PROBLEM 1. Give two branching programs B_1 and B_2 that agree on all Boolean input but have different output polynomials when arithmetized. (Try to find a simple example.)

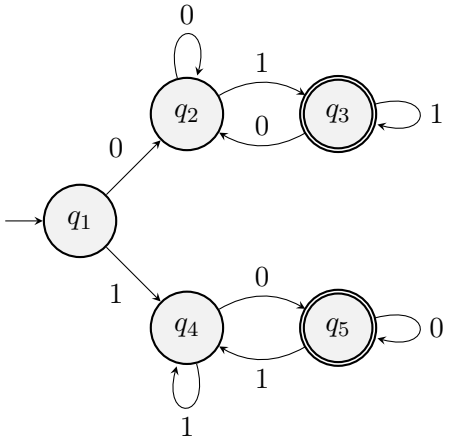
PROBLEM 2.

(a) Prove that if A is a regular language, a family of branching programs $(B_1, B_2, \ldots)$ exists where each B_n accepts exactly the strings in A of length n and is bounded in size by a constant times n. It may help to describe your BP one level at a time. (Take the size of a BP to be the number of nodes plus the number of edges.)

  (i) Describe the construction of each B_n.

  (ii) Prove that the size of B_n is $O(n)$.

(b) Illustrate your proof by creating B_3 for the regular language generated by the following DFA:



PROBLEM 3. The *parity function on n inputs* is the Boolean function that returns 1 if the number of 1s in the input is odd, and returns 0, otherwise. Show that the parity function with n inputs can be computed by a branching program that has $O(n)$ nodes by drawing the example when $n = 4$ in a way that clearly generalizes to arbitrary n.

PROBLEM 4. Here is Chat GPT's proof that EQBP is coNP-complete. I asked it to give a reduction of the complement of SAT to EQBP. Is the proof correct? If not, can you fix it?

Proof. Given a Boolean formula F, let's construct two branching programs P and Q as follows:

For each clause C in F, create a branching program that accepts all assignments that satisfy C. We can construct such a branching program using the standard technique of creating a binary decision tree with the variables of C as internal nodes and the literals of C as leaf nodes.

Combine the branching programs for all clauses using a binary AND operation, resulting in the branching program P that accepts all satisfying assignments of F.

Negate each literal in the branching programs for all clauses, and then combine the branching programs for all clauses using a binary OR operation. The resulting branching program Q accepts all non-satisfying assignments of F.

Finally, check whether P and Q are equivalent using EQBP.

The construction of P and Q takes a polynomial number of operations, as each clause has a polynomial number of variables, and each variable has a polynomial number of possible values. Thus, the total number of branching program nodes is polynomial in the size of the input formula. The EQBP problem can also be solved in polynomial time, so the entire reduction can be done in polynomial time.

Now, if F is unsatisfiable, then P and Q will be equivalent, as there are no satisfying assignments for F, and thus P and Q will both accept no inputs. Conversely, if F is satisfiable, then P and Q will not be equivalent, as there exists at least one satisfying assignment for F, which is accepted by P and not by Q.

Therefore, we have shown a polynomial-time reduction from the complement of SAT to EQBP, which implies that EQBP is at least as hard as the complement of SAT, and hence EQBP is coNP-complete. $\square$