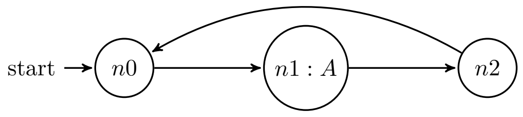
1 Task

I have implemented the nested DFS algorithm in the function “hasCycle” as requested.

2 Questions

1. The check in the blue_dfs is not sufficient, since it only checks the loops where the accepting state is at the first or last nodes of the lasso. An example of a graph where this check would not be sufficient (only the node marked with A is accepting):



   For this counter example, the algorithm will call the blue_dfs on $n0$, and color all the nodes cyan by order. At this point, the cycle will not be found by the blue check, and we will not encounter this check again (even if the code would not stop after the red check fails), which proves that it is not sufficient. Then it will start to retreat, and color node $n2$ blue since it is not accepting. Next, since $n1$ is accepting, it will call the red_dfs on it, color $n2$ red, and find that $n0$ is cyan, which will report the cycle.

2. The check in the red_dfs is sufficient. It will catch all cycle coughed by the check in the blue_dfs:

   - If s is accepting, red_dfs will be called on it and will find the cyan t, so the cycle will be found and reported.
   - Otherwise, after the blue_dfs retreats back to t, it will call the red_dfs on it while it is still cyan, find the loop going through s back to t, and report the cycle.

3. The check can be easily added without any significant affect on complexity, and in the cases where it reports a cycle, it eliminates the need to call the red_dfs on those nodes, reducing the complicity significantly in some cases.