P := {F, Q \ F};

W := {F};

while (W is not empty) do

choose and remove a set A from W

for each c in ∑ do

let X be the set of states for which a transition on c leads to a state in A

for each set Y in P for which X ∩ Y is nonempty do

replace Y in P by the two sets X ∩ Y and Y \ X

if Y is in W

replace Y in W by the same two sets

else

if |X ∩ Y| <= |Y \ X|

add X ∩ Y to W

else

add Y \ X to W

end;

end;

end;

I made a converting from DFA to minimal DFA by using Hopcroft algorithm. I find a pseudo code from Wikipedia by this link, <http://en.wikipedia.org/wiki/DFA_minimization#Hopcroft.27s_algorithm> and I made python code to converting dfa.

def minimize(self,dfa):

P = [set(dfa.finalstates),dfa.states.difference(set(dfa.finalstates))]

W = [set(dfa.finalstates)]

while len(W) != 0:

A = W[0]

W.remove(A)

X = set()

for c in dfa.alphabet:

for from\_state, to\_state in dfa.transitions.items():

for toNum, value in to\_state.items():

if c in value and toNum in A:

X.update(set([from\_state]))

for Y in P:

if not X.intersection(Y) == set():

P.append(X.intersection(Y))

P.append(Y.difference(X))

if Y in W:

W.append(X.intersection(Y))

W.append(Y.difference(X))

W.remove(Y)

else :

if len(X.intersection(Y)) <= len (Y.difference(X)):

W.append(X.intersection(Y))

#W.remove(Y)

else :

W.append(Y.difference(X))

#W.remove(Y)

P.remove(Y)

print P,W

my DFA consists of five indicators,

dfa.states = set() of states.

dfa.startstate = int number of start state number

dfa.finalstates = list structure consisting of final states

dfa.transitions = dictionary structure of dfa transitions.

ex) {from state : {to state1 : set of character to go to state1 , to state 2 : set of charac...}}

dfa.alphabet = set of dfa alphabets.

I don't know why this algorithm not work my code, is any problem in my code?