
- This assignment is **due in Week 7 on Wednesday 14 September, 11:59pm** on Gradescope.
- All work must be **done individually** without consulting anyone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.
- You will be evaluated not just on the correctness of your answers, but on your ability to present your ideas clearly and logically. **You should always explain how you arrived at your answer unless explicitly asked not to do so.** Your goal should be to convince the person reading your work that your answers are correct and your methods are sound.
- For clarifications, input formats, and more details on all aspects of this assignment (e.g., level of justification expected, late penalties, repeated submissions, what to do if you are stuck, etc.) you are expected to regularly monitor the Ed Forum post "Assignment FAQ".

Problem 1. (10 marks, 5 marks each)

Let $\Sigma = \{a, b\}$. For each of the following languages, provide a nondeterministic finite automaton for that language. No justification is necessary. For full marks, your automaton should use at most the specified number of states. Partial marks will be awarded for larger automata.

1. The set of strings with the property that there are two occurrences of the same letter that are exactly 3 positions apart (i.e. there are exactly 2 letters between them). For example, $abaa \in L, ababa \notin L$. (8 states)

2. The set of strings whose length is not divisible by 6 or that contain at least two bs (or both). For example, $ab, ababab \in L, \epsilon, aabaaa \notin L$. (8 states)

Problem 2. (20 marks, 10 marks each)

Let $\Sigma = \{a, b\}$. For each of the following languages, prove that the language is not regular.

1. The set $\{a^{2n}b^n : n \geq 1\}$.
2. The set of strings $\{y_0, y_1, y_2, \cdots\}$ where $y_0 = \epsilon$, $y_i = y_{i-1}a^i$ if i is an odd positive integer, and $y_i = y_{i-1}b^i$ if i is an even positive integer. For example, $y_0 = \epsilon, y_1 = a, y_2 = abb, y_3 = abbaaa, y_4 = abbaaabbbb$, etc.

Problem 3. (20 marks, 5/5/10)

Let $\Sigma = \{a, b\}$. For each of the following languages, provide a deterministic Turing Machine that decides that language. No justification is necessary. For full marks, your machines should work for any input and be reasonably efficient. Guidelines on this efficiency will be posted on Ed.

1. The language $\{a^n b^{n+1} : n \geq 0\}$.

2. The language from Problem 2, part 2.
3. The set of strings of the form uxu, where $u, x \in \Sigma^*$ are strings and $u \neq \epsilon$. For example, $bb, abbba, abbab, baaaaababaa, abbababbab \in L$ while $\epsilon, b, ba, aaabab, bbabaabbaa \notin L$.

Problem 4. (30 marks) An *implicit string* is a way of writing a string where we allow the use of exponents. For example, $(abc)^3 b$ is an implicit string that equals $abcabcabcb$. Implicit strings allow us to write very long strings much more concisely, such as

$$(abc)^{1000000000000000000000000000000000000}db$$

which is too large to store in a modern computer's memory but can be written in under 100 characters as an implicit string.

An implicit string is given in the form

$$(x_1)^{n_1}(x_2)^{n_2}\dots(x_k)^{n_k}$$

where $x_1, x_2, \dots, x_k$ are (ordinary) strings and $n_1, n_2, \dots, n_k$ are positive integers.

Your task is to write code that takes an automaton N and a sequence of implicit strings from stdin, and outputs whether N accepts each of the strings to stdout. Some skeleton code will be provided (in Python) that demonstrates how to handle the input/output. You may assume that the input is valid. Your program will not be tested on invalid input.

You are guaranteed that all input automata will have fewer than 100 states, all input implicit strings will have $k < 10$, and that all resulting strings will be of length less than 2^{62}.

Subproblem	Restrictions	Marks	Example implicit string
1	small string	12	$(ab)^{30}(cda)^{200}(bb)^{17}$
2	1 symbol, powers of 2	6	$(a)^{1125899906842624}$
3	1 string, powers of 2	4	$(abca)^{9007199254740992}$
4	powers of 2	4	$(abc)^{1125899906842624}(db)^{9007199254740992}$
5	no restrictions	4	$(abc)^{1234567891234567}(db)^{73}$

In each subproblem, 50% of the marks will be for DFA cases and 50% will be for NFA cases.

Here is an explanation of the restrictions:

- "small string" means that the resulting string will be of length at most 1000
- "1 symbol" means that the implicit string will be the exponent of a single symbol

- "1 string" means that the implicit string will be the exponent of a single string
- "powers of 2" means that all exponents will be powers of 2. The given example values are $2^{50} = 1125899906842624$ and $2^{53} = 9007199254740992$. You may find the Python function math.log$(x, 2)$ useful in determining which power of 2 a number is.

Hints:

- For the small string test cases, it will be sufficient to expand out the implicit string and run the automata on it.
- For the NFA cases, it is not recommended to convert the NFA to a DFA, because this can cause an exponential blowup in automata size. You should find a way to decide membership for NFAs while avoiding this exponential blowup.
- For the large string test cases, expanding out the implicit string will not be sufficient, because this causes an exponential blowup in string length. You will need to find a more efficient way to run the automaton on the implicit string. Think about what happens when the automaton reads a single character, what happens when it reads multiple characters in sequence, and what happens when it reads the same sequence twice. Furthermore, think about working with the entire transition function, rather than just the transition that actually gets taken.
- Although the general problem (i.e., subproblem 5) is quite difficult, the first few subproblems are easier, so you should attempt them to earn partial marks even if you cannot solve the general problem. The subproblems are in rough difficulty order, so it is recommended that you gradually extend your solution to handle subsequent subproblems and/or to handle NFAs for a subproblem.
- The restriction to powers of 2 provides a hint on how to handle the general problem. Remember that if $n = 2^i$, then for a string x we have that $x^n = (x^2)^{2^{\cdots 2}}$ with i squarings.