(20 points) Is $L = \{ u2v | u, v \in \{0,1\}^* \land u \neq v \}$ a CFL? Note that $011120111 \in L$, $0111120111 \in L$ and $010120111 \in L$. Justify your answer. Hint: In what ways can $u$ be different than $v$?
$L = \left\{ u2v \mid u, v \in \{0,1\}^* \land u \neq v \right\}$ is a CFL. We note that there are two ways for $u \neq v$:

- if $|u| \neq |v|$, or,
- if $u = u_1 \ldots u_m$ and $v = v_1 \ldots v_n$ and $u_k \neq v_k$ for $k \leq \min(m, n)$.

These conditions are not exclusive. We first give a CFG to generate $L_0 = \left\{ u2v \mid u, v \in \{0,1\}^* \land |u| \neq |v| \right\}$. We then give a PDA to accept $L_1 = \left\{ u_1 \ldots u_m 2v_1 \ldots v_n \mid u_1, \ldots, u_m, v_1, \ldots, v_n \in \{0,1\} \land (\exists k)(1 \leq k \leq \min(m, n) \land u_k \neq v_k) \right\}$.

Finally, we appeal to the fact that CFLs are closed under union to establish that $L = \left\{ u2v \mid u, v \in \{0,1\}^* \land u \neq v \right\} = L_0 \cup L_1$ must be a CFL.

We can generate strings in $L_0$ by generating all strings $u, v$ with $|u| < |v|$ and then with $|u| > |v|$. In the following grammar, $A$ generates all strings $u2v$ with $|u| = |v|$, and $B$ generates all nonempty strings over $\{0, 1\}$.

$S \rightarrow AB | BA$
$A \rightarrow 0A0 | 0A1 | 1A0 | 1A1 | 2$
$B \rightarrow 0B | 1B | 0 | 1$

We accept $L_1$ with PDA $M = \left( \{ s, q_0, q_1, p_0, p_1, r, t \}, \{ 0, 1, 2 \}, \{ \perp, 0, 1 \}, \delta, s, \perp, \{ t \} \right)$ which accepts by final state. State $s$ counts until $k$ by pushing $k-1$ 0’s onto the stack. Eventually it “guesses” that it is reading a $u_k$ such that $u_k \neq v_k$ by effectively storing $u_k$ in a register and jumping to state $q_{u_k}$.

$$
\begin{align*}
\delta(s, 0, \perp) &= \{(s, 0 \perp), (q_0, \perp)\} \\
\delta(s, 0, 0) &= \{(s, 00), (q_0, 0)\} \\
\delta(s, 1, \perp) &= \{(s, 0 \perp), (q_1, \perp)\} \\
\delta(s, 1, 0) &= \{(s, 00), (q_1, 0)\}
\end{align*}
$$

From state $q_{u_k}$, we consume 0’s and 1’s until encountering a 2, in which case we enter state $p_{u_k}$.

$$
\begin{align*}
\delta(q_0, 0, \perp) &= \delta(q_0, 1, \perp) = \{(q_0, \perp)\} \\
\delta(q_1, 0, \perp) &= \delta(q_1, 1, \perp) = \{(q_1, \perp)\} \\
\delta(q_0, 0, 0) &= \delta(q_0, 1, 0) = \{(q_0, 0)\} \\
\delta(q_1, 0, 0) &= \delta(q_1, 1, 0) = \{(q_1, 0)\} \\
\delta(q_0, 2, \perp) &= \{(p_0, \perp)\} \\
\delta(q_0, 2, 0) &= \{(p_0, 0)\} \\
\delta(q_1, 2, \perp) &= \{(p_1, \perp)\}
\end{align*}
$$
\[ \delta(q_1, 2, 0) = \{(p_1, 0)\} \]

In state \( p_u \) we consume \( v_1...v_{k-1} \) by popping \( k-1 \) 0’s off the stack. Once \( v_1...v_{k-1} \) has been read (is on the top of the stack), \( p_u \) compares \( v_k \) to \( u_k \). If \( v_k \neq u_k \), then the machine enters state \( r \), which consumes the rest of the input.

\[ \delta(p_0, 1, \bot) = \{(r, \bot)\} \]
\[ \delta(p_1, 0, \bot) = \{(r, \bot)\} \]

In state \( r \) we assure that the unconsumed part of the input string is all 0’s and 1’s, and then we enter the final state \( t \).

\[ \delta(r, 1, \bot) = \delta(r, 0, \bot) = \{(r, \bot), (t, \bot)\} \]