a. What is the probability of generating the string *cca* on the following probabilistic automaton? (Show your computation.)

\[ I_A(q_0) = 1 \\
\] 
\[ i \neq 0, I_A(q_i) = 0 \]

Two paths for *cca*: 
- Path 1: \(q_0-q_0-q_1\), probability, \(P_1 = 1 \times (1/4) \times (1/4) \times (1/8) \times (1/5) = 1/(2^7 \times 5)\)
- Path 2: \(q_0-q_0-q_2\), \(P_2 = 1 \times (1/4) \times (1/4) \times (1/2) \times 1 = 1/(2^5)\)

Total probability = \(P_1 + P_2\)

Max by path with \(P_2\).

b. What is an optimal path for generating this string *cca*, and why?

c. Define non-deterministic probabilistic finite state automaton.

d. Is epsilon-PFA equivalent to deterministic PFA?  Yes / No

The PFA definition (Definition 1, section 2.2, page 1015, paper-1) is for non-deterministic probabilistic automaton. Deterministic-PFA is a specialized one defined in Section 2.4 there).

e-PFA or lambda-PFA is equivalent to PFA by Proposition-2 (page 1016 bottom).

e. A Kullback-Leibler (KL) divergence measure is ..............-based measure.

f. Define KL divergence measure.

KL divergence is an “entropy” based measure of difference between two probability distributions, defined in Section 6.2 paper-1 page 1021.

g. The following diagram proves by contradiction that \(L_u\) is non-recursive, given that \(L_{d\_bar}\) is so, and that \(L_t\) is recursive. Can you restructure this proof to show that \(L_{d\_bar}\) is recursively enumerable, provided that we know \(L_u\) is so?

Just remove the “No” arrow from M’ below (because Lu is known to be r.e.), and the following whole box becomes a “correct” TM for \(L_{d\_bar}\).
Part-2

\[ \Sigma = \{0, 1\} \]
\[ L = \text{ww}^r \text{ where } w^r \text{ is reverse of the string } w \text{ on } \Sigma^* \]

h. If it is a regular language write a finite automaton. If not, then prove that by the corresponding Pumping Lemma.

No, it is not a regular language.
PL proof: Assume finite state machine \( M \) of size \( n \), and choose a string from \( L \) of size \( 2n \). Loop within prefix of size \( n \) (\( w \) part), its repetition cannot be matched with \( w^r \) part in the new string expected to be accepted by \( M \) as per PL.

i. Write a CFG for the language.

\[ S \rightarrow 0S0 | 1S1 | \epsilon \]

j. Write a PDA for the language.

From slide:

- Example PDA #2: For the language \( \{x \mid x = \text{ww}^r \text{ and } w \in \{0, 1\}^*\} \)
  
  Note: length \( |x| \) is even

\[ M = \{(q_0, q_1, \{0, 1\}, \{\#, B, G\}, \delta, q_0, \#) \}
\]

\[ \delta: \]

(1) \( \delta(q_0, 0, \#) = (q_1, B\#) \)

(2) \( \delta(q_0, 1, \#) = (q_1, G\#) \)

(3) \( \delta(q_1, 0, B) = (q_2, \text{BB}) \) \( \delta(q_1, 1, G) = (q_2, \text{GG}) \)

(4) \( \delta(q_1, 0, G) = (q_2, \text{BG}) \)

(5) \( \delta(q_1, 1, B) = (q_2, \text{GB}) \)

(6) \( \delta(q_1, 0, \#) = (q_2, \#) \)

(7) \( \delta(q_1, 1, \#) = (q_2, \#) \)

(8) \( \delta(q_2, 0, \#) = (q_3, \#) \)

(9) \( \delta(q_2, 1, \#) = (q_3, \#) \)

(10) \( \delta(q_3, \epsilon, \#) = (q_4, \#) \)

Notes:
- Rules #3 and #6 are non-deterministic: two options each
- Rules #9 and #10 are used to pop the final stack symbol off at the end of a computation.

k. Write a Turing Machine for the language.

Logic:
Use Roger’s machine to delete from both ends, checking if the characters match at two ends.
| \( q_2 \) | \((q_3, B, L)\) | \( (q_3, B, L) \) | - | // only 0 |
| \( q_4 \) | - | \((q_3, B, L)\) | - | // only 1 |
| \( q_3 \) | \((q_3, 0, L)\) | \((q_3, 1, L)\) | \((q_0, B, R)\) |
| \( q_4 \) | - | - | - |