https://github.com/SirADV24/VranciuAdrian-FLCDlab/tree/main/lab4

```
self.Q = [] # States
self.E = [] # Alphabet
self.q0 = None # initial state
self.F = None # Final state
self.S = {} # Transitions
```

Above you can see the structure of my finite automata class.

- In the Q field we keep the states of the FA
- In the E field we keep the alphabet
- In the q0 we keep the initial states
- In the F field we keep the final states
- In the S field we keep the transitions

fa.txt

```
Q:A B C
E:1 2
q0:A
F:A C
S: A 1 B, B 1 B, B 2 C, A 2 C
```

Above you can see how I represented the FA in the .in file.

To read it I used the following approach

- For the first 4 rows we separate by ":" first to eliminate the title of the row, then split the second part of the result by space

- For the last row (transitions) we split again by ":" to eliminate the title of the row, then split by ",". For every result we split by space. Notice that all results will have length 3 (origin, cost, destination). For every result we will build the dictionary as following: {(origin, cost): destination)}

To verify if a sequence is accepted, we iterate through the elements of the sequence and starting from the initial state we check if there is a pair in the dictionary keys with the form (current, sequence[I]).

This is similar to a depth first approach in graph search.