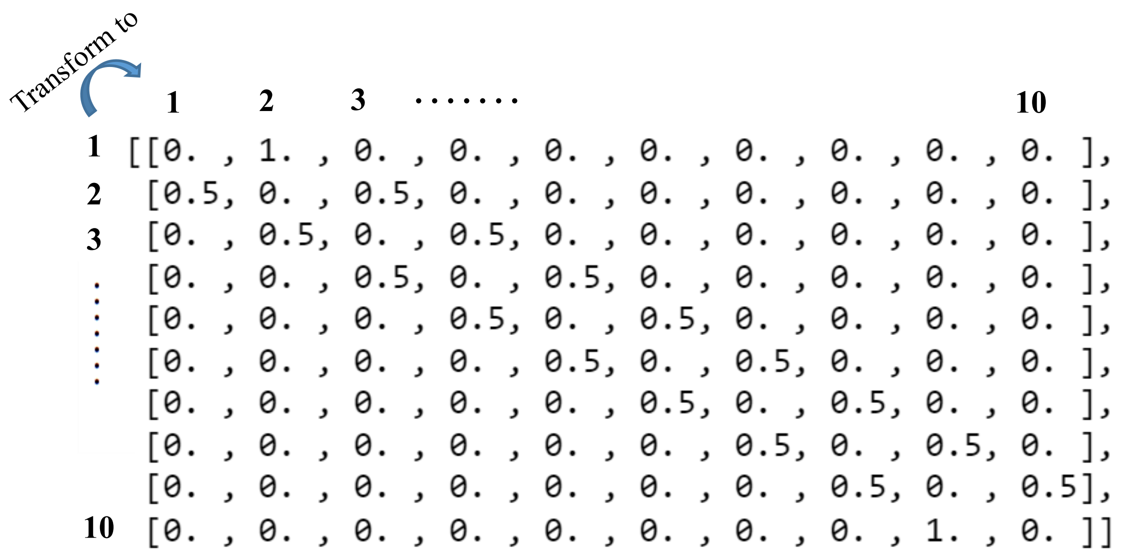
I. Data Structure

The data in this homework is a ten-number sequence— [8,6,4,6,5,4,5,5,7,9]—with each number representing a Hidden Markov Model's observable value at the timestep ti. The hidden values, corresponding to the observable values, are other ten numbers that each of them might be one of {1,2,3...,10}.

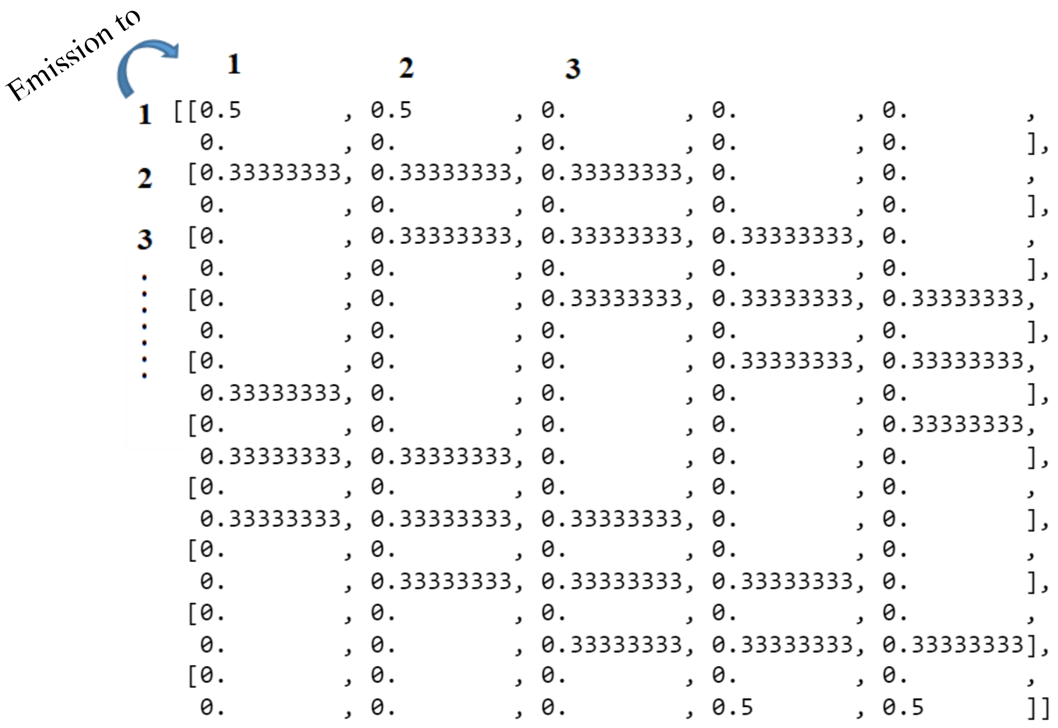
In addition, a hidden state value v has a probability of 0.5 each to transform to v+1 or v-1 at the next timestep. Meanwhile, a hidden state value v has the equal chance to be v, v-1 or v+1 on the observed states. That is, number 10 and 1 has a 1/2 chance each to become themselves or 9 and 2 respectively; and for the remaining numbers, it has the same probability of 1/3 to become v,v-1 or v+1 on the observed states.

Thus, from the above information, we can construct the following transition matrix:


Transform to
1
2
3
....
10
1
2
3
10

```
[[0. , 1. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
 [0.5, 0. , 0.5, 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
 [0. , 0.5, 0. , 0.5, 0. , 0. , 0. , 0. , 0. , 0. ],
 [0. , 0. , 0.5, 0. , 0.5, 0. , 0. , 0. , 0. , 0. ],
 [0. , 0. , 0. , 0.5, 0. , 0.5, 0. , 0. , 0. , 0. ],
 [0. , 0. , 0. , 0. , 0.5, 0. , 0.5, 0. , 0. , 0. ],
 [0. , 0. , 0. , 0. , 0. , 0.5, 0. , 0.5, 0. , 0. ],
 [0. , 0. , 0. , 0. , 0. , 0. , 0.5, 0. , 0.5, 0. ],
 [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.5, 0. , 0.5],
 [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 1. , 0. ]]
```

Every single row in this matrix contains a hidden state value's probabilities of transforming to another value (in columns) at the next timestep. And the emission matrix is another 10 x 10 matrix:


Emission to
1
2
3
1
2
3

```
[[0.5       , 0.5       , 0.        , 0.        , 0.        ,
  0.        , 0.        , 0.        , 0.        , 0.        ],
 [0.33333333, 0.33333333, 0.33333333, 0.        , 0.        ,
  0.        , 0.        , 0.        , 0.        , 0.        ],
 [0.        , 0.33333333, 0.33333333, 0.33333333, 0.        ,
  0.        , 0.        , 0.        , 0.        , 0.        ],
 [0.        , 0.        , 0.33333333, 0.33333333, 0.33333333,
  0.        , 0.        , 0.        , 0.        , 0.        ],
 [0.        , 0.        , 0.        , 0.33333333, 0.33333333,
  0.33333333, 0.        , 0.        , 0.        , 0.        ],
 [0.        , 0.        , 0.        , 0.        , 0.33333333,
  0.33333333, 0.33333333, 0.        , 0.        , 0.        ],
 [0.        , 0.        , 0.        , 0.        , 0.        ,
  0.33333333, 0.33333333, 0.33333333, 0.        , 0.        ],
 [0.        , 0.        , 0.        , 0.        , 0.        ,
  0.        , 0.33333333, 0.33333333, 0.33333333, 0.        ],
 [0.        , 0.        , 0.        , 0.        , 0.        ,
  0.        , 0.        , 0.33333333, 0.33333333, 0.33333333],
 [0.        , 0.        , 0.        , 0.        , 0.        ,
  0.        , 0.        , 0.        , 0.5       , 0.5       ]]
```

Here, every single row contains a hidden state value's emission probabilities of showing another number on the observed states.

II. Coding

1. Construct the transition matrix-***MatrixA***.
2. Construct the emission matrix-***MatrixB***.
3. Select the first potential hidden state values v1 under the observed state [8,6,4,6,5,4,5,5,7,9]. In this case, given 7,8, or 9 all shares the same possibilities to be the v1, we choose 7 as our v1 in the following algorithm example.

```
#Choose the v1,in the example below,7:
observe=[8,6,4,6,5,4,5,5,7,9]
t1=dict()
for i in range(len(matrixB)):
    t1_prob = (0.1*matrixB[i][(observe[0])-1])
    if t1_prob != 0:
        t1[i+1] = t1_prob
print(t1)
```

```
{7: 0.033333333333333333, 8: 0.033333333333333333, 9: 0.033333333333333333}
```

4. Calculate the potential v2 values with Viterbi Algorithm.

```
#Calculate the possible v2 value using Viterbi Algorithm:

#A dictionary to store the possible v2 values:
t2 = dict()

#A dictionary to record the route between timestep t1 and t2, which will be used to plot the relationship diagram:
transition_12 = dict()

for i in range(len(matrixA[6])):
    if matrixA[6][i] != 0:
        accu_prob = t1[7]+(matrixA[6][i]*matrixB[i][5])
        t2[i+1] = accu_prob
        transition_12[i+1]='from7'
        dot_12.node(f'{i+1}',f'{i+1}',**{'width':'0', 'height':'0'})
        dot_12.edge('7',f'{i+1}')
```

Here, three elements will be yielded:

(1) A dictionary ***t2***. The dictionary keys are potential v2; the values of them are probabilities each value accumulates along the route it passes by before:

```
t2
```

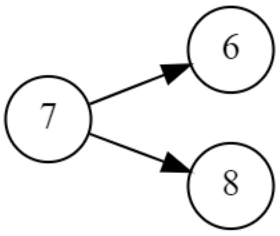
```
{6: 0.19999999999999998, 8: 0.033333333333333333}
```

(2) A list ***transition_12***. This list is used to record the route between timestep t1 and t2 which helps us draw the relationship diagram with graphviz package:

```
transition_12
```

```
{6: 'from7', 8: 'from7'}
```

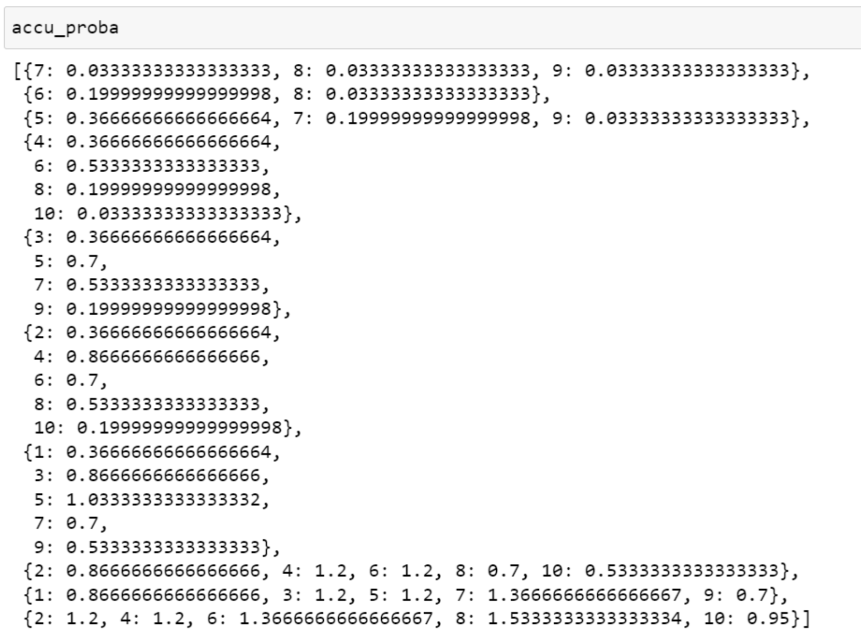
(3) The relationship diagram between t1 and t2:



5. Repeat the above step for t2 to t10 with ***for*** loop.

III. Outcome

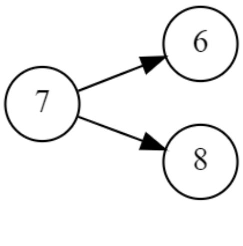
After the completion of computation, we can get the accumulated transition/emission probabilities of possible hidden values on every timestep:



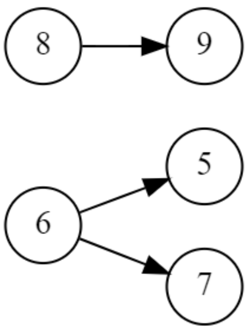
```
accu_proba
```

```
[{7: 0.03333333333333333, 8: 0.03333333333333333, 9: 0.03333333333333333},
 {6: 0.19999999999999998, 8: 0.03333333333333333},
 {5: 0.36666666666666664, 7: 0.19999999999999998, 9: 0.03333333333333333},
 {4: 0.36666666666666664,
  6: 0.5333333333333333,
  8: 0.19999999999999998,
  10: 0.03333333333333333},
 {3: 0.36666666666666664,
  5: 0.7,
  7: 0.5333333333333333,
  9: 0.19999999999999998},
 {2: 0.36666666666666664,
  4: 0.8666666666666666,
  6: 0.7,
  8: 0.5333333333333333,
  10: 0.19999999999999998},
 {1: 0.36666666666666664,
  3: 0.8666666666666666,
  5: 1.0333333333333332,
  7: 0.7,
  9: 0.5333333333333333},
 {2: 0.8666666666666666, 4: 1.2, 6: 1.2, 8: 0.7, 10: 0.5333333333333333},
 {1: 0.8666666666666666, 3: 1.2, 5: 1.2, 7: 1.3666666666666667, 9: 0.7},
 {2: 1.2, 4: 1.2, 6: 1.3666666666666667, 8: 1.5333333333333334, 10: 0.95}]
```

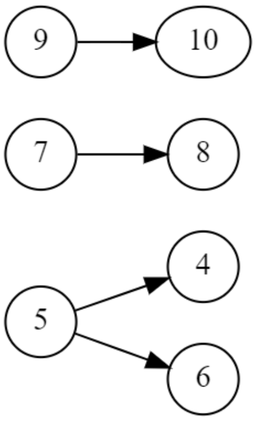
As well as the relationship diagrams showing the transition of hidden state values:



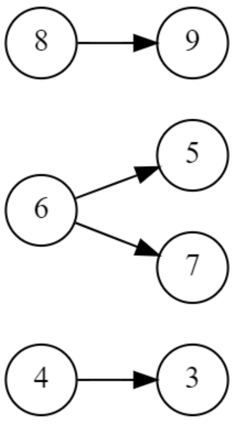
t1~t2



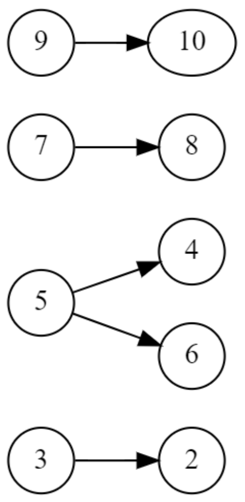
t2~t3



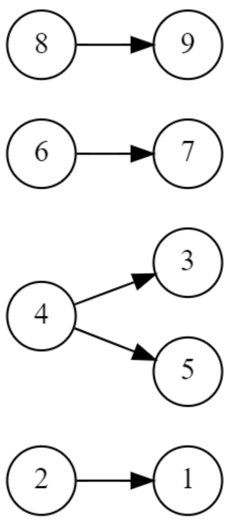
t3~t4



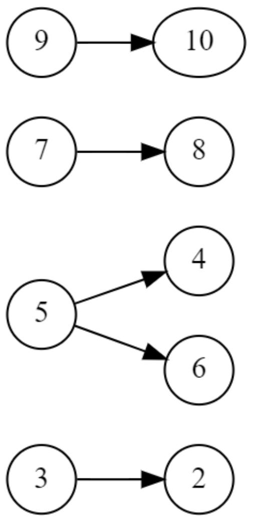
t4~t5



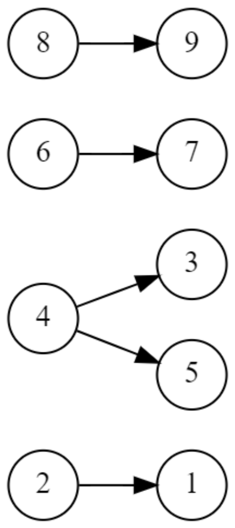
t5~t6



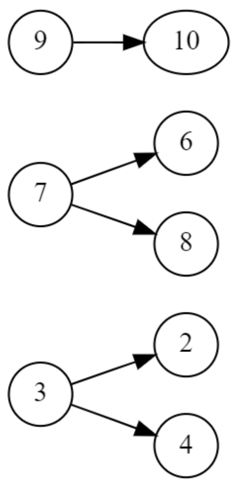
t6~t7



t7~t8



t8~t9



t9~t10

From observing all timesteps' accumulated transition/emission probabilities and the relationship graphs, we can derive the most possible hidden values for observable values [8,6,4,6,5,4,5,5,7,9] are:

[7,6,5,6,5,4,5,6,7,8]

IV. Challenges

The challenge I face in this homework is how to record the transition of hidden values among timesteps. When constructing the algorithm for this task, I need to consider first the calculation of hidden values' transition/emission probabilities; and second, the selection of the most probable hidden values. For the first work, the basic math is well enough; yet for the second one, it's where the challenge hide itself.

Speaking further, the selection of hidden layer values involves a sophisticated work: recording the possible routes and discarding those unsuitable simultaneously. When doing so, I've tried couple of algorithms with both advantages and disadvantages faced. For example, I use the python lists to store the accumulated transition/emission probabilities every rounds of timestep, which fails to capture the routes connecting timesteps. Then, I attempt to directly record the transition of hidden values using dictionaries:

```
transition_34
```

```
{4: 'from5', 6: 'from5', 8: 'from7', 10: 'from9'}
```

```
transition_45
```

```
{3: 'from4', 5: 'from6', 7: 'from6', 9: 'from8'}
```

. This approach gives me the idea of how the values connect one another; however, the way of presenting it is a bit complex.

Finally, after some testing and failure, I attempt to use graphviz package to visualize the recording of hidden values. Even though some glitches still exist with graphviz, like it's incapability to merge one timestep's diagram with another one's, it still offers me a cleaner way to identify the hidden values visually. Nonetheless, I keep reminding myself this approach is not immaculately equipped because once the numbers of Markov model layer increase, identifying value visually like this will be a grueling task. As for how to deal with that situation, there is still a broad knowledge space for me to learn.