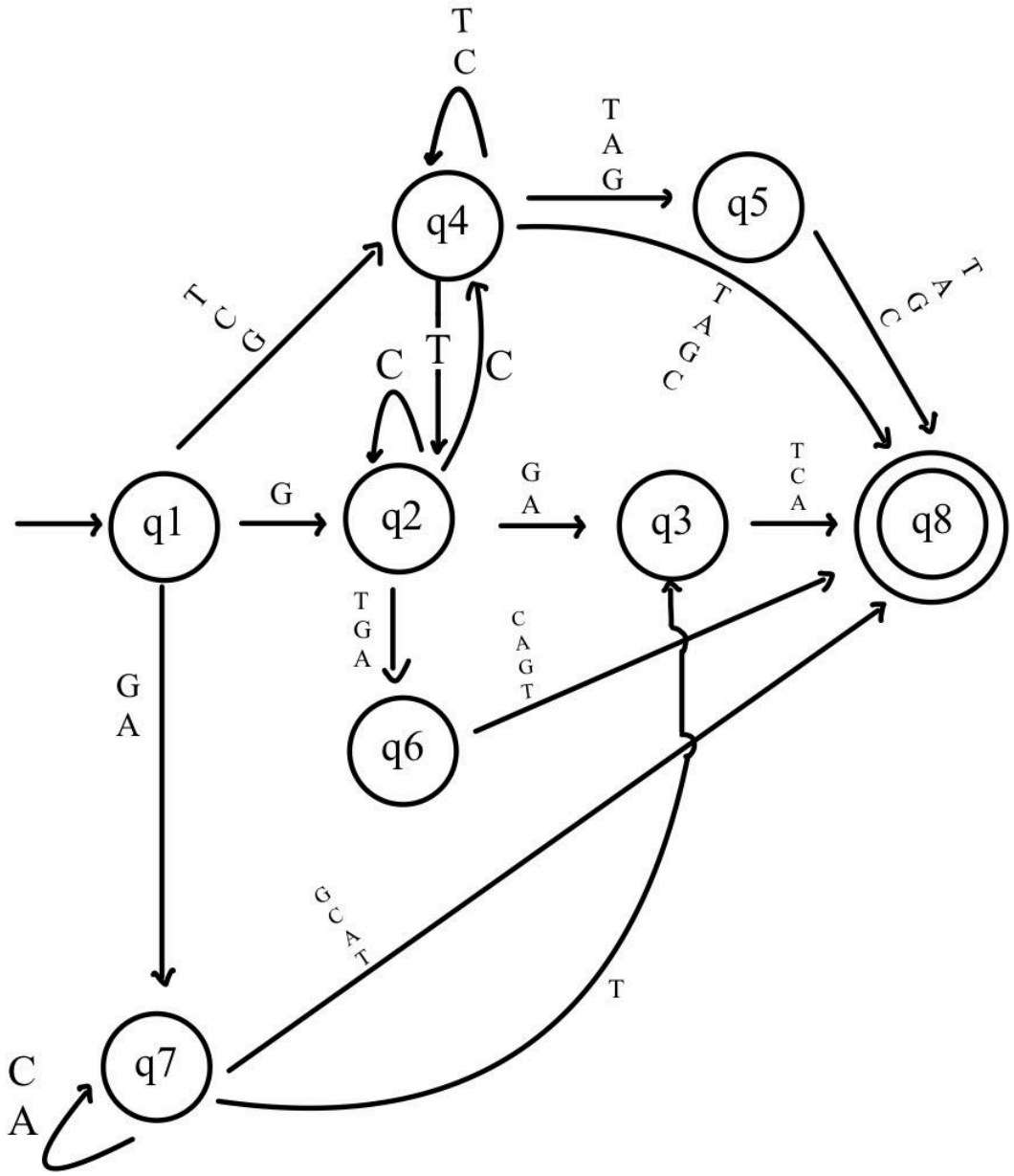
DIAGRAM TRANSISI MODEL FSA



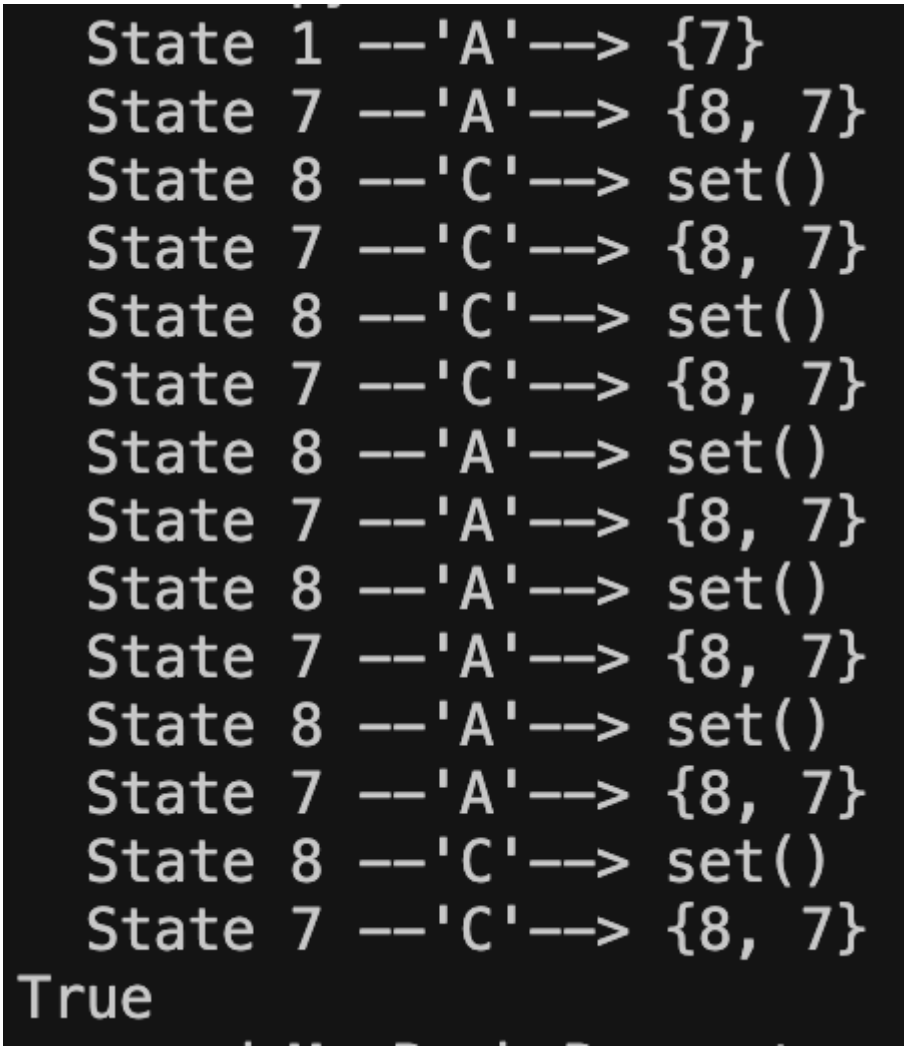
CODING MODEL FSA

```
class NonDeterministicFiniteAutomata:
   def __init__(self, Q, Sigma, delta, S, F):
       self.states = Q
       self.inputs = Sigma
       self.FTransisi = delta
       self.StateAwals = S
       self.StateAkhirs = F

   def do_delta(self, q, x):
       # Function to find the next states for a given state and input
       try:
           return self.FTransisi[(q, x)]
       except KeyError:
           return set()  # Return empty set if no transition exists

   def delta_nfa(self, word):
       # Process a string to see if the NFA accepts it
       P = self.StateAwals  # Start with initial states
       while word:

           Pnew = set()  # Temporary set for next states
           for i in P:
               next_states = self.do_delta(i, word[0])
               print(f"  State {i} --'{word[0]}'--> {next_states}")  # Print each
transition
               Pnew = Pnew | next_states

           word = word[1:]  # Move to the next character
           P = Pnew

       return (P & self.StateAkhirs) != set()  # Check if any final state is in P

   def hat_delta_nfa(self, curr, word):
       # Recursive function to get all reachable states for a given input string
       if word == "":
           return curr
       next_states = set()
       for state in curr:
           next_states = next_states | self.do_delta(state, word[0])
       return self.hat_delta_nfa(next_states, word[1:])

DNA2 = NonDeterministicFiniteAutomata({1,2,3,4,5,6,7,8},
                                      {"A","T","C","G"},

{(1,"G"):{4,2,7},(1,"A"):{7},(1,"T"):{4},(1,"C"):{4},
```

```
(2,"G"):{3,6},(2,"A"):{3,6},(2,"T"):{6},(2,"C"):{2,4},
                                    (3,"A"):{8},(3,"T"):{8},(3,"C"):{8},

(4,"G"):{5,8},(4,"A"):{5,8},(4,"T"):{2,4,5,8},(4,"C"):{4,5},

(5,"G"):{8},(5,"A"):{8},(5,"T"):{8},(5,"C"):{8},

(6,"G"):{8},(6,"A"):{8},(6,"T"):{8},(6,"C"):{8},

(7,"G"):{8},(7,"A"):{7,8},(7,"T"):{3,8},(7,"C"):{7,8},
                                    },
                                   {1},
                                   {8})
```

Input : “AACCAAAC”

Output



PENJELASAN

Source code yang terlihat diatas adalah sebuah FSA(Finite State Automata) berjenis NFA(Nondeterministic Finite Automata). Cara kerjanya adalah dengan pembuatan sebuah class bernama NonDeterministicFiniteAutomata yang didalamnya memuat semua logika dalam proses NFA. classnya juga memiliki constructor yang berisi semua parameter atau quintuple yang akan digunakan dalam nfa seperti Himpunan states(Q),himpunan input(Sigma), fungsi transisi(delta), himpunan state awal(S) dan himpunan state akhir(F).

Selanjutnya terdapat sebuah function bernama do_delta yang berfungsi untuk menemukan state selanjutnya berdasarkan state yang diberikan oleh parameter function dengan mengecek dictionary FTransisi. Function akan terus mengecek apakah next state tersedia dan jika tidak ada maka akan meengeluarkan pesan Error, untuk menanggulangi hal tersebut diimplementasikanlah syntax try dan except agar saat error program akan mengembalikan sebuah set yang kosong.

Setelah itu fungsi delta_nfa digunakan untuk memproses suatu string dan menentukan jikalau string tersebut memenuhi bahasa yang telah di spesifikasi. Variabel P menyimpan semua set awal dan function akan terus berjalan selagi inputan atau katanya masih tersedia. variabel Pnew menyimpan set sementara untuk state selanjutnya. Function mengimplementasikan nested loop untuk mengiterasi semua kemungkinan state dari himpunan state awal. Variabel Pnew juga diupdate secara berkala dan Pnew disimpan di variabel P dan input katanya dipotong satu persatu sampai habis semua. Terakhir function mengecek apakah P terdapat di dalam set himpunan state akhir.

selanjutnya terdapat function hat_delta_nfa yang secara rekursif akan menentukan input diterima atau tidak. Base case dari function rekursif tersebut merupakan string kosong yaitu dimana semua string sudah diproses oleh function dan secara rekursi mengiterasi input katanya dengan memanggil function dirinya sendiri.