minimization of a DFA:

the principle of this algorithm is to identify groups of states with similar behaviors and to group / collapse them.

at first, the only known "behavior" of states is whether they are final or not.

↳ initialization of the algorithms: states are partitioned into two sets of states:

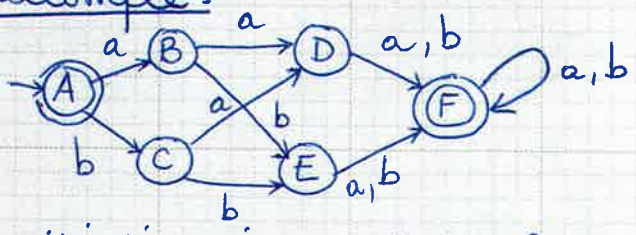
- Final (or Accepting)
- Non-Final (or Non-Accepting).

idea of the algorithm:

- each set of states is checked: each state (inside the set) is checked for its behavior. e.g., when reading an 'a', does this trigger going to a Final / Non-Final / Other set of states?
- constituant states are regrouped together (further breaking the initial Final / Non Final sets) by common behavior.

until all groups are stable.

example:



initialization: Final = {A, F}
Non-Final = {B, C, D, E}.

checking each set of states:

- Final

	a	b
A	NonF	NonF
F	Final	Final

} A and F do not have the same behavior.

↓ we split Final into {A} on one hand and {F} on the other hand.

from now on, we will consider three sets of states:
{A}
{F}
Non-Final

Note: {A} and {F} do not need to be further checked because they are singletons and cannot be further splitted.

- Non-Final

	a	b	
B	Non Final	Non Final	} same behavior
C	Non-Final	Non Final	
D	Final	Final	} same behavior
E	Final	Final	

} different behavior → split

from checking Non-Final, we conclude that this set needs to be split into two new sets:

$NF_1 = \{B, C\}$

$NF_2 = \{D, E\}$

from now on, we are considering four sets of states:

$\{A\}$, $\{F\}$ } no need to check these

$NF_1 = \{B, C\}$

$NF_2 = \{D, E\}$

NF_1	a	b
B	NF_2	NF_2
C	NF_2	NF_2

} same behavior ↓ no need to split.

NF_2	a	b
D	{F}	{F}
E	{F}	{F}

} same behavior ↓ no need to split.

all sets of states have been checked, yielding no further splitting.

= end of algorithm.

the resulting minimized DFA is as follows: