O(|V|) \) values, each takes time proportional to the number of incident edges to the corresponding vertex, thus taking a total of \( O(|V| + |E|) \) time.

2. **Ans.** Our idea is to use binary search strategy to find \( m \), but at every binary search step, we try to reduce the problem size so that the time becomes linear \( O(|E|) \) instead of \( O(|E| \log |E|) \).

First we check if \( m \leq |E|/2 \) by considering only edges in \( G \) whose label is at most \( |E|/2 \). It is easy to check that \( G \) can be connected by these edges only if and only if \( m \leq |E|/2 \).

To perform the above checking, it suffices to perform a DFS or BFS on \( G \) with only edges from 1 to \( |E|/2 \). This takes \( O(|V| + |E|) \) time, which is \( O(|E|) \) time since \( G \) is connected.

If \( m \leq |E|/2 \), we can remove all edges with labels greater than \( |E|/2 \) and repeat the same procedure to find the desired \( m \). (That is, we check if \( m \leq |E|/4 \) or not.)

Otherwise, if \( m > |E|/2 \), we contract all connected components formed by edges from 1 to \( |E|/2 \), which takes \( O(|E|) \) time (how?). Then knowing that \( G \) is still connected after the contraction (why?), with edges from \( |E|/2 + 1 \) to \( |E| \), we can repeat the same procedure to find the desired \( m \). (That is, we check if \( m \leq 3|E|/4 \) or not.)

In both cases, after \( O(|E|) \) time, we are left with a connected graph whose number of edges is halved, and the target remains to find the desired edges to be included in order to make \( G \) connected. In total, the running time is thus \( O(|E| + |E|/2 + |E|/4 + \ldots) = O(|E|) \).