Report for NFA to DFA

Logic:

For *n* states in NFA there will be 2^n ,as for each state in NFA there are 2 possibilities ,either it is there or not.The case in which no state is there is considered as dead state(0).

Consider following NFA transition function:

states\/ | alfa->	a	b
0	[0,1]	[2]
1	[0]	[1]
2	[]	[0,1]

- States in NFA=0,1,2
- Stated in DFA=[];[0];[1];[2];[0,1];[0,2];[1,2],[0,1,2]
- Numbering = 0,1, 2, 4, 3, 5, 6, 7,(Binary wise)
- eg.[1,2]=[110]=6 (one on 1 and 2 position)
- eg.[0,1]=[011]=3(one on 0 and 1 position)
- **ASSUMPTION:** Instead of representing states as sets in DFA (like [0,1,3]), I have represented state as a single number in whose binary representation there is one on the position of element in states. eg. state [0,1,3]=11 as 11 in binary is “1011” i.e one on 3rd,1st,0th position.

Code:

```
3 with open('input.json') as f:
4     d=json.load(f)
```

➢ Open input.json file as python object *“d”*

```
14 inp = [[[]] * len(l) for i in range(n)]
15
16 for e in tf:
17     inp[int(e[0])][l.index(e[1])]=e[2]
```

➢ Makes *inp* array to make transition table of t_func and fills it accordingly.

```
28 for i in range(0,2**n):
29     t=i
30     kiska=[]
31     j=0
32     while t>0:
33         if t%2==1:
34             kiska.append(j)
35         t = int(t/2)
36         j = j+1
```

➢ The for loop runs for all 2^n states and all alphabets to check what will be output for given state and input.

➢ The inside while loop (t>0) runs to check for *i*th state in DFA which all states will be considered from NFA.

➢ Eg. for i = 6 ,kiska=[1,2] ,therefore dfa(6,a) = nfa(1,a) U nfa(2,a)

```
37     for finalstate in f:
38         if(finalstate in kiska):
39             fina.append(i)
```

➢ For a final state in NFA say *"x"* ,all those states in DFA which have contribution of *x* will be final state.Above code is performing this operation.

```
40     for alfa in l:
41         union=[]
42         for state in kiska:
43             union= list(set(union)|set(inp[state][l.index(alfa)]))
44         fin=0
45         for num in union:
46             fin=fin+(2**num)
47         out[i][l.index(alfa)]=fin
48         dfa.append([i,alfa,fin])
```

➢ Above is taking union of multiple output states for a given state in DFA.

➢ Eg. like in the example given at the beginning we want to find out what will be the output when input *"b"* is given on state "3" ?

- State 3 = [011] i.e. in this case we have to consider output of both 0th and 1st state in NFA(as there is one on 0 and 1 position).
- nfa(0,b)=[2] , nfa(1,b)=[1], hence dfa(3,b)=[2]U[1]=[2,1]=6 (6=[110] one on 2nd and 1st position and zero on 0th)

```
65 final = []
66 for ele in fina:
67     if ele not in final:
68         final.append(ele)
```

➢ Remove duplicates from final

```
70 y={
71         "states": 2**n ,
72         "letters":l,
73         "t_func":dfa,
74         "start":2**int(d['start']),
75         "final":final
76         }
77
78 with open("output.json", "w") as write_file:
79         json.dump(y, write_file)
```

➢ Makes json object from python object and makes output.json file