Homework Assignment 1 - Grading Guidelines

All questions were worth 10 points, except question 5 (20 points total) and question 10 (30 points total). You will have until the end of office hours on Wednesday, Oct 11 to ask for a regrade for *specific* questions. If you would like a regrade, please come to one of the *TA* office hours with your question ready and be ready to explain/defend your answer. The TA has the right to ask you to qualify your answer, or defend your reasoning beyond what is written on the page. **The TAs reserve the right to *lower* your grade during a regrade.**

1. **Solve problem 0.5 on page 26 of [Sipser, Second Edition].**
   Question: If C is a set with c elements, how many elements are in the power set of C? Explain your answer.

2^c. The easiest way to do this is by binary encoding. If you have c elements, you can describe each subset as either containing the element, or not containing it. This means you can write a bit string where each bit represents an element of C. This string has length c, so you can represent 2^c numbers, or 2^c sets. We were looking for a description of a binary choice with each element.

Grading comments:

- -4 for not giving proof, but only citing function.
    - For future reference, if you are asked to prove a definition/characteristic of something (such as the cardinality of a certain set), the book definition is not the answer. You must use a proof/explanation. If there is a *proof* in the book, you are welcome to use it or paraphrase. The same goes for all proofs used in class. For example, if you are proving the countability of the set of rational numbers (which you probably proved in class), you can say “The set of rational numbers is countable, which we proved in class, because you can line up the elements of this set in a way in which it is possible to use dovetailing.” Please give atleast one sentence if you cite a proof to show to us that you understand why the proof works.
- -2 for explaining that there were 2^c combinations, but not explicitly stating why. Again, we are looking for the explanation of a binary choice on including elements.
- -3 for using a pattern in lieu of a proof
    - For future reference, patterns are not proofs. Technically, they don’t even count as induction (you shouldn’t ever need to use induction in this course). While it is good that you recognize this pattern is occurring, and you can use it as support to your reasoning, you need to explain the *causation* of the pattern.
    - If you elaborated on why the cardinality doubled with every increase in the cardinality of C, we gave full credit
- -1 for explaining that the set could be represented using binary sequences, but did not explain how the binary sequences correlate to the set or its elements

- Full credit for using combinatorics
- -1 for induction without a base case explanation

2. **Solve problem 0.12 on page 27 of [Sipser, Second Edition].**
   Show that every graph with 2 or more nodes contains two nodes that have equal degrees.

For a graph of N nodes, every node's degree is between 0 and N-1. Since the number of possible degrees is equal to n, in order to have unique degrees, each node must be one of these degrees (between 0 and n-1). A node with degree 0 is isolated from every other node, while a node of degree N-1 is connected to every other node, so a graph cannot have nodes with both degrees. Therefore, there are at most N-1 possibilities for the degree of each node. By the pigeonhole principle, at least two nodes must have the same degree.

Comments:
- -2 for not recognizing that a node can have degree 0
- Full credit for not explicitly explaining the doubling up (pigeonhole principle) of degrees.

3. **True or false: a countable union of countable sets is countable.**

   True. Proof by dovetailing.
   If we define this union as:
   $S = S_1 \cup S_2 \cup S_3 \cup \ldots\ldots \cup S_n$ *note that n can be a finite number or infinity
   And set up the dovetailing as:
   $S_1 = \{a_1, a_2, a_3\ldots\ldots a_n\}$
   $S_2 = \{b_1, b_2, b_3\ldots\ldots b_n\}$
   $S_3 = \{c_1, c_2, c_3\ldots\ldots c_n\}$
   ….
   ….
   $S_n = \{d_1, d_2, d_3\ldots\ldots d_n\}$

   Then you can use dovetailing to define S as:
   $S = \{a_1, b_1, a_2, c_1, b_2, a_3\ldots.\}$ etc

   (There is also another proof using Set Theory - Inclusion and Exclusion principle)
   If a set A is countable then n(A) (number of elements in A) exists.
   So. $n(S_1 \cup S_2 \cup S_3 \cup \ldots\ldots \cup S_n) = \Sigma\, n(S_1) - \Sigma\, n(S_1 \cap S_2) + \Sigma\, n(S_1 \cap S_2 \cap S_3) - \ldots\ldots$ where every element exists, Hence n(union) exist. Hence countable.

   Grading Comments:
   - -4 for choosing first element of each set, followed by second element of each set, etc...

        - You can't ever choose the second element of the first set if there are an infinite number of set because you will never reach the first element of the last set.
    - Full credit for wrong terminology with correct execution--full credit if they use the word "diagonalization" if they actually implement dovetailing
    - -2 if the proof only showed a union of two sets but *could* be expanded to an infinite union.
    - -4 if the proof only showed a union of two sets and *could not* be expanded to an infinite union, or if in doing so, they made an error.
    - -6 for no proof or incomprehensible proof
    - -2 for unclear ordering/dovetailing procedure
    - -4 for using finite instead of countable.

4. **True or false: if T is countable, then the set {S | S ⊆ T, S is finite} is also countable.**

True. Also, a proof by dovetailing. **It is not true *simply* because S is finite, because we are talking about the set of *all S's*.** S being finite is the key to this problem, but not the solution. If S were not finite, this would be the powerset of T which is uncountably infinite (because T is countably infinite). (You need to learn set builder's notation)
I'll call this set Q.

Q = T_1 U T_2 U T_3 U ……U T_i <--each subscript is the cardinality of S, so T_1 contains all S's with one element.

We can use the proof from number 3 if we also say that each T_i must be countable because each set in T_1 is a countable set because S is finite. Thus, T_1 is a countable union of countable(finite) sets. Then Q must also be countable, because it is a countable union of countable sets (T's).

https://www.mathsisfun.com/sets/set-builder-notation.html
Note about set notation: if you have {} surrounding something, it can be useful to rewrite with another label. So, if we have the set {S | S ⊆ T, S is finite}, we can let Q = {S | S ⊆ T, S is finite} and refer to the set Q, which contains all possible S, such that each instance of S is finite.

Grading Comments:
- -5 for stating S is finite, thus countable or another misinterpretation of the question
- -2 for sorting by size because there are infinitely many of each size
- -4 for a flawed proof that would allow the power set of T to be countable.
    - **The fact that S is finite is significant.**

5. **What is the cardinality of each of the following sets? [20pts, 5 pts for each part]**

- -3 for no proof with correct answer
- -5 for wrong answer with no proof
- 1 point for wrong answer with attempted proof

a. The set of all possible Java programs

Countable and Infinite.

One possible explanation

Countable :

Consider the Alphabets of the java language be L. The strings of the language with thus be Σ*. The input to a Java compiler is a Java program which is nothing but a string. So if the string is compiled, then the string is a java program and thus we can increase the count by 1. If not we disregard it and move to the next string.

Infinite : Consider a string within a valid Java Program - > System.out.println("Hello");
This line can be duplicated, and concatenated after this line. As a result, a new program emerges. Thus we can add do this infinite times and thus, the number of Java Programs are infinite.

b. The set of all finite strings over the alphabet {0, 1, 2}

Countable and infinite. This is essentially Σ*. You can iterate through the regular expression(0U1U2)* and list all possibilities while counting them. This is because we can determine this regular language as an DFA and every string of the DFA is countable.

All you had to say was produce strings in a lexicographic order.

Comments:

- -2 for unclear explanation

c. {Ø, N, Q, R}

Four. This set is finite, because it is a set *that contains* R -- R is not a subset of this set. Don't get your types mixed up. (Sets of reals versus Sets of Sets)

Comments:

- -4 for uncountably infinite
- Full credit for using "4" as explanation
- -1 for no explanation, since the explanation is simple

d. R - Q

Uncountable. R is uncountable and Q is countable. If R - Q were countable, then the union of R - Q and Q would also be countable (countability is closed under union).

Ie S = R - Q.

Then S U Q = R.

If S is countable and Q is countable, the result should be countable, but R is not.

Grading comments:

- Full credit for saying this is equal to the set of irrational numbers (Which is uncountably infinite)
- Full credit for demonstrating an adequate understanding that uncountable infinities are far larger than countable infinities
- -2 for the right answer with an unclear proof.
- -2 for a restatement of the problem.

6. **Prove without using induction that n^4-4n^2 is divisible by 3 for all $n \geq 0$.**

This can be factored as $f(x)=n^2(n+2)(n-2)$. If f(x) is divisible by 3 then either n^2 is divisible by 3 or (n+2) is divisible by 3 or (n-2) is divisible by 3.

Consider the consecutive numbers k-1,k,k+1. If k is divisible by 3, then (k-1)-2=k-3 is divisible by 3. Also k+1+2=k+3 is also divisible by 3.

k%3={0,1,2} is repetitive but can also be used to solve this question.

Grading comments:

- -3 for only factoring--no explanation why factors must be divisible by 3
- -3 for using a pattern
    - **Patterns are not proofs**
- -1 for missing the explanation of one of the factors

33

7. How many distinct boolean functions on N variables are there? In other words, what is the cardinality of $|\{f \mid f: \{0,1\}^N \rightarrow \{0,1\}\}|$?

2^{2^n}. There are 2^n possible input strings for a function on n variables (0 or 1 for each input, with n inputs). Any boolean function will have different combinations of these for true/false returns.

Power set proof: You can define a boolean function as a set that contains the input strings in which it returns true and it would return false if this string is not a set. ie {01, 11} is the boolean function where 01 → 1 and 11 → 1 and 00 -->0 and 10 → 0. Using this, the set of all boolean functions would be equal to the power set of the set of all inputs (because every subset of the inputs would represent a different bool function). Because the power set has a cardinality of 2^n, the power set of the inputs would be 2^{2^n}.

Another way is to make a chart:
N = 2

00	0	0	0	0	0	0	0	0	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0
10	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1

….

00	1	1	1	1
01	1	1	1	1
10	0	0	1	1
11	0	1	0	1

And explain that each column would represent a separate boolean function. You could also explain this by saying you could encode a boolean function using a bit string such that each bit represents a given input. Like this, you can have 2^n different bit strings, where n is equal to the number of inputs, aka 2^n, so again, we get $2^{2\text{^}n}$

Comments:

- -4 for not adequately demonstrating the understanding the combinations of the inputs mapping to the outputs
- -4 for an answer that did not match the proof
- -7 for 2^n with attempted proof

8. **Prove or disprove: every regular language is countable.**

Every language is a subset of Σ^*, which is a countable set (sort lexicographically). Therefore all languages (including all regular languages) are countable sets.

Comments:

- Full credit for using a DFA as a proof.
    - This overcomplicates things, but is possible.
- -4 for saying that all regular languages are finite (they're not, consider {a*} or Σ^*)
- -2 for saying a DFA could be mapped, but not explaining how

9. **Prove or disprove: the set of all regular languages is countable.**

True. Every regular language has a finite automata which can be defined with a 5-tuple $(Q, \Sigma, \delta, q_0, F)$. This is essentially a Cartesian product of 5 countable things, so it is also countable. You can prove a Cartesian product is countable by using dovetailing.

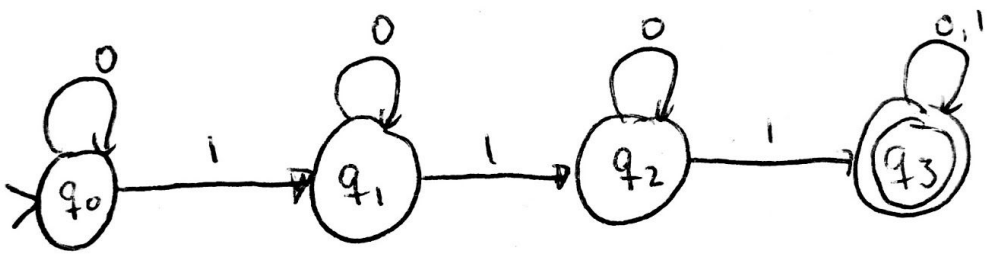
1. Q is countable because it is a finite subset of countable things (states would correspond to natural numbers) See 4 for this reasoning
2. Σ is finite by definition
3. δ is countable because it is a cartesian product of Q (countable) and sigma (finite), so it's countable
4. q_0 is countable because it is finite (has to be an element of Q, which is finite)
5. F is countable because it is finite (finite subset of Q, which is finite)

Grading comments:

- -2 for not showing that the combination of elements of the FA is countable
- -4 for arguing that this is a union of regular expressions (to solve this problem with regular expressions you could prove that every regular language corresponds to an RE, and every RE is a finite string [over the alphabet { | , +, *, (,), symbols }] and an infinite set of finite strings is countable)
- -2 for explaining that the elements of the FA are finite, but not explaining that they are possible finite subsets of a countably infinite set
- -3 for not explaining how to order all DFAs
- -4 for enumerating languages by their cardinality
    - Many regular languages have countably infinite cardinalities
        - How do you order these?
- -4 for assuming regular languages have a finite cardinality
- -7 for assuming the set of all regular languages is equal to the power set of sigma star.
    - The power set of sigma star is uncountably infinite.
- -4 for using the union of all regular languages as a proof
    - This would be equal to sigma star. (since sigma star is regular)
    - The set of all regular languages is a set containing sets. (not a union of languages)
- -2 for using the "length" of a DFA without explaining how they arrived at this length

10. Solve problems on pg 84 of [Sipser, Second Edition] JFLAP [30pts, 10 pts for each part]:

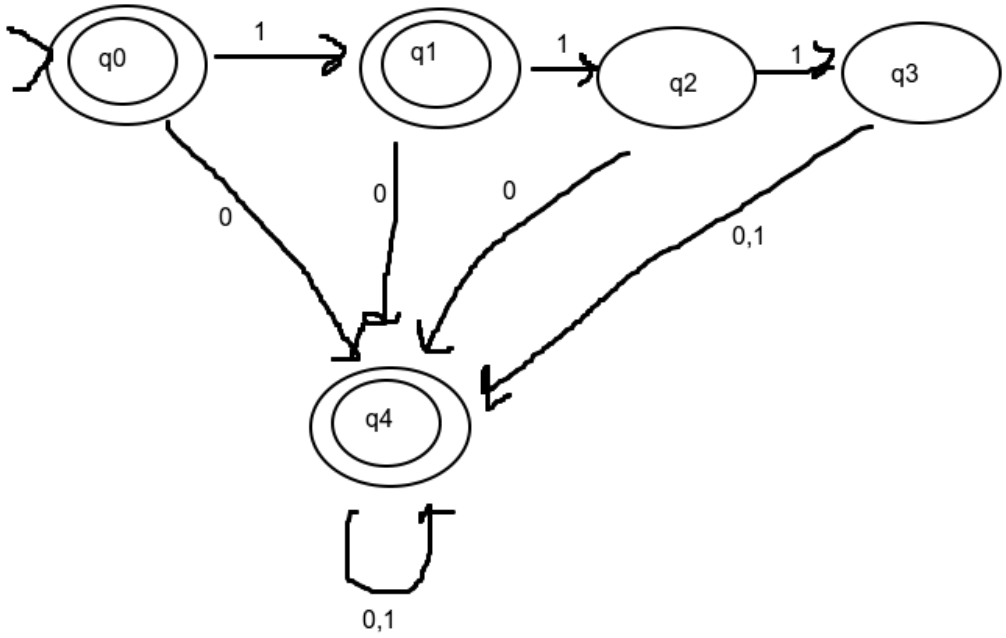
a. 1.6(b) {w | w contains at least three 1s}



Grading Comments:

- -2 for only accepting strings with “111” appearing consecutively in the string
- -2 for no final states

b. 1.6(h) {w | w is any string except 11 and 111}

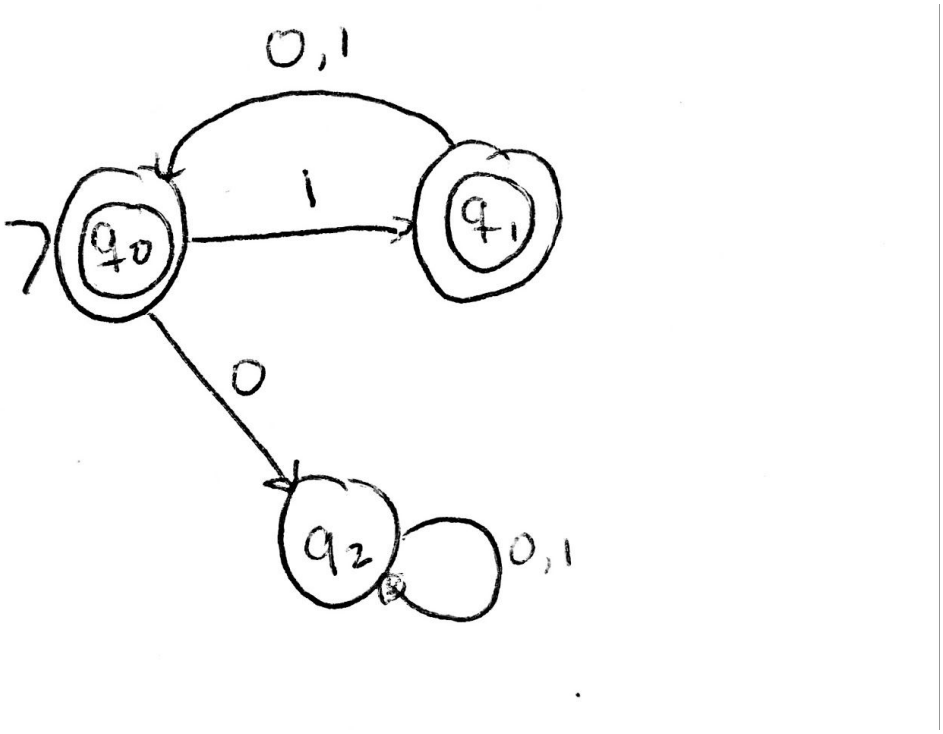


Grading Comments:

- -1 for no 1 on q3->q4 edge
- -1 for not accepting the empty string
- -3 for missing a state that was critical to transition functions

- -1 for not accepting “1”
- -4 for no start/final states, with correct structure

c. 1.6(i) {w | every odd position of w is a 1}



Grading comments:

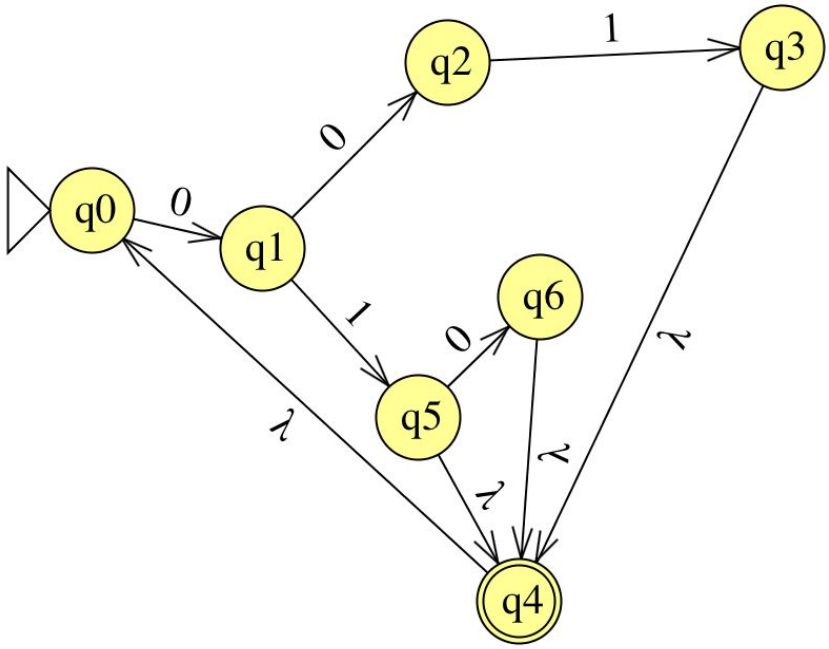
- -6 for significantly flawed design
- Full credit for missing transition functions that would lead to a reject/non-accepting state
- -1 for not accepting the empty string

11. Solve problem 1.17 on page 86 of [Sipser, Second Edition] JFLAP
Question is (01|001|010)*

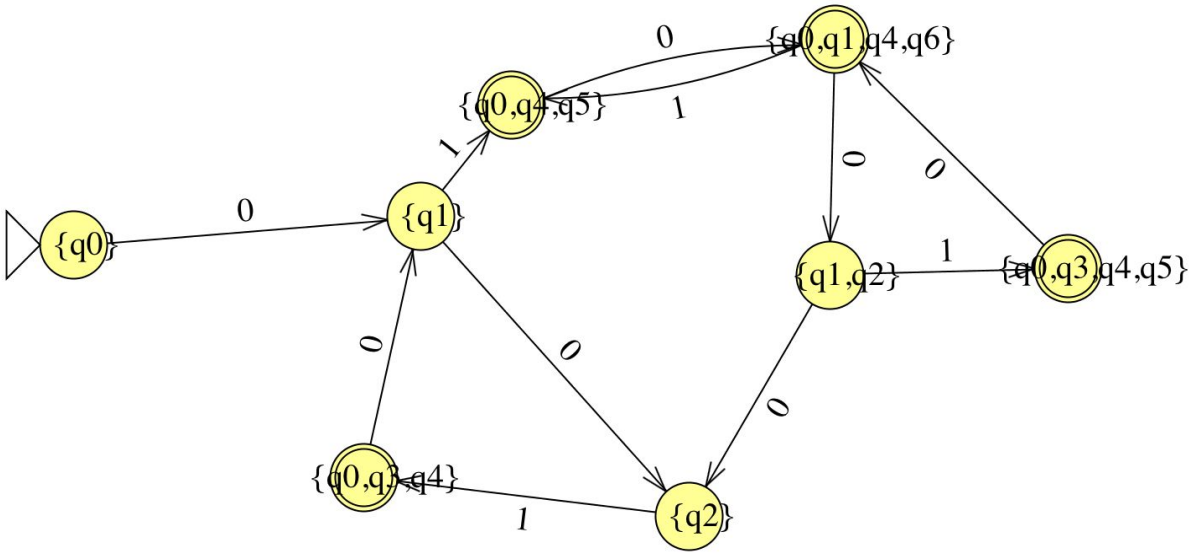
Please check the test cases for the NFA and check if the DFA has been converted correctly. Check if DFA is minimized. DFAs should be a DFAs.. No epsilon or lambda transitions. With an input say 1 it should only go to 1 state. All inputs to the state must be covered (Trap state for all not in the language)

A sample NFA is (More Lambda transitions possible):

Should merge q0 and q4, empty string is valid b/c uses Kleene closure.



The DFA can be the below:



Algorithm to convert NFA to DFA :

procedure: nfa-to-dfa

1. Create a graph G_D with vertex $\{q_0\}$. Identify this vertex as the initial vertex.

2. Repeat the following steps until no more edges are missing.

   Take any vertex $\{q_i,q_j,\ldots,q_k\}$ of G_D that has no outgoing edge for some a $\in \Sigma$ Compute $\delta_N^*(q_i,a), \delta_N^*(q_j,a), \ldots, \delta_N^*(q_k,a)$. If

   $$\delta_N^*(q_i,a) \cup \delta_N^*(q_j,a) \cup \ldots \cup \delta_N^*(q_k,a) = \{q_l, q_m, \ldots, q_n\},$$

   create a vertex for G_D labeled $\{q_l,q_m,\ldots,q_n\}$ if it does not already exist. Add to G_D an edge from $\{q_i,q_j,\ldots,q_k\}$ and label it with a.

3. Every state of G_D whose label contains any $q_f \in F_N$ is identified as a final vertex.

4. If M_N accepts λ, the vertex $\{q_0\}$ in G_D is also made a final vertex.

(Source : Peter Linz, An Introduction to Formal Languages and Automata, 4th Ed)