Final Exam

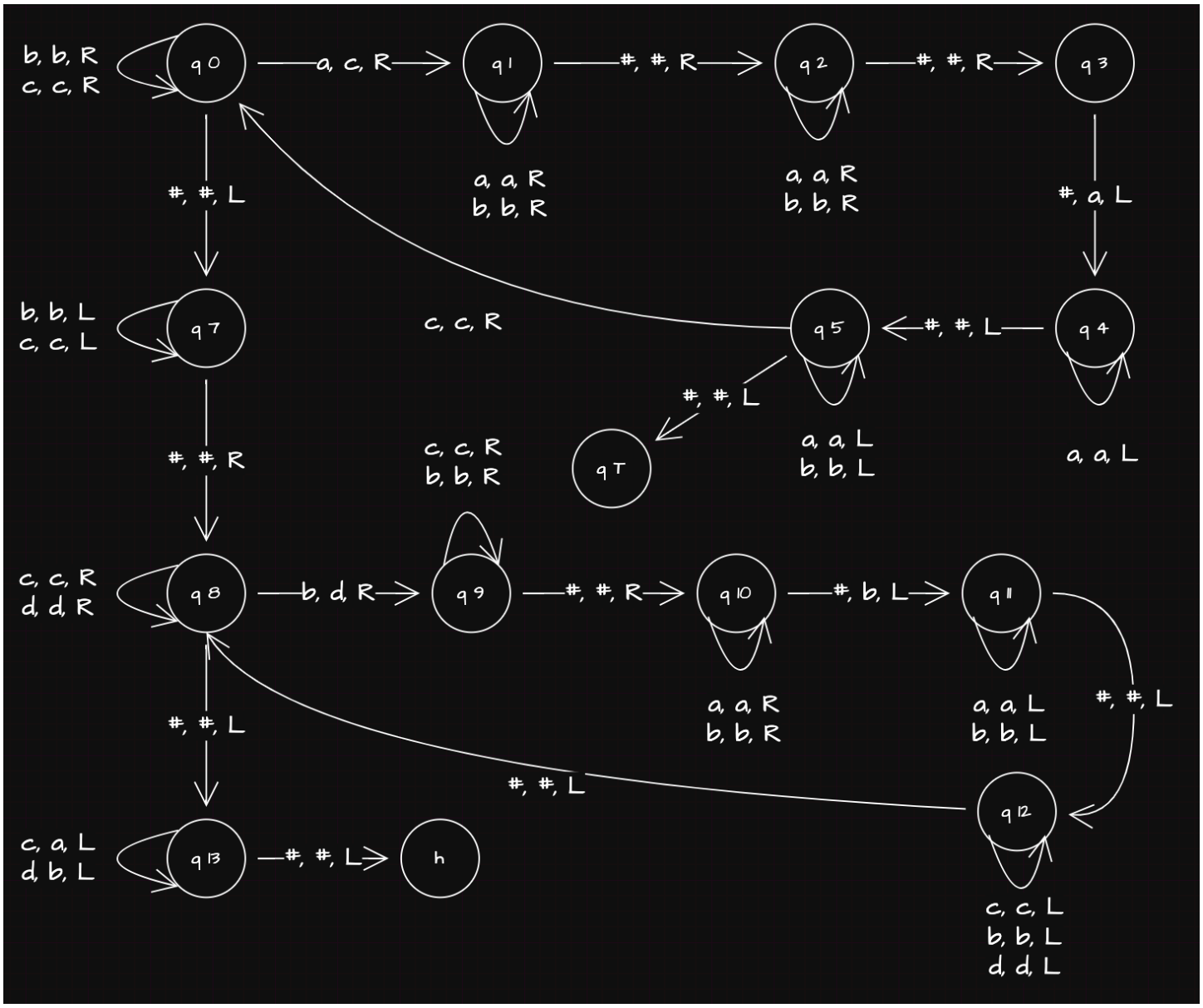
1. Given Σ = {a, b}, Let ORDER be an operation such that given any input string w, rearrange w into the form $a^n b^m$. For example, ORDER(ababbab) = aaabbbb. On the turning machine tape, the tape after the last step should look like …###ababbab#aaabbbb###... (15)

    a. Describe an algorithm for a Turing Machine to implement this operation (10)

For each 'a', replace with 'c' and go to right and place 'a'. go back to beginning until all 'a' are replaced with 'c' and written 'a' on the right.

Do the same with 'b' and 'd'. After go to the middle '#' and go left replacing ['d' with 'b'] and ['c' with 'a']

    b. Turn your algorithm into a Turing Machine diagram (5)



Question 1 *Visualization* (TM pointer at underline)

q0 -> q7 : if no ‘a’ until ‘#’

q0 -> q1 : ###ababbab##...

q1 -> q2 : ###cbabbab##...

q2 -> q3 : ###cbabbab###...

q3 -> q4 : ###cbabbab#a##...

q4 -> q5 : ###cbabbab#a##...

q5 -> qT : if all ‘a’ are converted to ‘c’ && n(b) == 0

q5 -> q6 : ###cbabbab#a##...

repeat : ###cbcbbcb#aaa##...

q0 -> q7 : ###cbcbbcb#aaa##...

q7 -> q8 : ###cbcbbcb#aaa##...

q8 -> q9 : ###cdcbbcb#aaa##...

q9 -> q10 : ###cdcbbcb#aaa##...

q10 -> q11 : ###cdcbbcb#aaab##...

q11 -> q12 : ###cdcbbcb#aaab##...

q12 -> q8 : ###cbcbbcb#aaab##...

repeat : ###cdcddcd#aaabbbb##...

q8 -> q13 : ###cdcddcd#aaabbbb##...

q13 -> h : ###ababbab#aaabbbb##...

This took me so long [T_T]

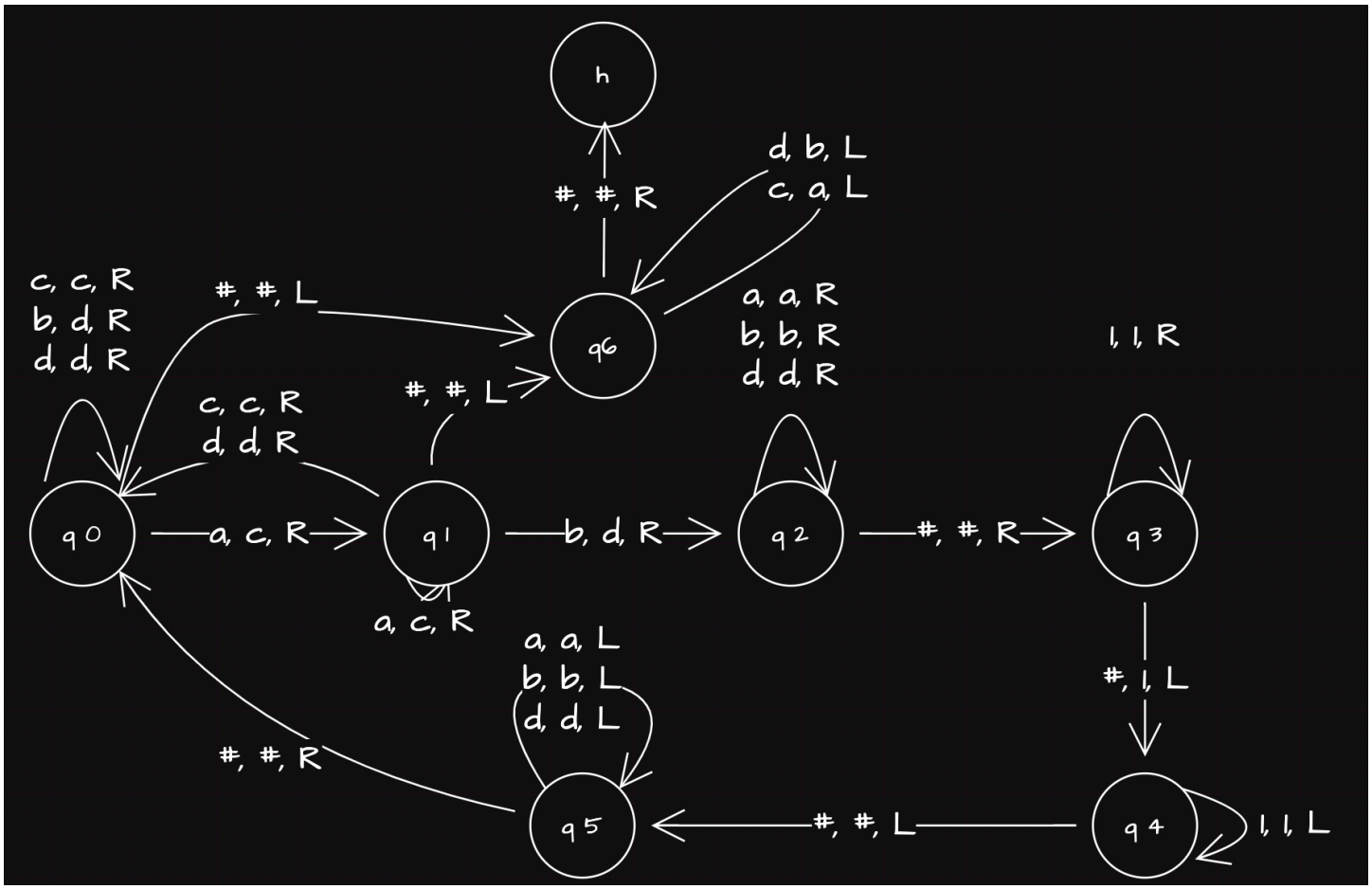
2. Given Σ = {a, b}, Let COUNT_AB be an operation such that given any input string w, count the number of substring ab in w using unary number. For example, COUNT_AB(aabbabaab) = 111. On the turning machine tape, the tape after the last step should look like …### aabbabaab#111###... (15)

   a. Describe an algorithm for a Turing Machine to implement this operation (10)

   Find 'ab' using a middleman state, to not repeat 'ab' replace 'ab' with 'ad'

   go to the '#' and keep going until '#' replace it with '1'. go back to very beginning and repeat.

   b. Turn your algorithm into a Turing Machine diagram (5)



3. Let 3-SIZE be this problem: Given Turing Machine TM, input string w, will w has more symbols than the number of transitions in TM? Explain (informal proof) why this problem is decidable. (12)

We count a number of symbols inside the input of the string until the string ends. From there we scan, while incrementing the count of each symbol, which upon ending is where this process halts. the Turing Machines sets of a finite set of states, a finite Alphabet, finite tape alphabet & a transition function. The transition function is a finite set of rules. we can count the number of transitions with z. This is a finite number determined by the machine's description. we compare the counts between |w| and |z|.
Since we are counting transitions in the machine's description, and the courses can be done with an algorithm that halves and gives a yes or no, we can say that the 3-SIZE problem is decidable..

4. Let $STATE_{TM}$ be this problem: Given Turing Machine TM, input string w, and a specific state q in TM, will TM go through state q on input w? Prove $STATE_{TM}$ is undecidable. (12)

Assume q state(TM) can decide whether TM has q state in it

new_q_state(T):

if q_state(T) == true, return false

else return true

Run new_q_state(new_q_state);

Proof

Assume Halt(TM, w) exists such that halt can decide whether input w on TM will halt

new_halt(TM):

if halt(TM, w)== true, loop forever

else halt

Run new_halt(new_halt);

5. Let $EMPTY\_TAPE_{TM}$ be this problem: Given Turing Machine TM, input string w, will TM produce an empty tape (not start with an empty tape)? Prove $EMPTY\_TAPE_{TM}$ is undecidable. (12)

An empty turing machine is a turing machine that, when language is inputted, returns ∅

Inputting input w into Turing machine returns n. if n != w, reject. otherwise simulate a turing machine with input w.

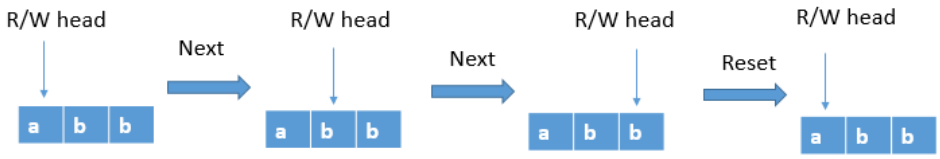
Turing Machine 'z' if Tm 'X' accepts, rejects, else accepts. This is contradictory.

6. Let $SAME\_TAPE_{TM}$ be this problem: Give Turing Machine TM1, TM2, input string w, will TM1 and TM2 produce the same content on their tape after they are done? Prove $SAME\_TAPE_{TM}$ is undecidable. (12)

If you travel back in time and prevent your grandfather from meeting your grandmother, you would never be born to make that trip, creating a contradiction.

Simulate Turing Machine A with input w and also Simulate Turing Machine B with input w. CMP and Return A && B

7. Imagine a machine that is very similar to Turing Machine, except it can only move the read/write head to the right unless a reset command is issued, where the read/write head will reset to initial position. For example:



Explain (informal proof) if this machine is more or less powerful than a standard Turing Machine. (12)

This new machine is more powerful than the Turing Machine. For certain cases, using such a machine will greatly reduce complexity. For n('ab') as unary, or w -> $a^n b^m$ for example, getting a circular tape which goes back to the very beginning, will have a faster, simpler and easier to understand approach.
This is the case and only the case if you are able to reset after as many '#' at the end of the tape as you want suppose the tape is 'ababbab' this will not be practical unless the tape has the ability to reach 'ababbab###...'

In terms of efficiency, such a machine will be less efficient. In terms of coding, using an indexer with a string searcher is more efficient in most cases. Such as an array with a pointer is generally better than a linked list.

8. Assume you have relatives that just learned that you are taking CS courses in school. The relatives want to share some of their latest AI startup ideas. Decide theoretically whether these ideas are feasible or infeasible (10):

   a. "An AI that checks if one book plagiarize another book"

This is feasible. It goes through the first book and compares if it has similar wording and sentence structure as the 'another' book.

   b. "An AI grading website where student upload their program and it checks against the teacher's solution to see if they work the same way"

It is possible to create an ai, use a hose, to run the programs and check if they output the same solution.

   c. "An AI tool that can decide whether a program people submit will run to the correct outcome given user input. Saving time on software testing."

Yes, this may not even have to be an ai. you can create a batch file or an input stream file with possible user inputs. run and save the information about the output, if it is correct / where it got wrong. Then write a report with the test cases.

   d. "An AI that checks other AIs, and produce an evaluation on the upper bound of accuracy the other AI can achieve. Makes it easier to evaluate chatGPT vs DeepSeek".

Infeasible. such a feat would be incredibly difficult. These ai's are always evolving, and feeding infinite amounts of information, it is not within our budget to match / surpass their learning and evolution speeds.