README.md

Name	Arnab Sen
Roll	510519006
Date	24-Feb-2021

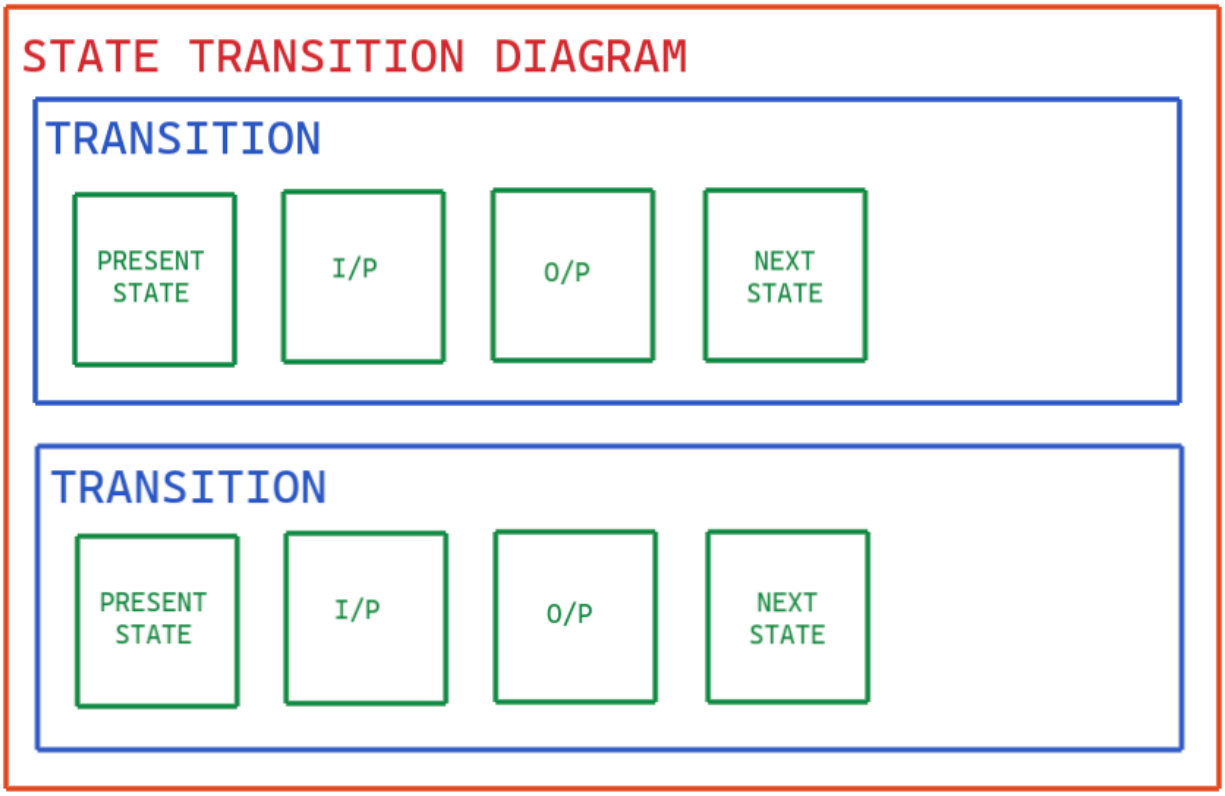
Data Structure favourable for the Algorithms.

XML data consists of:

1. Data of the state diagram
    i. States
    ii. Inputs/Outputs
2. Data of the drawn figure
    i. Coordinates of the images
    ii. Colors
    iii. Stroke Width

For the algorithm to generate the final sequential circuit we need to deal with pt 1 only.

The current XML state is structured this way:



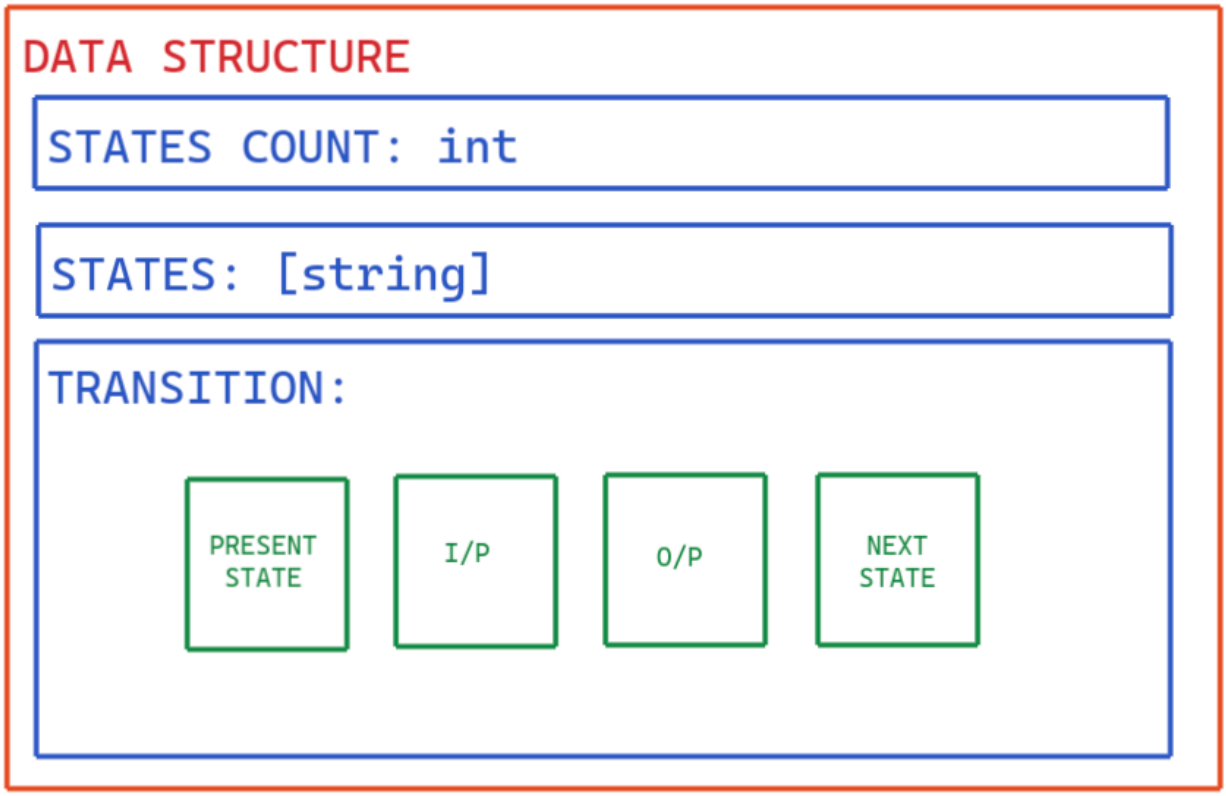
Every transition will have these properties:

1. Input
2. Output
3. Present State
4. Next State

To successfully solve a state transisiton diagram, we need to have:

1. Number of states
2. States
3. Transitions between them

So, we can structure it this way:



Generating the Data Structure

To get this data structure we can pass the RAW JSON to a function.

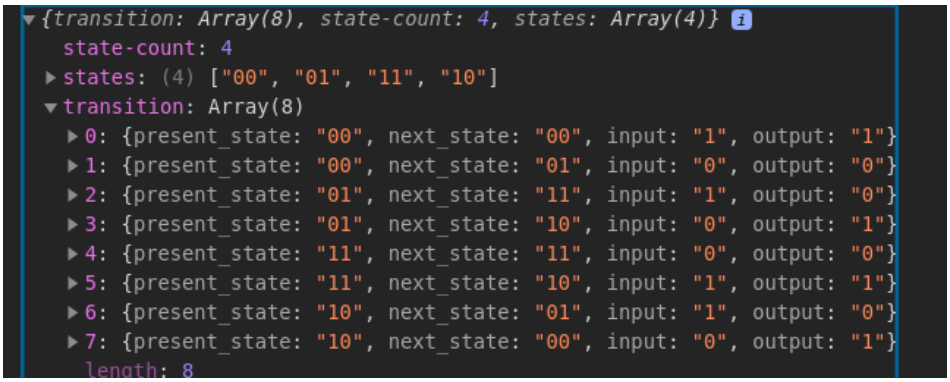
```
function parseDataStructure(data) {
    transitions = data['DATA']['state-diagram']['transition'];
    finalData = {};

    finalData['transition'] = [];

    states = new Set(); // Only unique elements will be included

    transitions.forEach((trans) => {
        finalData['transition'].push({
            present_state: trans['present_state']['#text'],
            next_state: trans['next_state']['#text'],
            input: trans['input']['#text'],
            output: trans['output']['#text'],
        });

        states.add(trans['present_state']['#text']);
        states.add(trans['next_state']['#text']);
    });
    finalData['state-count'] = states.size;
    finalData['states'] = [...states];

    console.log(states);
    return finalData;
}
```

Working of the function:

1. Filtering out the rest of the unnecessary data, only considering those needed for the computation.

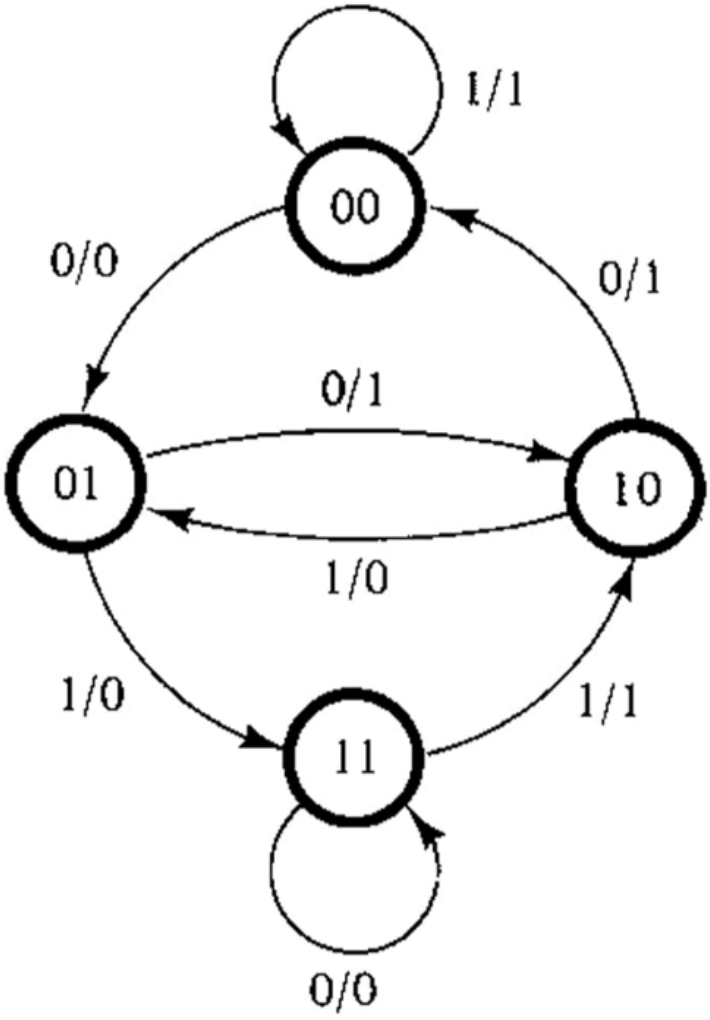
2. Going through all the transisitons and appending the state in a Set. Because at the end we would need an array of unique states. The size of the set will give us the state count.
3. Appending the transitions to an array.

So for the given XML input we will get this output:




```
{
   state-count: 4,
   states: Array(4)
   [
      0: "00"
      1: "01"
      2: "11"
      3: "10"
   ],
   transition: Array(8)
   [
      0: {present_state: "00", next_state: "00", input: "1", output: "1"}
      1: {present_state: "00", next_state: "01", input: "0", output: "0"}
      2: {present_state: "01", next_state: "11", input: "1", output: "0"}
      3: {present_state: "01", next_state: "10", input: "0", output: "1"}
      4: {present_state: "11", next_state: "11", input: "0", output: "0"}
      5: {present_state: "11", next_state: "10", input: "1", output: "1"}
      6: {present_state: "10", next_state: "01", input: "1", output: "0"}
      7: {present_state: "10", next_state: "00", input: "0", output: "1"}
   ]
```


The above output is for the following state diagram:



Verifying the Data Structure with the State Diagram

State Diagram Image	Data Structure generated
The number of states is 4	The value of the `state-count` is also 4
The number of transitions is 8	The length of the `transition` is also 8
Every transition has one input and one output value	All the elements of the `transition` has attributes `input` and `output`

Also, if we go through each transition manually, there exists a corresponding element in `transition` .