1. Suppose the rational capacities are $a_i/b_i$, $i = 1, 2, \ldots, n$, we can multiply them all by the least common multiple of all $b_i$ to scale them to integers. The Ford-Fulkerson algorithm augments the capacity by at least one for each augmenting path, therefore it must terminate. To find the original max-flow, we only need to divide the result by the least common multiple.

The rescaling technique above for rational capacities does not apply to irrational capacities. Since the augmenting path can be arbitrarily chosen, it is possible to have an infinite number of loops.

Consider the network in Figure 1 where $c(v_2, v_1) = c(v_2, v_3) = 1$, $c(v_4, v_3) = r = \frac{\sqrt{5} - 1}{2}$ and the capacity of all other edges equals 2. Now we choose the series of augmenting paths as follows: $p_0, p_1, p_2, p_1, p_3, p_1, p_2, p_1, p_3, \cdots$, where

$$
\begin{align*}
p_0 &= < s, v_2, v_3, t > \\
p_1 &= < s, v_4, v_3, v_2, v_1, t > \\
p_2 &= < s, v_2, v_3, v_4, t > \\
p_3 &= < s, v_1, v_2, v_3, t > .
\end{align*}
$$

The flows of this series of augmenting paths are:

$$1, r, r^2, r^2, \cdots ,$$

which never terminates. In other words, the algorithm fails to return the max-flow of this network with this particular choice of augmenting paths.

The network in Figure 1 has max-flow of 5 by choosing the following paths $< s, v_1, t >$, $< s, v_4, t >$ and $< s, v_{2,3}, t >$. However, the algorithm returns $1 + r + r^2 + r^2 + \cdots = 1 + 2 \times \frac{\sqrt{5} - 1}{2} \neq 5$.

2. Given a maximum flow, we first find an edge $(u, v)$, such that $f(u, v)$ (larger than 0) is minimum. We then find a path from $s$ to $t$ containing $(u, v)$, and reduce the flow on that path by $f(u, v)$. The edge $(u, v)$ will not be selected again because its flow is now 0. Repeat the same process until the total flow is reduced to 0. Since one edge is removed each time, we do this at most $|E|$ times. Each path on which we reduced flow could have been an augmenting path, so we could get to our max-flow with at most $|E|$ augmentations.
3. We will prove that (1) any overflowing vertex \( x \) in the initialization step has a simple path to \( s \) and (2) suppose the vertex \( x \) has a simple path to \( s \), and after a RELABEL or PUSH, \( x \) and any new overflowing vertex \( u \) induced by RELABEL and PUSH still have a simple path to \( s \).

**Initialization** At the very beginning of the algorithm, we simply send the flow from \( s \) on height \( |V| \) to its neighbors \( N_s \) on height 0 (because every vertex except \( s \) is assigned to height 0 in the initialization). At this point, the only overflowing vertices are those that in \( N_s \). In the residual network there are exactly the same number of edges from \( N_s \) to \( s \), which can be considered as simple paths from overflowing vertices to \( s \), as desired.

**Induction** Suppose there is a simple path \( p \) from \( x \) to \( s \). If \( x \) is relabeled, nothing is changed except for \( x.h \). So \( p \) is still a simple path from \( x \) to \( s \). If there is a PUSH at \( x \), then \( x \) itself is still connected to \( s \) by its previous simple path.

We also need to show that any \( x \)’s downhill vertex that becomes overflowing also has a simple path back to \( s \). Let’s say the PUSH affects this path \( x \rightarrow u \rightarrow \cdots \rightarrow v \). By the definition of residual network, as long as \( x \) pushes flow along \( u \rightarrow \cdots \rightarrow v \), there must be a path \( v \rightarrow \cdots \rightarrow u \) in \( G_f \). In other words, any new overflowing vertex in the affected path \( u \rightarrow \cdots \rightarrow v \) has a simple path to \( x \rightarrow s \) in \( G_f \), as desired.

4. If we always choose the maximum height overflowing vertex to push, without relabeling, each vertex can have at most one non-saturating push, because it is not overflowing anymore after one non-saturating push. As there are \( |V| \) vertices, there are at most \( |V| \) non-saturating pushes between two relabel operations. After a relabel, some vertices may have a non-saturating push again. By Corollary 26.21 (Bound on relabel operations), there are less than \( 2|V|^2 \) relabel operations. Therefore, there are at most \( 2|V|^3 \) non-saturating push operations.