DFA Minimization Algorithm:

Note: *Throughout this algorithm, we will use a container C (you pick your data structure; e.g., stack, queue, priority queue, set) of sets of states.*

Initialization:
Separate Final from Non-Final states.
Put each of these two subsets in C provided that they have at least one state.

Example:
If Final = {A} and Non-Final = {B,C,D}, then you only store Non-Final in your container C.

Loop while our container C is non empty (i.e., as long as we have sets of states to check for consistent behavior).

While (C is not empty) {

Take a set out of C: let's call this set S
Check the behavior of the states inside of S *(i.e., build their transition table)*

If the behaviors of the states inside S are not consistent, **Then {**

(1) Split S in however many different sets are necessary to ensure that the new sets have consistent behavior.

(2) Update the set of new states *(e.g., NF is now replaced by NF1 and NF2 and we will use these and no longer NF to check consistent behaviors)*

(3) Store in C any new sets of states (generated from splitting S) that contains at least 2 states.

(4) Restore in C (if not yet there) **all sets of states that contain 2 or more states**.

}

}
