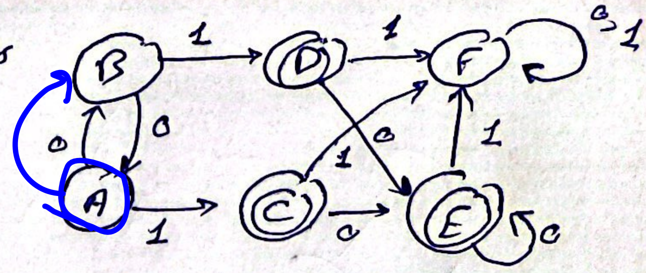
steps:

1: Build n×n matrix, where n = # states



	A	B	C	D	E	F
A	T					
B		T				
C			T			
D				T		
E					T	
F						T

T → Represent

F → for sure they're not the same

dest = dump node (B, f, 1)

2. iterate for i = 0 to n
   - 2.1. iterate for j = 0 to i
     1. if ~~m[i][j]~~ state[i].isTerminal != state[j].isTerminal

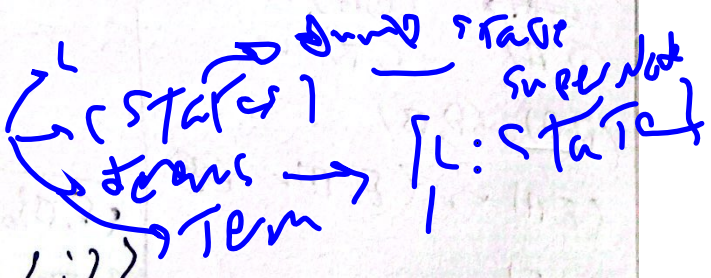
لو واحد Terminal والتاني لأ يبقى نحطهم انهم اكيد مش زي بعض

       1. m[i][j] = 'F' // initialization step
          m[j][i] = F

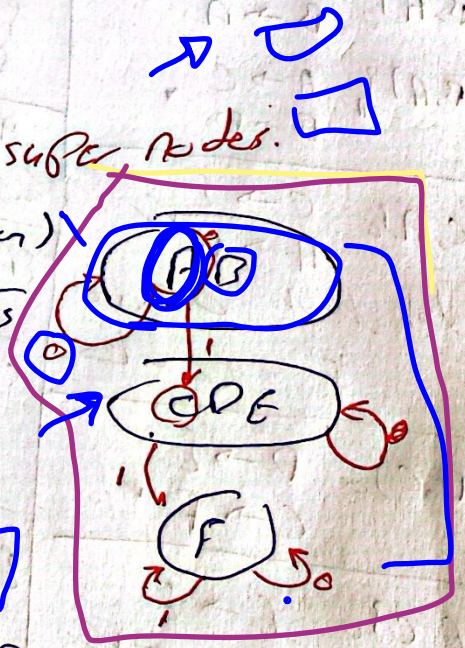
3. bool nothing changed is initialized to False
4. repeat until nothing changed = True.
   - 4.1. for i = 0 to n
     - 4.1.1. for j = 0 to i — if(m[i][j] != 'f' || m[j][i] != 'f')
       - 4.1.1.1. For all possible transition. (trans in Transitions)
         1. s1T = states[i].[trans]
         2. s2T = states[j][trans]
         3. if m[s1T][s2T] == 'F'
            1. m[i][j] = m[j][i] = 'f'
            2. break no need to continue
            2. nothing changed = False

5. list of list = [] to hold super groups
6. list of bool to mark inserted nodes

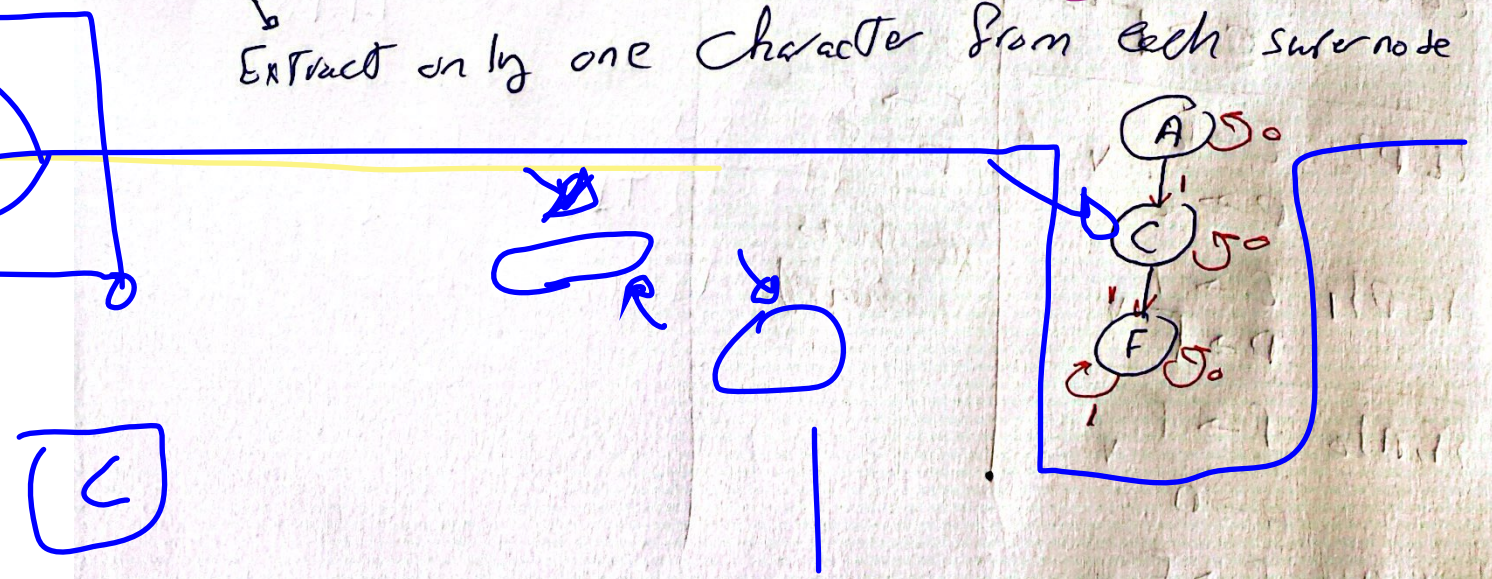
7. for i = n-1 to 0
   1. if ! inserted[i]
      ~~if m[i][j] != ~~
      1. list = [state[i]]
      2. for j=0 to i
         1. if ! inserted[j]
            1. if m[i][j] != 'f'
               1. list.append [states[j]]



8. //Now we need to connect states together.
9. for transition in transitions.
   1. for i=0 to list.size() # of super nodes.
      1. Dest = list[i][0][transition]
      2. search for Dest in all list components
      3. Connect them together
      4.



10. final Min DFA
    ↳ Extract only one character from each supernode



Tracing my Algo:

~~{A, B}~~ ~~A→B~~
~~B→A~~

1. Initialization

~~(A, B)~~, (C, A) ✓
(B, A)   (C, B) ✓
(D, A), (D, B)
(E, A) ✓, (E, B) ✓
(F, E)

	A	B	C	D	E	F
A			'F'	'F'	'F'	'F'
B			'F'	'F'	'F'	'F'
C	'F'	'F'		'F'	'F'	'F'
D	'F'	'F'	'F'	~~F~~	~~'F'~~	'F'
E	'F'	'F'	'F'			'F'
F	'F'	'F'	'F'	'F'	'F'	

{B, A}₀ A→B
B→A ✗

{B, A}₁ A→C
B→D ✗

{C, A}₀ C→E ✓✓
A→B
did check does 'F' - B, cols

{D, C}₀ D→E
C→F ✓

{E, C}₀ E→E
C→F ✓✓

{E, D}₀ E→E
D→E ✗

{E, D}₁ E→F
D→F ✗

{F, A}₀ F→F ✗
A→B

{F, A}₁ F→F ✓✓
A→C

{F, B}₀
F→F ✗
B→A

F→F ✓✓
B→F

∴ there are changes
Repeat

{A, B}₀ A→B
B→A

{A, B}₁ →
A→C ✗
B→D

{E, D}₀ →
E→E ✗
D→E
E→F ✗
D→F

No others
changed.