Converting a given NFA to DFA:

Colors:
Red: Files
Purple: Python objects (variables, dicts, etc)
Orange: State elements
Green: Functions
Blue: Arguments to functions

Algorithm:

1. Read input from input.json.
2. Define a dict out to store the definition of DFA.
3. Define two dictionaries:
    a. dfa: To hold the states of DFA
    b. nfa: To hold the states of NFA
4. Add the given states (0 - n-1) to nfa and every subset of set of NFA states to dfa
5. Define the transitions of the states as:
    a. The transitions from subsets of single elements (in DFA) will be given by the input transition function.
    b. The transitions from subsets of multiple elements will be given by taking union of the transitions of subsets of single elements.
6. Define the transition function in the out dictionary.
7. Write the out dict to output.json.

Class defined:

The class State contains information and methods for a state.

It contains:

(i) name: The name given to the state (In case of DFA, this will consist of the subset that the state is formed of).

(ii) bin: The binary index of the state (For printing).

(iii) In: States which can reach the concerned state consuming a single character from the alphabet.

(iv) Out: States which can be reached from the concerned state consuming single character from the alphabet.

(v) transitions: Defines the next state reached on consuming different characters from the alphabet on this state.

(vi) addTransitions(self, letter, output, automata): function that adds the transition of type:

self --letter--> output

It also adds the current state to In of output and output to Out of self.

If (automata == 'dfa') The function works in the dfa dict, else it works in nfa dict. (By default, automata = 'dfa')

Functions:

1. fz(set):

   Takes an int or a set as input and returns a frozenset. (This is needed because a set is unhashable but frozenset is hashable)

2. union(states):

   Takes input a list of states (or a set) and returns a frozenset of the union of set of each state.

   If the list (or set) is empty, it returns frozenset({'phi'}).

   If the union of sets is not present in the current list of states in dfa, it adds the state to dfa.

3. trans_union(letter, states):

   Takes an input character from alphabet and a list of states and returns a combination of transitons of every state in states on consuming letter.

   It is used to define the transitions of subsets of more than one element in the DFA.

4. get_bin(x):

   Takes an int x and returns binary representation of x.

5. findsubsets(s, n):

    Takes in a set s and an int n and returns a list of all subsets of s with n elements.

6. isfinal(state):

    Adds state to the set of final states of DFA if any element of state is in set of final states of NFA.

7. b_s(states):

    Takes input a set states and returns a list of individual states in the set.

8. State_construction():
    a. First it adds the state "phi" to dfa.
    b. A set is defined which consists of numbers 0 - n-1. These represent the states of the input NFA where n is the number of states of the NFA.
    c. All subsets of the set of size 1 - n-1 are added to dfa and states are added to nfa as well.
    d. Each added state is sent to isfinal() function.
9. Transition_construction():
    i. Defines the transition from 'phi' on every letter.
    ii. Defines the transitions of rest of the states of the DFA starting from states with only one element (Directly from the input transition function) and uses them to define the transition of rest of the states.

iii. Finally checks if transition on any letter is still undefined and defines such transitions as:

state --letter--> "phi"

10. Generate_output(reduce):

Updates the out dict with all the defined transitions.

If (reduce == true), the DFA is simplified before updating

Additional functions:

1. State_reduction():

Simplifies the DFA.

i. Adds all states with no element in In to a queue.
ii. Starts working on the queue and removes any state in the queue from dfa if their In is empty.
iii. On removing a state, all states in the Out of that state are added to the queue.
iv. When the queue empties, all useless states have been removed.

2. Print_Table():

Prints the table of transitions of various states on consuming different characters from the alphabet.

3. nfa_run(In, curr_state):

Runs the NFA on given input In.

4. dfa_run(In):

   Runs the DFA on given input In

5. testNFA_DFA():

   Asks for a number and runs the same number of inputs on both, NFA and the DFA giving the result of both.

   It asks for the input before every run.