1. **Show that NP is closed under union:**

   Assume we have two TMs that decide our languages A and B, M_A and $M_B \in$ NP.

   $M_{A \cup B}$: on w:

   1. run M_A on w

      if M_A accepts, accept

   2. run M_B on w

      if M_B accepts, accept

   3. reject

   since we know step 1 $\in$ NP and step 2 $\in$ NP, our TM runs in c*n + 2 * NP which $\in$ NP. We also know that this will give us the proper answer because all we need to make sure is that one or both of our machines accept or else we reject.

   Show that NP is closed under concatenation:

   Assume we have two TMs that decide our languages A and B, M_A and $M_B \in$ NP.

   M = "on input w:

   1. Nondeterministically split w into two parts such that w_1 w_2 = w

      2. run w_1 on M_A. If M_A rejects, reject

      3. run w_2 on M_B. If M_B rejects, reject

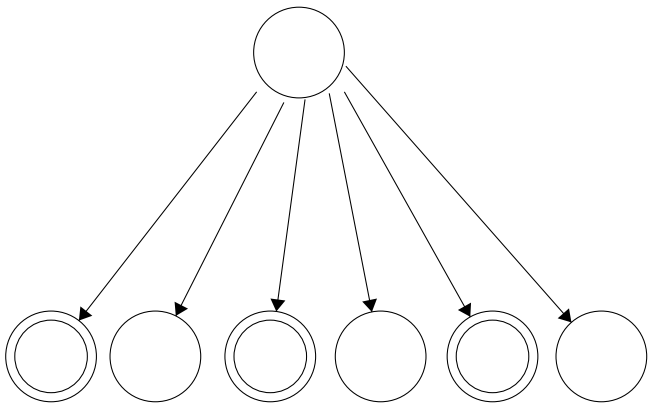
   4. else: accept.

   This solves our problem as it will try every permutation of w where w_1 w_2 = w. Therefore step 1 will be O(n). Step 2 and 3 are $\in$ NP as given in the question. Therefore this construction is $\in$ NP because c * n * 2 * n^k which is $\in$ NP. This also properly finds the solution because we will try every possible split of w so if the string w is a concatenation of A and B we will eventually find it in polynomial time.

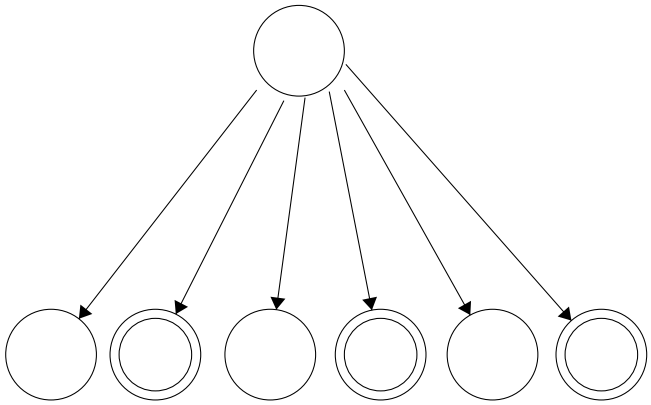
   Show that it is not know whether NP is closed under complementation:

   i. A NTM will have many branches and making the deterministic machine run in exponential time. The deterministic TM will not run in polynomial time because at each decision in the NTM, the NTM can make k splits allowing for exponential amount of different paths that the deterministic TM will have to solve one at a time creating an exponential amount of work for it.

   ii. This construction isn't quite right. The machine detailed accepts if there is any accepting branch which creates a problem for us.



Now if we flip the accept and reject states as detailed in the construction are paths will reflect this:



Before we flipped the accept and reject state, we would accept because one of the tree branches accepted. After flipping the accept and reject state we still accept because there is still an accepting path. This is because there are a combination of branches that accept and reject. For this construction to work we would need all branches to accept or reject which we cannot assume will be the case.