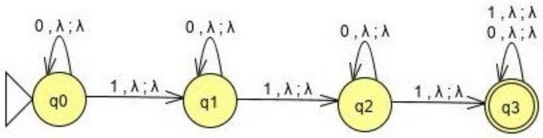
CSE 355: Intro to Theoretical Computer Science
Recitation #8 Solution

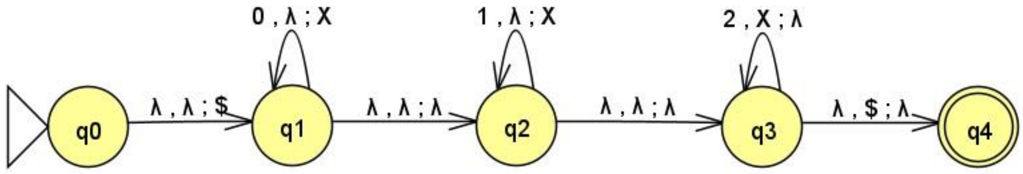
Use JFLAP (https://www.jflap.org/jflaptmp/, download JFLAP7.1.jar) to draw the state diagram of the PDA (pushdown automata) that recognize the following languages.

1. [5 pts] L = $\{\omega \in \{0,1\}^\star|$ ω contains at least three 1s$\}$

This is a regular language, and we can convert any NFA/DFA into a PDA without using the stack.

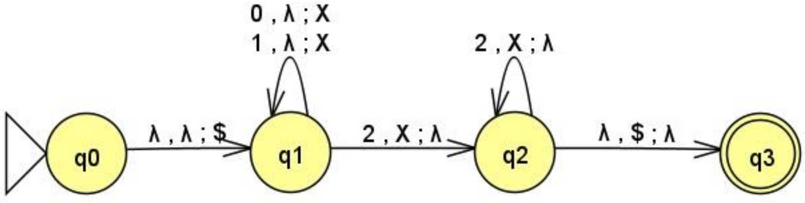


2. [5 pts] L = $\{0^i1^j2^k \mid i,j,k \geq 0 \; and \; i+j = \; k\}$

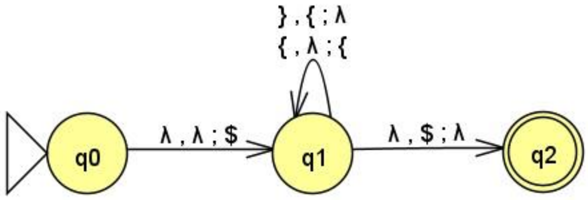


Note: above PDA guarantee to accept 0s first, whenever it sees a 0, it pushes an X on top of the stack as a counter, it then accepts only 1s; similarly, whenever it sees a 1, it pushes an X on top of the stack. Later when it sees a 2, it pops an X from the stack to match it until the input string is finished and the stack is empty, it goes to accepting state q4.

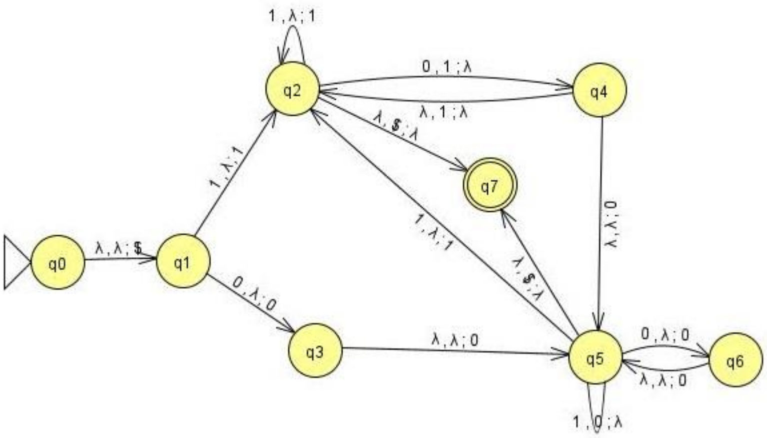
The following PDA is WRONG since it accepts 100102 which is not inside the language.



3. [5 pts] L = $\{all\ string\ of\ properly\ balanced\ left\ and\ right\ brackets\}$. Such as { }, { }{ }, {{ }}, { { }{ }}, etc.



4. [5 pts] $L = \{\omega \in \{0,1\}^* | \mid \omega\ has\ twice\ as\ many\ 1's\ as\ 0's\}$ (Note: this is a little complicated. Use sample strings, such as 011, 101, 110, 001111, 111100, 010111 or 101101, etc to check your PDA's state diagram)



Note: This is a little complicated PDA. The key idea is to match every 0 from the input string with two 1s. In case we see a 0 from the input string, we need to push two 0s on top of stack in order to match two 1s later on from the input string.

At state q1, if we see a 1 from the input string, just push it on top of the stack and go to state q2. State q2 is used to handle the case where we have 1 on top of the stack. From here we need to consider two sub-cases, 1) where there's a 1 from the input string, for this case, we simply just push this 1 on top of stack again; 2) where there's a 0 from the input string, for this case we need to pop two 1s from the stack, this is done through state q4. We also need to consider a special case where there isn't enough 1 (i.e. just one 1) from the stack to match the input 0 (think about string 101), for this special case, instead we will push one extra 0 on top of the stack and goes to state q5 which will match a 1 later from the input string.

At state q1, if we see a 0 from the input string, first push it on top of the stack and go to state q3, then without consuming any input string, we push another 0 on top of stack and go to state q5. State q5 is used to handle the case where we have 0 on top of the stack. From here we also need to consider two sub-cases, 1) where there's a 1 from the input string, for this case we simply pop one 0 from the stack to match it (the loop), or 2) there's a 0 from the input string, for this case we need to push two 0s on top of the stack, this is done through state q6. There's also a special case we need to consider: at state q5, there's a 1 from the input string, but stack is empty, i.e. no 0 to match it, for this case, we push the 1 on top of the stack and goes to state q2.

At state q2 and q5, if we finish reading the input string and there's nothing left on stack, we go to accepting state q7.