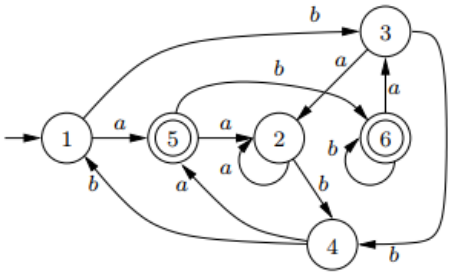
Homework Problems

H3.1 Construct the minimal automaton corresponding to the following deterministic finite automaton:



Exercise H3.1

- Step 1: Remove unreachable state. Since all states in this automaton are reachable, there's no state that can be dropped
- Step 2: The states are partitioned into non-accept states I and accept states II

		a	b
I: →	1	5	3
	2	2	4
	3	2	4
	4	5	1
II: ←	5	2	6
←	6	3	6

- Step 3: marking which class the transition function maps each pair

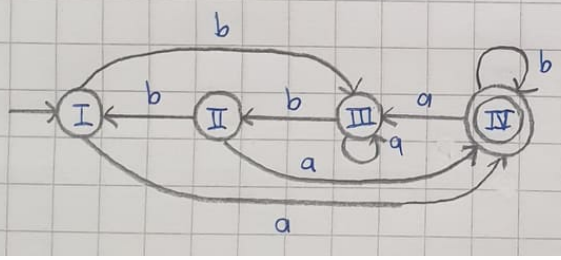
		a	b
I: →	1	5, II	3, I
	2	2, I	4, I
	3	2, I	4, I
	4	5, II	1, I
II: ←	5	2, I	6, II
←	6	3, I	6, II

⇒ Class I has 3 kinds of states {1}, {2, 3} and {4}

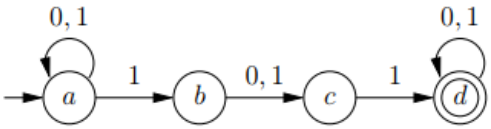
⇒

		a	b
I: →	1	5, IV	3, III
II:	4	5, IV	1, I
III:	2	2, III	4, II
	3	2, III	4, II
IV: ←	5	2, III	6, IV
←	6	3, III	6, IV

- Step 4: The resulting minimal finite automaton is



H3.2 Consider the following nondeterministic finite automaton M recognising a language $\mathcal{L}(M)$ over the alphabet $\{0,1\}$:



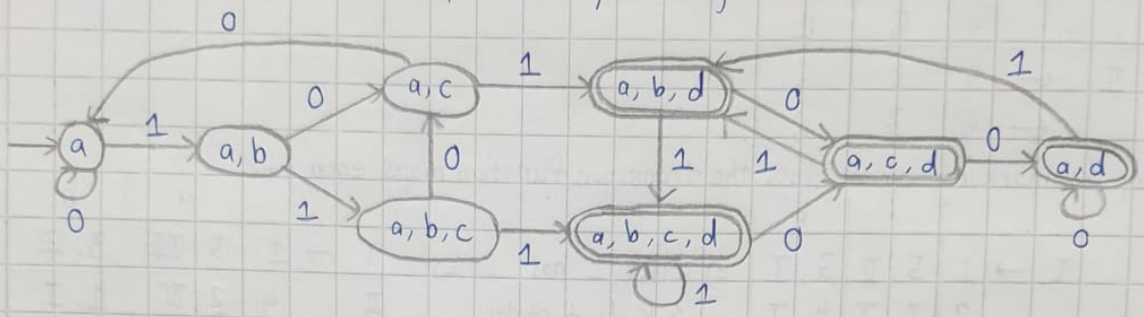
Construct the *minimal deterministic* finite automaton that recognises the *complement* of the language $\mathcal{L}(M)$.

Exercise 3.2

The transition function is:

		0	1
→	a	{a}	{a, b}
	b	{c}	{c}
	c	∅	{d}
←	d	{d}	{d}

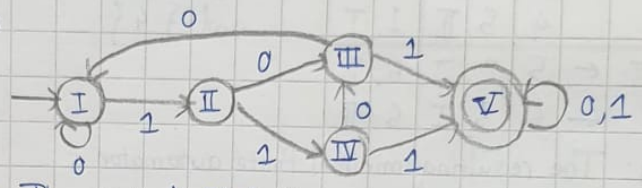
- The deterministic automaton produced by the algorithm is:



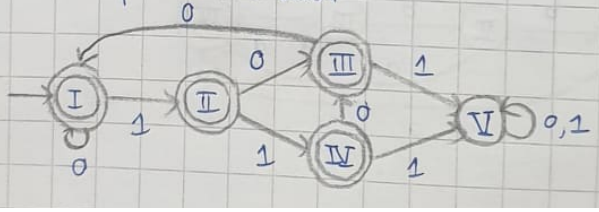
- Marking the classes of the transition function like ex H3.1

The minimal FA would be:

	0	1
→ q_1 : I	q_1 : I	q_2 : II
q_2 : II	q_3 : III	q_4 : IV
q_3 : III	q_1 : I	q_5 : V
q_4 : IV	q_3 : III	q_6 : V
← q_5 : V	q_7 : V	q_6 : V
← q_6 : V	q_7 : V	q_6 : V
← q_7 : V	q_8 : V	q_5 : V
← q_8 : V	q_8 : V	q_5 : V



⇒ The minimal DFA that recognizes the complement of the language is by making above automaton accept as non-accept and vice versa



H3.3 Show that if a language $L \subseteq \{a, b\}^*$ is recognised by some finite automaton, then there are also finite automata that recognise the following languages, consisting of all the prefixes and suffixes of the words in L:

(i) $\text{Pref}(L) = \{x \in \{a, b\}^* \mid xy \in L \text{ for some } y \in \{a, b\}^*\}$,

(ii) $\text{Suff}(L) = \{y \in \{a, b\}^* \mid xy \in L \text{ for some } x \in \{a, b\}^*\}$.

Exercise 3.3

Let M be DFA for the language $L \subseteq \{a, b\}^*$

M_p that recognizes all prefixes of L or Pref(L)

M_S that recognizes all suffixes of L or Suff(L)

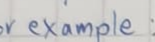
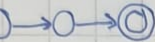
- How to construct M_p

Let Q be set of all states of M that leads to any of accepting states of M.

Q includes both non-accept and accepting states and exclude non-reachable states

For any language L, M can recognize it by going from start to end of the words, or it scans through each character.

=) M_p can be constructed from M by turning all states in Q into accepting states

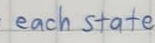
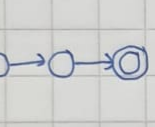
For example : M:  =) M_p : 

because M_p can recognize all prefixes of M and FA works by scanning from start to end

- How to construct M_S

Since M_S recognizes the suffixes or the ending partitions of L, it means initial state in Q of M can reach all states in Q without scanning

=) M_S can be constructed from M by adding ε-transitions from initial state of M to each state in Q

For example : M:  =) M_S : 

Theorem 3.3 : $A = L(M_S)$ for ε-automaton M =) NFA $\hat{M}_S$ exists so $L(\hat{M}_S) = A$

Theorem 3.2 : $A = L(\hat{M}_S)$ for NFA M =) DFA $\hat{\hat{M}}_S$ exists so $L(\hat{\hat{M}}_S) = A$

- How to construct automaton M_{PS} that recognizes both

Since DFA M_p and $\hat{\hat{M}}_S$ can be constructed, the product algorithm can create M_{PS} that recognizes both M_p and $\hat{\hat{M}}_S$, or $M_{PS} = M_p \times \hat{\hat{M}}_S$

=) There exists FA M_{PS} that satisfies the requirements (proven)