FinalStoch

February 19, 2022

1 Q1

[276]:
```
import numpy as np
import json
from tqdm import tqdm

train_data = np.load('train_data.npy')

class HiddenMarkovModel():
    def __init__(self, initial_probabilities, transition_probs, emission_probs):
        self.transition_probs = transition_probs
        self.emission_probs = emission_probs
        self.initial_probabilities = initial_probabilities
        self._log_transition_probs = self._compute_log_probs(transition_probs)
        self._log_emission_probs = self._compute_log_probs(emission_probs)
        if not self._valid_hmm(): raise ValueError("not a valid markov model")

    def baum_welch(self, seq, iterations=100):
        for itr in tqdm(xrange(iterations)):
            initial_update = dict([(s, 0.0) for s in self.transition_probs])
            transition_update1 = dict(
                [(s, dict([(s, 0) for s in self.transition_probs])) for s in␣
↪self.transition_probs])
            transition_update2 = dict([(s, 0) for s in self.transition_probs])
            emission_update1 = dict([(s, dict([(o, 0) for o in self.
↪emission_probs[s]])) for s in self.emission_probs])
            emission_update2 = dict([(s, dict([(o, 0) for o in self.
↪emission_probs[s]])) for s in self.emission_probs])

            for s in seq:
                init, tr1, tr2, em1, em2 = self._baum_welch_iteration(s)
                for s in self.transition_probs:
                    initial_update[s] += init[s]
                for s in self.transition_probs:
                    for x in self.transition_probs[s]:
                        transition_update1[s][x] += tr1[s][x]
```

```
                for s in self.transition_probs:
                    transition_update2[s] += tr2[s]

                for s in self.emission_probs:
                    for o in self.emission_probs[s]:
                        emission_update1[s][o] += em1[s][o]
                        emission_update2[s][o] += em2[s][o]

            self.initial_probabilities = dict([(s, (1 / len(seq)) *␣
↪initial_update[s]) for s in self.transition_probs])
            for i in self.transition_probs:
                for j in self.transition_probs[i]:
                    self.transition_probs[i][j] = (transition_update1[i][j] /␣
↪transition_update2[i])

            for i in self.emission_probs:
                for o in self.emission_probs[i]:
                    self.emission_probs[i][o] = (emission_update1[i][o] /␣
↪emission_update2[i][o])

            self._log_transition_probs = self._compute_log_probs(self.
↪transition_probs)
            self._log_emission_probs = self._compute_log_probs(self.
↪emission_probs)


    def _baum_welch_iteration(self, seq, cons=100):
        forward_probabilities = [dict([(s, 0.0) for s in self.
↪transition_probs]) for _ in xrange(len(seq))]
        eps = 0.00000000000000001
        forward_probabilities[0] = dict(
            [(s, np.log(self.initial_probabilities[s] + eps) + self.
↪_log_emission_probs[s][seq[0]]) for s in
             self.transition_probs])

        for i in xrange(1, len(seq)):
            for j in self.transition_probs:
                a = self._log_emission_probs[j][seq[i]]
                b = self._log_sum_exp([forward_probabilities[i - 1][k] + self.
↪_log_transition_probs[k][j] for k in
                                      self.transition_probs[j]])
                forward_probabilities[i][j] = a + b

        backward_probabilities = [dict([(s, 0.0) for s in self.
↪transition_probs]) for _ in xrange(len(seq))]
```

```
        for j in self.transition_probs:
            backward_probabilities[len(seq) - 1][j] = 0

        for i in range(len(seq) - 2, -1, -1):
            for j in self.transition_probs:
                backward_probabilities[i][j] = self._log_sum_exp([
                    backward_probabilities[i + 1][k] + self.
↪_log_transition_probs[j][k] + self._log_emission_probs[k][
                        seq[i + 1]]
                    for k in self.transition_probs[j]
                ])
        gamma = [dict([(s, 0) for s in self.transition_probs]) for t in␣
↪xrange(len(seq))]
        for t in xrange(len(seq)):
            norm = sum(
                [np.exp(forward_probabilities[t][k] +␣
↪backward_probabilities[t][k] + 1200) for k in
                 self.transition_probs])
            for j in self.transition_probs:
                gamma[t][j] = np.exp(forward_probabilities[t][j] +␣
↪backward_probabilities[t][j] + 1200) / norm

        zeta = [dict([(s, dict([(s, 0) for s in self.transition_probs])) for s␣
↪in self.transition_probs]) for i in
                xrange(len(seq))]
        for t in xrange(len(seq) - 1):
            norm = sum([sum([
                np.exp(forward_probabilities[t][i] + self.
↪_log_transition_probs[i][j]
                       + backward_probabilities[t + 1][j] + self.
↪_log_emission_probs[j][seq[t + 1]] + 1200)
                for j in self.transition_probs[i]]) for i in self.
↪transition_probs])

            for i in self.transition_probs:
                for j in self.transition_probs:
                    if j in self._log_transition_probs[i]:

                        zeta[t][i][j] = np.exp(forward_probabilities[t][i] +␣
↪self._log_transition_probs[i][j]
                                               + backward_probabilities[t +␣
↪1][j] + self._log_emission_probs[j][
                                                   seq[t + 1]] + 1200) / norm
                    else:
                        zeta[t][i][j] = 0
```

```
        init = dict([(s, gamma[0][s]) for s in self.transition_probs])
        trans_prob1 = {}
        trans_prob2 = {}
        for i in self.transition_probs:
            trans_prob2[i] = 0
            for t in range(len(seq) - 1):
                trans_prob2[i] += gamma[t][i]

        for i in self.transition_probs:
            trans_prob1[i] = {}
            for j in self.transition_probs[i]:
                trans_prob1[i][j] = 0
                for t in range(len(seq) - 1):
                    trans_prob1[i][j] += zeta[t][i][j]

        em_prob1 = {}
        for i in self.emission_probs:
            em_prob1[i] = {}
            for o in self.emission_probs[i]:
                em_prob1[i][o] = sum([(seq[t] == o) * gamma[t][i] for t in␣
↪xrange(len(seq))])

        em_prob2 = {}
        for i in self.emission_probs:
            em_prob2[i] = {}
            for o in self.emission_probs[i]:
                em_prob2[i][o] = sum([gamma[t][i] for t in xrange(len(seq))])

        return init, trans_prob1, trans_prob2, em_prob1, em_prob2

    def _log_sum_exp(self, arr):
        a = np.max(arr)
        # print(arr)
        return a + np.log(np.sum([np.exp(x - a) for x in arr]))

    def _compute_log_probs(self, prob_matrix):
        log_prob_matrix = {}
        for i in prob_matrix:
            log_prob_matrix[i] = {}
            for j in prob_matrix[i]:
                log_prob_matrix[i][j] = np.log(prob_matrix[i][j] + 0.
↪00000000001)
        return log_prob_matrix

    def _valid_hmm(self):
        if (self.transition_probs != None) and (self.emission_probs != None):
            valid = True
```

```
            # print("1",set(self.transition_probs.keys()), set(self.
↪emission_probs.keys()))
            valid &= (set(self.transition_probs.keys()) == set(self.
↪emission_probs.keys()))

            return valid
        else:
            return True

    def likelihood(self, seq, log=True):
        alpha = [dict([(s, 0) for s in self.transition_probs]) for _ in␣
↪xrange(len(seq))]
        alpha[0] = dict([(i, np.log(self.initial_probabilities[i]) + self.
↪_log_emission_probs[i][seq[0]]) for i in
                                 self.emission_probs])
        for i in xrange(1, len(seq)):
            for j in self.transition_probs:
                alpha[i][j] = self._log_emission_probs[j][seq[i]] + self.
↪_log_sum_exp([
                    alpha[i - 1][k] + self._log_transition_probs[k][j] for k in␣
↪self.transition_probs[j]])

        if log:
            return self._log_sum_exp(list(alpha[-1].values()))
        else:
            return sum([np.exp(p) for p in alpha[-1].values()])  # return the␣
↪sum of the last row

    def backward_likelihood(self, seq, log=True):
        backward_probabilities = [dict([(s, 0.0) for s in self.
↪transition_probs]) for _ in xrange(len(seq))]

        for j in self.transition_probs:
            backward_probabilities[len(seq) - 1][j] = 0

        for i in range(len(seq) - 2, -1, -1):
            for j in self.transition_probs:
                backward_probabilities[i][j] = self._log_sum_exp([
                    backward_probabilities[i + 1][k] + self.
↪_log_transition_probs[j][k] + self._log_emission_probs[k][
                        seq[i + 1]]
                    for k in self.transition_probs[j]
                ])

        if log:
```

```
            return self._log_sum_exp([(backward_probabilities[0][k] + self.
↪_log_emission_probs[k][seq[0]] + np.log(
                self.initial_probabilities[k])) for k in self.transition_probs])
        else:
            # return sum([np.exp(p) for p in alpha[-1].values()])  # return the␣
↪sum of the last row
            return

    def viterbi(self, seq):
        T = [dict([(s, 0) for s in self.transition_probs]) for _ in␣
↪xrange(len(seq))]
        T[0] = dict([(i, np.log(self.initial_probabilities[i]) + self.
↪_log_emission_probs[i][seq[0]]) for i in
                     self.emission_probs])
        T_bp = [dict([(s, 0) for s in self.transition_probs]) for _ in␣
↪xrange(len(seq))]
        T_bp[0] = dict([(i, i) for i in self.emission_probs])

        for i in xrange(1, len(seq)):
            for j in self.transition_probs:
                T_bp[i][j] = max(self.transition_probs[j],
                                 key=lambda k: T[i - 1][k] + self.
↪_log_transition_probs[k][j] +
                                              self.
↪_log_emission_probs[j][seq[i]])

                prev_state = T_bp[i][j]
                T[i][j] = T[i - 1][prev_state] + self.
↪_log_transition_probs[prev_state][j] + self._log_emission_probs[j][seq[i]]

        # construct the most likely sequence
        state_sequence = [None for i in xrange(len(seq))]
        state_sequence[-1] = max(self.transition_probs,
                                 key=lambda k: T[-1][k])

        for i in xrange(len(seq) - 1):
            i = len(seq) - 2 - i  # go in reverse order
            state_sequence[i] = T_bp[i + 1][state_sequence[i + 1]]

        return state_sequence


transition = {
    0: {1: 0.5, 5: 0.5},
    1: {0: 0.5, 2: 0.5},
    2: {1: 0.5, 3: 0.5},
```

```
    3: {2: (1 / 3), 4: (1 / 3), 8: (1 / 3)},
    4: {3: 0.5, 9: 0.5},
    5: {0: (1 / 3), 6: (1 / 3), 10: (1 / 3)},
    6: {5: 0.25, 7: 0.25, 11: 0.25, 12: 0.25},
    7: {6: (1 / 3), 8: (1 / 3), 12: (1 / 3)},
    8: {3: 0.2, 7: 0.2, 9: 0.2, 12: 0.2, 13: 0.2},
    9: {8: (1 / 3), 4: (1 / 3), 14: (1 / 3)},
    10: {5: 0.5, 15: 0.5},
    11: {6: (1 / 3), 12: (1 / 3), 16: (1 / 3)},
    12: {6: 0.125, 7: 0.125, 8: 0.125, 11: 0.125, 13: 0.125, 16: 0.125, 17: 0.
 ↪125, 18: 0.125},
    13: {8: 0.25, 12: 0.25, 14: 0.25, 18: 0.25},
    14: {9: (1 / 3), 13: (1 / 3), 19: (1 / 3)},
    15: {10: 0.5, 20: 0.5},
    16: {11: (0.25), 12: (0.25), 17: (0.25), 21: 0.25},
    17: {12: (0.25), 16: (0.25), 18: (0.25), 22: 0.25},
    18: {12: 0.2, 13: 0.2, 17: 0.2, 19: 0.2, 23: 0.2},
    19: {14: (1 / 3), 18: (1 / 3), 24: (1 / 3)},
    20: {15: 0.5, 21: 0.5},
    21: {16: (1 / 3), 20: (1 / 3), 22: (1 / 3)},
    22: {17: (1 / 3), 21: (1 / 3), 23: (1 / 3)},
    23: {18: (1 / 3), 22: (1 / 3), 24: (1 / 3)},
    24: {19: 0.5, 23: 0.5}
}

emission = {}
for i in range(25):
    emission[i] = {0: (1 / 11), 1: (1 / 11), 2: (1 / 11), 3: (1 / 11), 4: (1 /␣
 ↪11), 5: (1 / 11), 6: (1 / 11), 7: (1 / 11),
        8: (1 / 11), 9: (1 / 11), 10: (1 / 11)}

initial = {0: 0.04, 1: 0.04, 2: 0.04, 3: 0.04, 4: 0.04, 5: 0.04, 6: 0.04, 7: 0.
 ↪04, 8: 0.04, 9: 0.04, 10: 0.04, 11: 0.04,
           12: 0.04, 13: 0.04, 14: 0.04, 15: 0.04, 16: 0.04, 17: 0.04, 18: 0.
 ↪04, 19: 0.04, 20: 0.04, 21: 0.04, 22: 0.04,
           23: 0.04, 24: 0.04}
```

[ ]:
```
model = HiddenMarkovModel(initial, transition, emission)
model.baum_welch(train_data)

output = json.dumps(model.transition_probs)
f = open("transition_probs.json", "w")
f.write(output)
f.close()

output = json.dumps(model.emission_probs)
f = open("emission_probs.json", "w")
```

```
f.write(output)
f.close()

output = json.dumps(model.initial_probabilities)
f = open("initial_probs.json", "w")
f.write(output)
f.close()
```

[278]:
```
from pathlib import Path
initial_converged = json.loads(Path('initial_probs.json').read_text())
transition_converged = json.loads(Path('transition_probs.json').read_text())
emission_converged = json.loads(Path('emission_probs.json').read_text())

initial_converged = {int(k): v for k, v in initial_converged.items()}
transition_converged = {int(k): {int(x): z for x, z in v.items()} for k, v in␣
↪transition_converged.items()}
emission_converged = {int(k): {int(x): z for x, z in v.items()} for k, v in␣
↪emission_converged.items()}

model =␣
↪HiddenMarkovModel(initial_converged,transition_converged,emission_converged)
```

2 Q2

[275]:
```
test_data= np.load('test_data.npy')

with open('forward_backward_output.txt', 'w') as f:
    for test in tqdm(test_data):
        f_prob = model.likelihood(test, log=True)
        b_prob = model.backward_likelihood(test, log=True)
        print(f_prob, "\t", np.exp(f_prob))
        print(b_prob, "\t", np.exp(b_prob))
        print("--------------------")
        f.write(str(np.exp(f_prob))+"\t"+str(np.exp(b_prob))+"\n")
```

```
  2%|          | 2/100 [00:00<00:13,  7.21it/s]

-87.40822221594169       1.0941866674560593e-38
-87.40822221594168       1.0941866674560748e-38
--------------------
-96.61173593224723       1.1016818575067096e-42
-96.61173593224727       1.1016818575066626e-42
--------------------

  4%|          | 4/100 [00:00<00:12,  7.44it/s]
```

```
-92.86014462227607       4.691930838968532e-41
-92.8601446222761        4.691930838968398e-41
--------------------
-94.30465596383002       1.1066449260434495e-41
-94.30465596383004       1.106644926043418e-41
--------------------

  6%|         | 6/100 [00:00<00:12,  7.41it/s]

-82.78489959778312       1.1142301856444717e-36
-82.78489959778302       1.1142301856445827e-36
--------------------
-92.82644902727122       4.852722004156363e-41
-92.82644902727121       4.852722004156433e-41
--------------------

  8%|         | 8/100 [00:01<00:12,  7.33it/s]

-89.58799416423282       1.2371674629415378e-39
-89.58799416423292       1.2371674629414148e-39
--------------------
-77.89742952855609       1.4776657300430085e-34
-77.89742952855612       1.4776657300429664e-34
--------------------

 10%|         | 10/100 [00:01<00:12,  7.11it/s]

-85.43701721839028       7.855518790205181e-38
-85.43701721839041       7.855518790204177e-38
--------------------
-110.09694789417114      1.5328619896912804e-48
-110.09694789417118      1.532861989691215e-48
--------------------

 12%|         | 12/100 [00:01<00:12,  7.17it/s]

-86.32498227308703       3.232485224350067e-38
-86.32498227308707       3.2324852243499294e-38
--------------------
-89.46806690436863       1.3948009576418284e-39
-89.46806690436867       1.3948009576417688e-39
--------------------

 14%|         | 14/100 [00:01<00:11,  7.32it/s]

-76.12267947149063       8.71646434179358e-34
-76.12267947149067       8.716464341793208e-34
--------------------
-92.233509597237         8.780024653149411e-41
-92.23350959723705       8.780024653149036e-41
--------------------

 16%|         | 16/100 [00:02<00:11,  7.38it/s]
```

```
-79.72073490522291        2.3862983304297343e-35
-79.72073490522287        2.386298330429836e-35
--------------------
-88.9677774652936         2.3003037086900776e-39
-88.96777746529358        2.3003037086901102e-39
--------------------

 18%|         | 18/100 [00:02<00:11,  7.39it/s]

-87.67616308202875        8.37001389620064e-39
-87.67616308202875        8.37001389620064e-39
--------------------
-93.33199988064571        2.9270319758946686e-41
-93.33199988064571        2.9270319758946686e-41
--------------------

 20%|         | 20/100 [00:02<00:10,  7.44it/s]

-74.67811147176513        3.695798363785381e-33
-74.67811147176518        3.6957983637852234e-33
--------------------
-94.41126153955418        9.947411496260651e-42
-94.41126153955422        9.947411496260228e-42
--------------------

 22%|         | 22/100 [00:03<00:10,  7.21it/s]

-96.24420477400315        1.5910101544166728e-42
-96.2442047740032         1.591010154416605e-42
--------------------
-89.2083520623479         1.8084435612305197e-39
-89.20835206234788        1.808443561230545e-39
--------------------

 24%|         | 24/100 [00:03<00:10,  7.25it/s]

-95.20628119183672        4.491976089313837e-42
-95.20628119183671        4.4919760893139004e-42
--------------------
-95.86368650839314        2.3277156928730194e-42
-95.86368650839314        2.3277156928730194e-42
--------------------

 26%|         | 26/100 [00:03<00:10,  7.38it/s]

-88.71821988597522        2.952341961933446e-39
-88.71821988597517        2.952341961933572e-39
--------------------
-91.41467720362908        1.991178184654762e-40
-91.41467720362905        1.9911781846548182e-40
--------------------

 28%|         | 28/100 [00:03<00:09,  7.26it/s]
```

```
-84.46502179894641       2.0763813474967497e-37
-84.46502179894637       2.0763813474968382e-37
--------------------
-90.39670154992757       5.510757934273053e-40
-90.39670154992758       5.510757934272975e-40
--------------------

 30%|         | 30/100 [00:04<00:10,  6.82it/s]

-83.43356570651397       5.8245557722310415e-37
-83.43356570651393       5.82455577223129e-37
--------------------
-85.24263616957687       9.54099105462171e-38
-85.24263616957694       9.54099105462103e-38
--------------------

 32%|        | 32/100 [00:04<00:10,  6.55it/s]

-108.3600088687375       8.706549817337239e-48
-108.36000886873757      8.70654981733662e-48
--------------------
-80.69702990879753       8.989286136980333e-36
-80.69702990879752       8.989286136980461e-36
--------------------

 34%|        | 34/100 [00:04<00:09,  6.82it/s]

-88.93375316634295       2.3799166365846443e-39
-88.9337531663429        2.3799166365847797e-39
--------------------
-79.7246428064165        2.3769911100295996e-35
-79.72464280641643       2.3769911100297685e-35
--------------------

 36%|        | 36/100 [00:05<00:09,  6.52it/s]

-97.77431326612859       3.4447307085015524e-43
-97.77431326612852       3.444730708501797e-43
--------------------
-82.83795802259169       1.056651900881171e-36
-82.83795802259172       1.0566519008811409e-36
--------------------

 39%|        | 39/100 [00:05<00:07,  8.66it/s]

-96.23146604232804       1.6114072465802652e-42
-96.23146604232801       1.611407246580311e-42
--------------------
-85.37467418816064       8.360843721370662e-38
-85.37467418816067       8.360843721370424e-38
--------------------
-89.48353211462116       1.3733960101993952e-39
```

```
-89.48353211462118       1.373396010199356e-39
--------------------

 41%|        | 41/100 [00:05<00:06,  9.76it/s]

-97.0700156740002        6.966711632751486e-43
-97.07001567400025       6.966711632751189e-43
--------------------
-100.72028238787712      1.81024411049534e-44
-100.72028238787705      1.8102441104954685e-44
--------------------
-84.15934908173891       2.8187664468113575e-37
-84.15934908173895       2.8187664468112372e-37
--------------------

 45%|        | 45/100 [00:05<00:04, 11.33it/s]

-88.55051706951673       3.491395654405544e-39
-88.55051706951677       3.491395654405395e-39
--------------------
-83.0923545087351        8.19311307143034e-37
-83.09235450873508       8.193113071430458e-37
--------------------
-84.56911402210281       1.8711148293847844e-37
-84.56911402210285       1.8711148293847046e-37
--------------------

 47%|        | 47/100 [00:05<00:04, 11.91it/s]

-104.77868156501671      3.1274960885960125e-46
-104.77868156501674      3.1274960885959234e-46
--------------------
-85.29474897722375       9.056516524497316e-38
-85.29474897722379       9.05651652449693e-38
--------------------
-78.15197486864379       1.1455882873647317e-34
-78.15197486864399       1.145588287364504e-34
--------------------

 51%|        | 51/100 [00:06<00:04, 12.16it/s]

-88.22345648905699       4.842165982706708e-39
-88.22345648905697       4.842165982706777e-39
--------------------
-89.6800844768694        1.1283249071996753e-39
-89.68008447686934       1.1283249071997396e-39
--------------------
-103.20594217582423      1.5074010498752065e-45
-103.20594217582425      1.507401049875185e-45
--------------------

 53%|        | 53/100 [00:06<00:03, 12.55it/s]
```

```
-87.0192308458694        1.6144634750761637e-38
-87.01923084586947       1.614463475076049e-38
--------------------
-94.67955922563597       7.606587742950879e-42
-94.67955922563598       7.60658774295077e-42
--------------------
-80.08134438715506       1.6638494568055258e-35
-80.08134438715507       1.6638494568055023e-35
--------------------

 57%|       | 57/100 [00:06<00:03, 12.76it/s]

-78.74913040170539       6.305030791610312e-35
-78.74913040170547       6.305030791609774e-35
--------------------
-91.9892161841703        1.1209623255062784e-40
-91.98921618417036       1.1209623255062148e-40
--------------------
-84.86323534339658       1.3943287231048913e-37
-84.86323534339657       1.3943287231049111e-37
--------------------

 59%|       | 59/100 [00:06<00:03, 12.90it/s]

-95.44014252875895       3.5552713056826696e-42
-95.44014252875903       3.555271305682417e-42
--------------------
-77.67738555392103       1.841365853404063e-34
-77.67738555392107       1.8413658534039845e-34
--------------------
-99.46139090397452       6.374802994627711e-44
-99.46139090397452       6.374802994627711e-44
--------------------

 63%|       | 63/100 [00:07<00:02, 13.02it/s]

-79.3590738144842        3.4260378246379254e-35
-79.35907381448418       3.4260378246380227e-35
--------------------
-89.26822913313974       1.7033373840019986e-39
-89.26822913313983       1.7033373840018534e-39
--------------------
-92.34380214104321       7.863145142123464e-41
-92.34380214104314       7.863145142124023e-41
--------------------

 67%|       | 67/100 [00:07<00:02, 13.44it/s]

-97.55762225940336       4.278218734086457e-43
-97.55762225940337       4.278218734086397e-43
--------------------
-74.08024314532335       6.719843889124379e-33
```

```
-74.08024314532337       6.719843889124283e-33
--------------------
-89.46985072590093       1.3923150994831403e-39
-89.46985072590093       1.3923150994831403e-39
--------------------
-93.53214853494003       2.3960948776385237e-41
-93.53214853494005       2.3960948776384895e-41
--------------------

 69%|          | 69/100 [00:07<00:02, 13.45it/s]

-89.75822792080102       1.0435107047283274e-39
-89.75822792080093       1.0435107047284163e-39
--------------------
-94.11244207472551       1.3411758618634945e-41
-94.1124420747255        1.3411758618635137e-41
--------------------
-92.7607900983177        5.182039550720493e-41
-92.76079009831767       5.182039550720641e-41
--------------------

 73%|          | 73/100 [00:07<00:02, 12.92it/s]

-85.71546437227586       5.946299905501121e-38
-85.71546437227586       5.946299905501121e-38
--------------------
-101.76610049832081      6.361274093218553e-45
-101.76610049832074      6.361274093219005e-45
--------------------
-85.80169650808843       5.455024036966462e-38
-85.80169650808841       5.455024036966539e-38
--------------------

 75%|          | 75/100 [00:08<00:02, 12.14it/s]

-95.83219188936275       2.402192870878932e-42
-95.83219188936279       2.4021928708788293e-42
--------------------
-92.20531347753138       9.031110467468759e-41
-92.20531347753135       9.031110467469016e-41
--------------------
-100.95550384875033      1.4308093210985211e-44
-100.95550384875031      1.4308093210985413e-44
--------------------

 79%|          | 79/100 [00:08<00:01, 12.33it/s]

-95.51918532740343       3.2850720290462225e-42
-95.51918532740349       3.285072029046036e-42
--------------------
-83.52554612554279       5.3127113406937134e-37
-83.52554612554289       5.312711340693185e-37
```

```
--------------------
-80.02804019016165       1.7549459603667815e-35
-80.0280401901616        1.7549459603668564e-35
--------------------

 83%|      | 83/100 [00:08<00:01, 13.00it/s]

-79.89658996072365       2.001482776400959e-35
-79.89658996072366       2.0014827764009307e-35
--------------------
-100.26602924093567      2.851128414417456e-44
-100.26602924093565      2.851128414417496e-44
--------------------
-87.77547967364865       7.578679326328344e-39
-87.77547967364868       7.578679326328129e-39
--------------------
-89.77416246064237       1.0270146196328767e-39
-89.77416246064236       1.0270146196328914e-39
--------------------

 87%|      | 87/100 [00:09<00:00, 13.68it/s]

-86.78206685282748       2.0465729876691823e-38
-86.78206685282753       2.046572987669095e-38
--------------------
-82.54134969095658       1.4215014282104077e-36
-82.54134969095655       1.4215014282104481e-36
--------------------
-89.94704926650442       8.639584170620975e-40
-89.94704926650445       8.639584170620728e-40
--------------------
-86.57053316188643       2.5286874455360856e-38
-86.57053316188644       2.5286874455360496e-38
--------------------

 89%|      | 89/100 [00:09<00:00, 13.58it/s]

-77.56779224395252       2.0546405500782094e-34
-77.56779224395248       2.054640550078297e-34
--------------------
-99.93085977925003       3.9863830359225533e-44
-99.93085977925004       3.9863830359224966e-44
--------------------
-105.45007169439032      1.5981434045489174e-46
-105.45007169439035      1.5981434045488719e-46
--------------------

 93%|      | 93/100 [00:09<00:00, 13.30it/s]

-94.75320538551817       7.066522700501191e-42
-94.75320538551826       7.066522700500588e-42
--------------------
```

```
-95.06009364773068       5.199072571359702e-42
-95.06009364773072       5.1990725713594805e-42
--------------------
-90.88138295281098       3.394039371099483e-40
-90.88138295281095       3.3940393710995794e-40
--------------------
```

```
 95%|     | 95/100 [00:09<00:00, 13.49it/s]
```

```
-91.24412018883183       2.3614681685546735e-40
-91.24412018883181       2.361468168554707e-40
--------------------
-90.69811940737132       4.076685628753131e-40
-90.69811940737128       4.076685628753305e-40
--------------------
-96.87895765951693       8.433411191589496e-43
-96.87895765951696       8.433411191589257e-43
--------------------
```

```
 99%|     | 99/100 [00:09<00:00, 12.81it/s]
```

```
-95.96284038104089       2.10798712865435e-42
-95.96284038104089       2.10798712865435e-42
--------------------
-90.09279498686587       7.467861897169648e-40
-90.0927949868658        7.46786189717018e-40
--------------------
-100.38579699711615      2.529311407187092e-44
-100.3857969971162       2.529311407186948e-44
--------------------
```

```
100%|     | 100/100 [00:10<00:00,  9.93it/s]
```

```
-89.59896282555579       1.223671543435187e-39
-89.59896282555573       1.2236715434352567e-39
--------------------
```

[277]:

```
test_data = np.load('test_data.npy')

with open('viterbi_output.txt', 'w') as f:
    for test in tqdm(test_data):
        s = model.viterbi(test)
        print(s)
        print("=====================")
        f.write(str(s)+"\n")
```

```
  8%|         | 8/100 [00:00<00:02, 37.02it/s]
```

```
[16, 11, 12, 16, 12, 18, 17, 18, 17, 22, 23, 22, 23, 22, 21, 16, 21, 20, 21, 22,
23, 22, 21, 22, 23, 22, 17, 12, 8, 9, 4, 3, 2, 3, 4, 3, 4, 3, 8, 7, 8, 13, 12,
```

```
11, 12, 8, 7, 8, 3, 2]
======================
[16, 21, 16, 12, 17, 12, 11, 16, 21, 22, 17, 12, 16, 11, 12, 6, 11, 16, 11, 12,
11, 6, 5, 10, 5, 6, 11, 12, 16, 21, 20, 15, 20, 21, 16, 21, 20, 15, 20, 21, 16,
21, 16, 12, 6, 5, 6, 5, 6, 12]
======================
[21, 16, 12, 11, 6, 5, 6, 11, 6, 11, 6, 7, 8, 7, 8, 12, 17, 12, 11, 12, 7, 8, 9,
4, 3, 4, 9, 8, 12, 16, 21, 20, 21, 20, 15, 20, 21, 16, 12, 17, 12, 18, 17, 22,
17, 12, 13, 18, 17, 18]
======================
[17, 12, 11, 12, 11, 12, 16, 21, 16, 21, 16, 11, 12, 17, 22, 23, 22, 17, 12, 11,
12, 6, 7, 8, 13, 14, 19, 14, 19, 14, 13, 12, 16, 21, 16, 11, 12, 6, 11, 12, 8,
7, 8, 13, 12, 18, 19, 18, 19, 14]
======================
[21, 16, 11, 6, 7, 6, 12, 18, 17, 18, 17, 12, 18, 12, 6, 11, 16, 21, 16, 12, 11,
6, 5, 6, 5, 10, 5, 0, 5, 10, 5, 10, 5, 10, 5, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 0,
1, 0, 1, 0]
======================
[20, 21, 16, 21, 22, 23, 22, 23, 22, 17, 18, 19, 18, 19, 18, 17, 18, 12, 18, 17,
22, 23, 22, 17, 12, 18, 19, 24, 23, 22, 23, 22, 23, 22, 21, 16, 11, 6, 12, 18,
17, 12, 13, 18, 17, 18, 17, 12, 7, 6]
======================
[5, 10, 5, 0, 5, 6, 11, 16, 12, 16, 21, 22, 23, 22, 21, 22, 17, 12, 16, 21, 22,
23, 22, 23, 22, 17, 18, 17, 22, 17, 18, 17, 22, 23, 24, 19, 18, 12, 18, 19, 18,
19, 18, 17, 18, 19, 24, 19, 18, 19]
======================
[18, 12, 11, 12, 18, 12, 17, 12, 18, 17, 22, 17, 22, 23, 22, 23, 22, 23, 22, 23,
22, 23, 22, 17, 22, 23, 22, 17, 22, 17, 22, 23, 22, 17, 22, 23, 22, 17, 12, 17,
22, 23, 22, 21, 20, 15, 20, 15, 20, 15]
======================
[12, 11, 6, 5, 10, 5, 6, 5, 6, 7, 6, 5, 10, 5, 0, 5, 10, 5, 10, 5, 6, 7, 8, 13,
14, 13, 14, 13, 8, 13, 12, 18, 17, 22, 17, 22, 23, 22, 17, 22, 17, 12, 13, 18,
19, 18, 17, 18, 13, 14]
======================

 16%|          | 16/100 [00:00<00:02, 37.65it/s]

[19, 14, 13, 8, 3, 2, 1, 0, 1, 0, 1, 2, 1, 0, 1, 0, 5, 0, 1, 0, 1, 0, 1, 2, 3,
2, 1, 2, 3, 8, 9, 4, 9, 8, 12, 13, 8, 9, 4, 9, 14, 13, 14, 13, 14, 19, 24, 23,
22, 23]
======================
[5, 0, 1, 2, 3, 4, 9, 8, 12, 17, 18, 19, 18, 17, 22, 23, 22, 23, 22, 23, 22, 23,
22, 17, 22, 17, 22, 23, 22, 23, 22, 23, 22, 17, 12, 6, 5, 6, 7, 8, 7, 6, 12, 8,
7, 6, 7, 6, 11, 12]
======================
[15, 20, 15, 20, 15, 20, 15, 20, 21, 22, 23, 22, 23, 22, 17, 18, 17, 22, 17, 22,
23, 22, 17, 18, 12, 16, 12, 13, 18, 19, 14, 19, 18, 12, 17, 22, 23, 24, 19, 24,
23, 22, 17, 18, 17, 12, 16, 12, 16, 21]
======================
```

```
[4, 3, 2, 1, 0, 1, 2, 1, 0, 1, 0, 1, 2, 1, 2, 1, 2, 3, 4, 3, 4, 3, 8, 13, 12,
18, 17, 22, 17, 22, 23, 22, 23, 22, 23, 22, 23, 22, 23, 22, 17, 18, 12, 16, 11,
6, 5, 10, 5, 6]
======================
[15, 20, 15, 20, 21, 20, 21, 20, 15, 20, 21, 16, 11, 6, 5, 10, 5, 6, 11, 12, 11,
16, 21, 20, 21, 16, 11, 12, 18, 12, 11, 16, 11, 12, 18, 17, 22, 23, 22, 21, 20,
15, 20, 15, 20, 15, 20, 15, 20, 15]
======================
[6, 11, 12, 18, 17, 18, 19, 14, 19, 24, 23, 22, 23, 22, 17, 18, 19, 24, 23, 22,
23, 22, 17, 18, 17, 18, 17, 12, 11, 16, 11, 12, 18, 17, 22, 23, 22, 23, 22, 17,
18, 12, 11, 6, 5, 0, 1, 2, 3, 4]
======================
[22, 17, 12, 6, 11, 16, 21, 22, 17, 22, 23, 22, 23, 22, 17, 18, 17, 22, 23, 22,
23, 22, 23, 22, 17, 12, 8, 3, 2, 1, 2, 3, 8, 13, 12, 17, 12, 11, 12, 18, 12, 16,
21, 22, 17, 22, 23, 22, 23, 22]
======================
[16, 11, 12, 11, 12, 11, 16, 12, 16, 11, 12, 17, 22, 21, 22, 23, 22, 17, 18, 17,
22, 23, 22, 23, 22, 23, 22, 17, 22, 21, 16, 21, 20, 21, 22, 23, 22, 23, 22, 23,
24, 19, 14, 13, 12, 6, 5, 6, 5, 10]
======================

 24%|          | 24/100 [00:00<00:02, 37.39it/s]

[2, 1, 0, 5, 10, 5, 6, 7, 12, 16, 21, 22, 23, 22, 23, 22, 17, 22, 23, 22, 23,
22, 17, 18, 19, 18, 12, 8, 7, 8, 7, 6, 12, 7, 8, 12, 17, 12, 16, 11, 12, 11, 12,
17, 18, 19, 24, 23, 22, 17]
======================
[15, 20, 15, 20, 15, 20, 21, 16, 21, 16, 12, 16, 12, 18, 12, 16, 12, 18, 19, 18,
17, 22, 23, 22, 23, 24, 19, 18, 19, 18, 17, 22, 21, 16, 21, 16, 21, 20, 21, 16,
12, 16, 21, 22, 23, 22, 21, 22, 23, 22]
======================
[23, 24, 19, 24, 23, 22, 17, 22, 17, 22, 17, 16, 12, 6, 5, 6, 12, 18, 19, 14, 9,
8, 12, 6, 5, 6, 11, 12, 16, 17, 12, 18, 12, 17, 18, 19, 14, 19, 18, 13, 14, 19,
14, 13, 8, 13, 12, 18, 17, 22]
======================
[13, 8, 9, 14, 19, 24, 23, 22, 23, 22, 17, 12, 13, 14, 9, 8, 9, 8, 3, 2, 3, 8,
9, 4, 9, 8, 13, 14, 19, 18, 17, 12, 17, 22, 23, 24, 19, 14, 19, 14, 19, 18, 17,
22, 23, 22, 23, 22, 23, 22]
======================
[18, 17, 18, 17, 22, 23, 22, 23, 22, 21, 16, 12, 11, 16, 11, 12, 16, 11, 12, 11,
16, 11, 12, 16, 21, 20, 15, 20, 15, 20, 15, 20, 15, 20, 21, 20, 21, 20, 15, 20,
21, 16, 11, 6, 12, 16, 21, 20, 15, 20]
======================
[16, 12, 16, 11, 12, 16, 12, 17, 22, 23, 22, 17, 18, 17, 22, 23, 22, 23, 22, 23,
24, 19, 18, 19, 18, 17, 12, 8, 3, 4, 3, 2, 1, 0, 1, 0, 5, 6, 7, 6, 11, 16, 12,
18, 12, 16, 21, 20, 15, 20]
======================
[10, 5, 10, 5, 6, 11, 6, 12, 11, 6, 5, 10, 5, 10, 5, 6, 11, 16, 21, 20, 21, 16,
12, 17, 16, 12, 17, 18, 12, 6, 5, 0, 1, 2, 1, 2, 1, 0, 1, 2, 3, 4, 3, 8, 3, 4,
```

```
3, 8, 3, 2]
======================
[13, 18, 17, 22, 23, 22, 23, 22, 23, 22, 17, 16, 12, 18, 19, 24, 19, 18, 17, 12,
8, 7, 12, 8, 7, 12, 11, 12, 16, 12, 16, 17, 18, 17, 22, 23, 22, 23, 22, 21, 16,
12, 7, 8, 12, 8, 7, 6, 11, 12]
======================
[9, 14, 19, 14, 19, 14, 19, 14, 19, 18, 17, 18, 17, 18, 17, 22, 21, 16, 12, 16,
11, 6, 5, 6, 12, 11, 16, 11, 12, 11, 12, 6, 7, 6, 11, 12, 17, 18, 19, 18, 12,
11, 16, 21, 20, 15, 20, 15, 20, 15]
======================

 33%|          | 33/100 [00:00<00:01, 39.04it/s]

[11, 12, 11, 12, 16, 21, 20, 15, 20, 15, 20, 15, 20, 15, 20, 21, 16, 21, 16, 12,
18, 12, 16, 12, 17, 12, 16, 21, 22, 17, 18, 19, 18, 12, 11, 16, 12, 16, 11, 6,
12, 18, 12, 16, 12, 18, 17, 12, 16, 21]
======================
[1, 2, 3, 8, 13, 12, 17, 18, 19, 18, 17, 22, 21, 16, 12, 16, 21, 16, 21, 20, 21,
22, 17, 18, 17, 22, 23, 22, 23, 22, 23, 22, 21, 16, 12, 13, 18, 17, 22, 23, 22,
21, 20, 15, 20, 15, 20, 15, 20, 21]
======================
[22, 23, 24, 19, 18, 17, 22, 17, 22, 17, 12, 18, 17, 22, 23, 22, 17, 18, 17, 12,
18, 17, 22, 21, 20, 15, 20, 21, 22, 17, 18, 17, 12, 18, 17, 22, 23, 22, 23, 22,
23, 22, 17, 18, 17, 22, 17, 12, 16, 11]
======================
[6, 5, 6, 7, 8, 9, 4, 3, 4, 3, 4, 9, 4, 3, 4, 3, 8, 7, 6, 5, 6, 12, 16, 11, 12,
8, 3, 2, 1, 2, 1, 2, 1, 0, 5, 0, 5, 10, 15, 20, 21, 20, 21, 16, 11, 6, 7, 6, 5,
6]
======================
[16, 12, 11, 12, 17, 22, 17, 12, 11, 16, 11, 16, 17, 16, 21, 20, 21, 16, 17, 12,
17, 12, 11, 16, 21, 16, 11, 12, 6, 11, 12, 18, 17, 18, 17, 18, 19, 18, 19, 24,
19, 14, 19, 18, 17, 22, 17, 22, 17, 22]
======================
[18, 17, 18, 17, 22, 23, 22, 17, 22, 17, 22, 17, 12, 17, 22, 23, 22, 23, 22, 21,
20, 15, 20, 15, 20, 21, 22, 17, 22, 23, 22, 17, 22, 21, 20, 21, 16, 11, 6, 12,
18, 19, 14, 19, 14, 19, 18, 12, 16, 21]
======================
[15, 20, 21, 20, 21, 22, 23, 22, 23, 22, 23, 22, 23, 22, 17, 12, 17, 22, 17, 12,
17, 16, 21, 20, 21, 16, 12, 7, 8, 12, 16, 21, 22, 23, 22, 23, 22, 17, 18, 17,
12, 16, 12, 17, 18, 19, 18, 12, 16, 21]
======================
[13, 12, 18, 17, 22, 23, 22, 23, 22, 23, 22, 23, 22, 17, 22, 21, 20, 15, 20, 15,
20, 15, 20, 21, 20, 15, 20, 15, 20, 15, 20, 21, 16, 11, 16, 12, 11, 6, 5, 10, 5,
10, 5, 6, 11, 6, 11, 6, 5, 0]
======================
[15, 20, 21, 16, 12, 17, 22, 17, 18, 17, 22, 17, 12, 11, 16, 11, 12, 11, 16, 12,
17, 22, 17, 12, 11, 12, 17, 12, 7, 8, 13, 14, 13, 12, 6, 11, 12, 11, 12, 11, 6,
7, 8, 3, 8, 12, 16, 17, 22, 21]
======================
```

```
[19, 18, 17, 12, 16, 21, 22, 23, 22, 23, 22, 21, 20, 15, 20, 15, 20, 15, 20, 21,
22, 17, 22, 23, 24, 19, 18, 17, 12, 18, 17, 22, 17, 22, 21, 20, 15, 20, 21, 20,
15, 20, 21, 22, 17, 18, 19, 14, 19, 14]
======================

 42%|        | 42/100 [00:01<00:01, 39.03it/s]

[14, 13, 12, 17, 12, 17, 22, 23, 22, 17, 12, 16, 12, 6, 11, 12, 11, 12, 11, 6,
12, 16, 21, 22, 23, 22, 23, 22, 21, 16, 11, 12, 18, 19, 18, 12, 11, 12, 17, 22,
21, 16, 12, 6, 5, 6, 12, 17, 18, 19]
======================
[21, 16, 11, 16, 21, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 21, 22,
17, 12, 16, 21, 20, 15, 20, 21, 16, 12, 18, 12, 17, 22, 23, 22, 23, 22, 23, 22,
23, 22, 23, 22, 23, 22, 17, 18, 12, 18]
======================
[6, 5, 6, 7, 6, 7, 6, 12, 11, 12, 18, 17, 12, 8, 9, 8, 12, 7, 8, 7, 8, 7, 8, 12,
16, 21, 20, 15, 20, 15, 20, 21, 20, 21, 16, 21, 22, 23, 24, 19, 14, 19, 14, 19,
18, 17, 12, 18, 12, 16]
======================
[12, 11, 12, 17, 12, 16, 11, 12, 8, 7, 8, 9, 8, 13, 14, 13, 8, 12, 16, 12, 17,
12, 6, 11, 6, 5, 6, 5, 6, 12, 17, 22, 23, 22, 23, 22, 23, 22, 17, 16, 12, 16,
21, 20, 15, 20, 21, 16, 21, 16]
======================
[0, 5, 0, 1, 2, 3, 4, 9, 14, 19, 24, 23, 24, 19, 18, 19, 24, 23, 22, 21, 16, 11,
12, 16, 21, 22, 23, 22, 23, 22, 23, 22, 17, 12, 8, 9, 4, 3, 2, 1, 0, 5, 10, 5,
10, 5, 6, 11, 16, 21]
======================
[4, 9, 4, 9, 4, 9, 4, 3, 2, 1, 2, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 1, 0, 5, 6, 11,
16, 17, 18, 19, 24, 23, 24, 19, 18, 17, 22, 17, 12, 17, 18, 17, 22, 23, 22, 21,
20, 15, 20, 15]
======================
[18, 19, 18, 17, 12, 11, 12, 7, 8, 3, 2, 3, 8, 9, 8, 13, 12, 13, 14, 19, 14, 19,
14, 13, 14, 19, 18, 12, 11, 16, 21, 22, 23, 22, 23, 22, 17, 18, 17, 22, 23, 22,
23, 22, 23, 22, 17, 22, 17, 22]
======================
[21, 20, 15, 20, 21, 20, 15, 20, 15, 20, 15, 20, 21, 16, 12, 16, 21, 22, 17, 22,
23, 22, 23, 22, 17, 12, 8, 3, 2, 1, 0, 1, 0, 1, 0, 5, 0, 5, 10, 5, 6, 11, 6, 7,
6, 12, 8, 7, 6, 11]
======================
[1, 0, 1, 2, 1, 2, 3, 2, 3, 4, 3, 4, 3, 2, 3, 8, 13, 12, 8, 13, 14, 19, 18, 17,
12, 17, 12, 18, 19, 18, 17, 22, 23, 22, 23, 22, 23, 22, 23, 22, 21, 16, 11, 12,
17, 12, 16, 21, 20, 15]
======================

 52%|        | 52/100 [00:01<00:01, 40.42it/s]

[17, 22, 17, 12, 18, 17, 12, 6, 5, 0, 1, 0, 5, 0, 1, 0, 5, 6, 5, 6, 5, 10, 5, 6,
12, 11, 16, 21, 22, 17, 18, 12, 13, 14, 13, 14, 19, 14, 13, 8, 3, 2, 1, 2, 1, 0,
5, 10, 5, 10]
======================
```

```
[5, 10, 5, 6, 11, 12, 18, 17, 22, 23, 22, 17, 22, 23, 22, 23, 22, 17, 22, 23,
22, 23, 22, 17, 12, 18, 12, 8, 7, 6, 5, 0, 1, 2, 3, 4, 9, 8, 12, 16, 11, 12, 8,
13, 12, 8, 3, 2, 1, 0]
======================
[0, 1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 5, 10, 15, 20, 15, 20, 15, 20, 15, 20,
21, 16, 21, 20, 15, 20, 15, 20, 15, 20, 15, 20, 21, 22, 23, 24, 19, 14, 19, 14,
19, 14, 19, 14, 19, 14, 19]
======================
[22, 23, 22, 23, 22, 23, 24, 19, 14, 13, 8, 12, 17, 22, 17, 22, 23, 22, 17, 18,
17, 22, 21, 20, 15, 20, 21, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 21, 16,
12, 18, 12, 7, 8, 7, 12, 18, 19, 24]
======================
[15, 20, 21, 20, 21, 16, 21, 16, 12, 16, 12, 6, 7, 8, 9, 8, 3, 2, 1, 2, 3, 4, 9,
8, 9, 8, 7, 8, 9, 4, 3, 8, 12, 7, 8, 13, 12, 16, 11, 16, 11, 12, 16, 21, 16, 12,
18, 12, 7, 8]
======================
[5, 10, 5, 0, 1, 0, 1, 2, 3, 4, 9, 8, 12, 17, 18, 17, 18, 17, 18, 17, 18, 12,
18, 19, 14, 13, 14, 13, 18, 17, 12, 11, 12, 6, 7, 8, 9, 4, 9, 4, 9, 14, 19, 18,
19, 24, 19, 18, 17, 12]
======================
[7, 6, 11, 6, 5, 10, 5, 0, 5, 10, 5, 6, 5, 10, 5, 10, 5, 10, 5, 10, 5, 6, 12,
16, 17, 18, 19, 18, 19, 24, 19, 14, 13, 12, 11, 12, 11, 16, 21, 16, 11, 12, 17,
22, 17, 12, 11, 12, 11, 12]
======================
[22, 17, 12, 18, 19, 18, 17, 18, 17, 18, 17, 12, 18, 17, 18, 17, 12, 11, 6, 7,
8, 3, 8, 12, 18, 12, 17, 22, 17, 22, 23, 22, 23, 22, 17, 18, 17, 18, 17, 12, 18,
19, 14, 19, 18, 19, 18, 17, 22, 17]
======================
[22, 23, 22, 17, 22, 23, 22, 23, 22, 23, 22, 23, 22, 23, 22, 23, 22, 23, 22, 21,
22, 23, 22, 23, 22, 23, 22, 17, 22, 23, 22, 17, 22, 23, 22, 23, 22, 23, 22, 23,
22, 23, 22, 17, 18, 17, 22, 23, 22, 23]
======================
[1, 2, 1, 2, 3, 2, 3, 2, 3, 8, 12, 16, 21, 16, 11, 12, 16, 21, 20, 15, 20, 21,
16, 11, 12, 18, 12, 17, 22, 17, 12, 17, 22, 23, 22, 23, 22, 23, 22, 23, 22, 23,
22, 23, 22, 17, 18, 17, 12, 6]
======================

 62%|      | 62/100 [00:01<00:00, 40.80it/s]

[16, 21, 20, 21, 16, 11, 16, 12, 16, 11, 12, 16, 17, 12, 16, 21, 16, 12, 11, 12,
18, 12, 18, 17, 22, 17, 22, 17, 18, 17, 22, 23, 22, 23, 24, 19, 24, 23, 22, 23,
24, 19, 18, 19, 18, 12, 11, 12, 16, 12]
======================
[9, 4, 9, 14, 19, 18, 17, 22, 23, 22, 23, 22, 17, 12, 16, 17, 22, 17, 22, 17,
22, 17, 12, 7, 6, 7, 8, 12, 8, 7, 6, 11, 16, 12, 11, 12, 17, 22, 23, 22, 17, 22,
21, 20, 15, 20, 15, 20, 21, 16]
======================
[0, 1, 0, 5, 6, 12, 11, 12, 11, 12, 16, 17, 18, 17, 12, 18, 17, 18, 17, 12, 16,
12, 18, 17, 18, 17, 18, 12, 17, 18, 17, 18, 19, 18, 17, 12, 16, 17, 22, 23, 22,
```

```
23, 22, 17, 18, 19, 24, 23, 22, 23]
======================
[11, 6, 7, 6, 5, 6, 5, 10, 5, 0, 5, 10, 5, 10, 5, 0, 1, 2, 1, 0, 1, 2, 3, 2, 1,
0, 5, 10, 5, 6, 11, 16, 11, 6, 11, 16, 11, 6, 11, 16, 21, 22, 17, 12, 18, 19,
14, 19, 24, 23]
======================
[14, 13, 18, 19, 18, 12, 6, 7, 6, 7, 6, 5, 10, 5, 0, 1, 0, 1, 0, 5, 0, 5, 0, 1,
0, 1, 2, 1, 0, 5, 0, 1, 2, 3, 2, 1, 2, 1, 0, 5, 10, 5, 0, 5, 10, 15, 20, 15, 20,
15]
======================
[11, 12, 17, 22, 17, 18, 17, 22, 23, 22, 23, 22, 23, 22, 23, 22, 17, 12, 11, 12,
6, 11, 16, 12, 16, 17, 16, 12, 17, 18, 12, 13, 8, 3, 2, 1, 2, 1, 0, 1, 2, 3, 2,
1, 0, 5, 0, 5, 6, 12]
======================
[2, 3, 2, 1, 0, 5, 10, 5, 0, 1, 0, 5, 6, 12, 16, 21, 16, 12, 16, 21, 22, 17, 18,
17, 18, 19, 14, 19, 18, 17, 12, 11, 16, 17, 22, 23, 22, 23, 22, 17, 12, 18, 19,
14, 13, 12, 16, 21, 20, 15]
======================
[12, 16, 11, 6, 11, 12, 17, 22, 23, 22, 23, 22, 23, 22, 17, 18, 17, 18, 17, 18,
17, 12, 11, 16, 12, 17, 18, 12, 7, 6, 11, 12, 18, 17, 22, 17, 12, 6, 5, 0, 1, 2,
1, 2, 1, 2, 1, 0, 1, 0]
======================
[15, 20, 21, 16, 11, 16, 11, 6, 11, 16, 11, 12, 18, 17, 22, 17, 12, 16, 21, 16,
21, 20, 21, 16, 11, 12, 16, 17, 22, 17, 22, 17, 22, 23, 22, 17, 22, 23, 22, 21,
16, 17, 18, 17, 12, 18, 17, 22, 23, 22]
======================
```

72%| | 72/100 [00:01<00:00, 40.71it/s]

```
[6, 5, 10, 5, 10, 5, 6, 7, 8, 9, 8, 7, 6, 5, 0, 5, 10, 5, 6, 12, 16, 11, 12, 6,
11, 6, 5, 10, 5, 10, 5, 0, 5, 0, 1, 0, 1, 2, 1, 2, 3, 8, 7, 8, 3, 8, 12, 6, 7,
6]
======================
[20, 21, 16, 11, 6, 7, 12, 18, 19, 18, 19, 14, 13, 18, 17, 12, 17, 22, 23, 22,
21, 20, 15, 20, 21, 20, 21, 20, 21, 16, 12, 11, 12, 18, 12, 17, 22, 17, 18, 17,
12, 13, 14, 13, 12, 17, 22, 17, 12, 18]
======================
[10, 5, 6, 5, 10, 5, 6, 5, 6, 11, 12, 16, 11, 12, 16, 11, 6, 5, 10, 5, 10, 5, 6,
11, 6, 11, 16, 12, 16, 17, 12, 11, 12, 17, 22, 17, 12, 11, 12, 16, 12, 18, 17,
22, 17, 22, 21, 16, 12, 18]
======================
[15, 20, 15, 20, 15, 20, 21, 16, 21, 22, 17, 18, 13, 14, 13, 14, 19, 18, 12, 11,
12, 11, 16, 11, 12, 13, 18, 19, 18, 17, 16, 12, 8, 9, 4, 9, 4, 9, 8, 9, 4, 9, 8,
9, 4, 3, 2, 1, 0, 1]
======================
[18, 19, 18, 17, 22, 17, 22, 23, 22, 17, 18, 17, 22, 17, 22, 23, 22, 23, 22, 23,
22, 17, 18, 12, 11, 16, 21, 20, 21, 20, 21, 20, 15, 20, 15, 20, 15, 20, 21, 20,
21, 22, 17, 18, 17, 18, 19, 14, 13, 14]
======================
```

```
[20, 15, 20, 15, 20, 21, 22, 23, 22, 17, 18, 12, 17, 22, 17, 12, 16, 21, 20, 21,
16, 11, 16, 12, 17, 22, 17, 22, 23, 22, 17, 22, 17, 18, 17, 12, 16, 21, 16, 12,
16, 21, 20, 15, 20, 15, 20, 15, 20, 15]
======================
[16, 12, 16, 11, 16, 12, 18, 17, 12, 13, 8, 3, 2, 1, 0, 5, 6, 7, 12, 7, 6, 7, 8,
9, 4, 9, 8, 12, 16, 21, 16, 21, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15,
20, 15, 20, 15, 20, 21]
======================
[7, 8, 13, 14, 19, 18, 17, 18, 17, 22, 23, 22, 17, 18, 17, 18, 17, 22, 23, 22,
17, 18, 12, 17, 18, 19, 14, 19, 14, 9, 8, 13, 8, 12, 16, 11, 6, 5, 0, 5, 10, 5,
10, 5, 6, 11, 6, 7, 8, 9]
======================
[5, 10, 5, 10, 5, 0, 1, 0, 1, 2, 3, 2, 1, 2, 1, 2, 3, 8, 9, 4, 3, 8, 7, 6, 7, 6,
11, 12, 16, 12, 6, 7, 8, 3, 8, 12, 17, 18, 19, 14, 19, 14, 19, 14, 19, 18, 19,
18, 17, 18]
======================
[15, 20, 15, 20, 15, 20, 15, 10, 5, 10, 5, 6, 11, 6, 5, 6, 5, 6, 5, 10, 5, 6, 5,
6, 5, 10, 5, 6, 11, 16, 12, 7, 6, 7, 6, 11, 16, 21, 16, 12, 16, 12, 8, 7, 8, 13,
12, 7, 8, 12]
======================

 81%|      | 81/100 [00:02<00:00, 39.59it/s]

[13, 14, 19, 18, 17, 18, 19, 14, 19, 18, 12, 18, 17, 22, 21, 22, 23, 22, 17, 18,
17, 18, 17, 22, 21, 16, 12, 11, 6, 5, 10, 5, 10, 5, 10, 5, 10, 5, 6, 12, 11, 16,
21, 20, 15, 20, 15, 20, 15, 20]
======================
[16, 11, 12, 18, 19, 14, 13, 12, 7, 8, 3, 2, 3, 2, 1, 2, 3, 4, 3, 2, 3, 8, 12,
7, 8, 12, 13, 18, 17, 22, 17, 16, 12, 18, 19, 18, 17, 22, 17, 16, 12, 11, 16,
21, 16, 12, 13, 8, 7, 8]
======================
[13, 14, 19, 24, 23, 24, 19, 18, 17, 12, 11, 12, 11, 6, 7, 8, 7, 6, 12, 11, 12,
16, 21, 16, 21, 16, 11, 12, 11, 16, 12, 13, 8, 7, 8, 3, 4, 9, 14, 19, 24, 23,
22, 23, 22, 17, 22, 23, 24, 19]
======================
[2, 3, 8, 9, 8, 3, 4, 3, 2, 1, 0, 1, 0, 1, 2, 1, 0, 1, 0, 1, 2, 3, 2, 3, 8, 12,
17, 12, 11, 16, 17, 18, 12, 11, 12, 18, 19, 18, 12, 17, 22, 17, 12, 18, 17, 22,
23, 22, 17, 12]
======================
[19, 24, 23, 22, 23, 22, 17, 18, 17, 18, 12, 8, 7, 6, 7, 6, 7, 6, 11, 12, 18,
12, 17, 22, 17, 12, 18, 17, 18, 19, 14, 19, 14, 19, 24, 23, 22, 23, 22, 17, 18,
19, 18, 17, 12, 7, 8, 9, 8, 3]
======================
[4, 9, 8, 3, 4, 3, 4, 3, 4, 9, 8, 9, 4, 3, 2, 1, 2, 1, 0, 1, 2, 1, 2, 1, 0, 1,
2, 1, 0, 5, 6, 11, 6, 11, 12, 16, 21, 16, 11, 12, 16, 21, 20, 21, 20, 15, 20,
21, 20, 15]
======================
[12, 11, 6, 7, 6, 7, 8, 12, 16, 21, 16, 11, 12, 7, 8, 12, 11, 12, 7, 8, 13, 14,
19, 18, 12, 11, 16, 21, 16, 21, 20, 21, 20, 15, 20, 21, 22, 23, 22, 17, 18, 17,
```

```
12, 18, 12, 11, 12, 18, 19, 24]
======================
[2, 3, 8, 13, 18, 17, 22, 17, 12, 7, 8, 12, 16, 12, 18, 17, 12, 18, 17, 22, 23,
22, 21, 16, 12, 18, 12, 16, 21, 22, 23, 22, 21, 20, 15, 20, 15, 20, 21, 16, 17,
22, 23, 22, 21, 20, 21, 20, 15, 20]
======================
[15, 20, 15, 20, 21, 16, 11, 16, 11, 6, 7, 6, 12, 11, 16, 21, 20, 21, 16, 11,
12, 18, 17, 22, 23, 22, 17, 12, 18, 17, 12, 11, 12, 16, 12, 16, 21, 16, 12, 16,
12, 11, 12, 18, 19, 24, 23, 22, 23, 22]
======================
[22, 17, 18, 17, 12, 16, 11, 16, 21, 16, 21, 20, 21, 22, 23, 22, 21, 20, 15, 20,
15, 20, 21, 20, 21, 22, 23, 22, 17, 18, 12, 16, 17, 22, 23, 22, 23, 22, 23, 24,
19, 18, 19, 14, 13, 14, 13, 12, 16, 21]
======================

 89%|      | 89/100 [00:02<00:00, 39.21it/s]

[22, 17, 18, 19, 18, 19, 18, 17, 18, 19, 18, 17, 18, 19, 24, 23, 22, 17, 18, 12,
16, 11, 12, 11, 6, 5, 6, 5, 10, 5, 10, 5, 6, 11, 12, 18, 17, 22, 23, 22, 23, 22,
23, 22, 21, 16, 12, 11, 12, 16]
======================
[2, 1, 0, 1, 2, 1, 0, 5, 6, 5, 10, 5, 10, 5, 6, 5, 6, 5, 10, 5, 0, 5, 0, 1, 0,
1, 2, 1, 0, 5, 6, 5, 0, 1, 2, 1, 0, 5, 6, 12, 8, 9, 8, 12, 17, 18, 17, 12, 16,
21]
======================
[5, 10, 5, 10, 5, 0, 1, 2, 3, 2, 3, 2, 1, 2, 3, 4, 3, 8, 3, 4, 3, 4, 9, 8, 12,
16, 11, 12, 18, 12, 16, 11, 12, 13, 8, 7, 6, 12, 11, 16, 21, 20, 21, 16, 12, 7,
8, 7, 6, 5]
======================
[15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20, 15, 20,
15, 20, 15, 20, 15, 20, 15, 20, 21, 22, 23, 22, 23, 22, 23, 22, 23, 22, 21, 20,
21, 16, 21, 20, 15, 20, 15, 20, 15, 20]
======================
[10, 5, 6, 5, 6, 5, 6, 5, 6, 11, 16, 21, 16, 12, 16, 21, 16, 12, 6, 11, 6, 5, 6,
12, 13, 8, 9, 4, 3, 2, 3, 8, 7, 8, 7, 8, 7, 12, 16, 21, 16, 21, 20, 21, 22, 23,
22, 17, 12, 16]
======================
[5, 10, 5, 6, 12, 18, 12, 18, 12, 7, 8, 12, 13, 14, 13, 12, 16, 21, 20, 21, 20,
21, 16, 12, 13, 14, 19, 24, 19, 24, 23, 22, 17, 12, 6, 5, 10, 5, 10, 5, 6, 11,
12, 11, 16, 21, 16, 21, 16, 12]
======================
[5, 10, 5, 10, 5, 10, 5, 10, 5, 10, 5, 0, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 5, 6, 5,
6, 12, 11, 12, 18, 12, 17, 18, 12, 16, 17, 12, 8, 3, 8, 9, 8, 12, 8, 13, 12, 8,
7, 8, 9]
======================
[6, 7, 6, 11, 6, 5, 6, 12, 17, 18, 17, 18, 19, 18, 12, 18, 12, 8, 7, 6, 5, 10,
5, 6, 5, 10, 5, 6, 11, 16, 21, 20, 21, 20, 21, 16, 11, 6, 5, 6, 5, 6, 5, 10, 5,
6, 11, 12, 8, 7]
======================
```

```
100%|     | 100/100 [00:02<00:00, 39.40it/s]

[14, 19, 24, 23, 22, 23, 24, 19, 18, 17, 18, 17, 22, 17, 12, 16, 12, 18, 17, 22,
21, 16, 12, 11, 16, 12, 7, 8, 12, 6, 11, 12, 8, 13, 14, 19, 24, 23, 24, 19, 18,
17, 12, 11, 6, 7, 8, 7, 6, 11]
======================
[7, 6, 7, 8, 13, 12, 6, 7, 8, 12, 17, 18, 12, 17, 22, 23, 22, 17, 22, 23, 22,
17, 12, 13, 14, 13, 8, 12, 18, 17, 18, 12, 11, 12, 17, 18, 17, 18, 17, 18, 12,
18, 12, 11, 12, 11, 12, 7, 8, 12]
======================
[12, 18, 17, 18, 19, 24, 23, 22, 23, 24, 19, 14, 9, 4, 9, 8, 13, 18, 19, 24, 23,
18, 17, 22, 23, 22, 17, 18, 17, 12, 16, 21, 20, 15, 20, 15, 20, 21, 20, 21, 20,
15, 20, 15, 20, 15, 20, 15, 20, 15]
======================
[23, 22, 17, 22, 17, 12, 16, 21, 16, 21, 16, 12, 16, 21, 16, 21, 16, 12, 17, 22,
21, 20, 21, 16, 21, 16, 11, 16, 12, 11, 6, 5, 6, 7, 8, 13, 12, 18, 19, 24, 23,
22, 17, 22, 23, 24, 19, 18, 12, 6]
======================
[5, 10, 5, 0, 5, 10, 5, 6, 7, 8, 7, 8, 3, 2, 1, 0, 1, 0, 1, 2, 1, 2, 1, 2, 3, 2,
1, 0, 1, 0, 5, 0, 1, 2, 3, 4, 9, 8, 7, 8, 13, 14, 19, 24, 23, 22, 17, 12, 16,
21]
======================
[19, 14, 19, 14, 19, 14, 13, 8, 12, 17, 22, 23, 22, 23, 22, 21, 20, 15, 20, 21,
16, 12, 11, 16, 12, 7, 8, 9, 8, 12, 16, 11, 6, 7, 6, 5, 0, 1, 0, 5, 10, 5, 6, 5,
10, 5, 0, 5, 6, 11]
======================
[23, 22, 21, 16, 12, 17, 18, 12, 16, 11, 6, 11, 16, 21, 16, 12, 8, 7, 12, 18,
12, 17, 22, 17, 22, 17, 12, 11, 16, 21, 20, 15, 20, 21, 22, 23, 22, 23, 22, 23,
22, 17, 22, 17, 16, 12, 7, 8, 12, 6]
======================
[5, 6, 5, 6, 11, 6, 11, 16, 12, 17, 12, 16, 17, 22, 23, 22, 23, 22, 21, 20, 15,
20, 15, 20, 21, 22, 17, 22, 17, 12, 8, 9, 4, 9, 4, 3, 4, 3, 2, 1, 0, 1, 2, 1, 0,
5, 10, 5, 6, 5]
======================
```

3 Question 4

[168]:
```python
def convert_dictionary_to_matrix(dictionary):
    rows=len(dictionary)
    matrix=np.zeros((rows,rows),dtype=np.float64)
    for i in range(rows):
        for j in range(rows):
            value = dictionary[i].get(j)
            matrix[i,j] = value if value is not None else 0
    return matrix
```

[169]:
```python
def get_stationary_distribution(Q):
    evals, evecs = np.linalg.eig(Q.T)
    evec1 = evecs[:,np.isclose(evals, 1)]
    evec1 = evec1[:,0]
    stationary = evec1 / evec1.sum()
    stationary = stationary.real
    stationary = stationary / stationary.sum()
    return stationary
```

[170]:
```python
def remove_edge(dictionary, left, right):
    dictionary[left].pop(right)
    dictionary[right].pop(left)
```

[171]:
```python
def normalize_matrix(matrix):
    """
    input: two dimensional matrix
    returns a matrix which its rows sum to 1
    """
    return matrix / np.sum(matrix,1)[:,None]
```

[216]:
```python
import copy
transition_matrix = convert_dictionary_to_matrix(model.transition_probs)
transition2 =copy.deepcopy(model.transition_probs)

remove_edge(transition2,8,12)
remove_edge(transition2,12,13)
remove_edge(transition2,18,19)
remove_edge(transition2,16,17)
remove_edge(transition2,6,12)

transition_matrix2 = convert_dictionary_to_matrix(transition2)
transition_matrix2 = normalize_matrix(transition_matrix2)

stationary_distribution=get_stationary_distribution(transition_matrix)
stationary_distribution
```

[216]:
```
array([0.02112507, 0.0285067 , 0.02265319, 0.01969918, 0.01347683,
       0.03876496, 0.04224324, 0.02300952, 0.03733263, 0.01653681,
```

```
        0.01811561, 0.04662704, 0.11368048, 0.02230126, 0.01947599,
        0.02869246, 0.07049459, 0.07837297, 0.05633585, 0.03169953,
        0.0491273 , 0.05711506, 0.08066256, 0.05072891, 0.01322225])
```

[180]:
```
from random import uniform
import random
import pandas as pd
def w_choice(seq):
    """
    choose an item from list of items whose probability is given
    example: input [0.3,0.4,0.3]
    output: it may choose from [0,1,2] with given probabilty
    """
    total_prob = sum([item for item in seq])
    chosen = random.uniform(0,total_prob)
    cumulative = 0
    for i in range(len(seq)):
        cumulative += seq[i]
        if cumulative > chosen:
            return i
```

[181]:
```
state_count = len(transition_matrix)
a_array = np.zeros((state_count,state_count))
for i in range(state_count):
    for j in range(state_count):
        r_nominator = stationary_distribution[j]*transition_matrix2[j,i]
        r_denominator = stationary_distribution[i]*transition_matrix2[i,j]

        a = 0
        if r_denominator == 0:
            a = 1
        else:
            a = min(r_nominator / r_denominator,1)

        a_array[i,j] = a
```

[182]:
```
new_transition_matrix=a_array*transition_matrix2
stationary2=␣
 ↪get_stationary_distribution(normalize_matrix(new_transition_matrix))
stationary2
```

[182]:
```
array([0.02180875, 0.02994216, 0.0234925 , 0.02023764, 0.01302789,
       0.04241142, 0.03536303, 0.02535294, 0.03216247, 0.01639951,
       0.01965901, 0.0513758 , 0.10217224, 0.02017778, 0.02124298,
       0.03131307, 0.07050133, 0.08362803, 0.05162414, 0.01804102,
       0.05382912, 0.06143372, 0.08667532, 0.05355925, 0.01456888])
```

[241]:
```
### check if they are close
(stationary2-stationary_distribution) < 0.01
```

[241]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

[100]:
```
### sampling
iterations = 1000
sample_items = 100
x_array = np.zeros((iterations,sample_items))
for e in range(iterations):
    x_array[e,0] = int(random.uniform(0,state_count))
    for i in range(sample_items-1):
        x = int(x_array[e,i])
        y = w_choice(transition_matrix2[x])

        if random.uniform(0,1) <= a_array[x,y]:
            x_array[e,i+1] = y
        else:
            x_array[e,i+1] = x
```

[113]:
```
x_array=x_array.reshape(-1)
imperical_stationary=pd.Series(x_array).value_counts().sort_index()/len(x_array)
np.array(imperical_stationary)-stationary_distribution < 0.01
```

[113]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

[114]:
```
stationary2=␣
 ↪get_stationary_distribution(normalize_matrix(new_transition_matrix))
stationary2
```

[114]:
```
array([0.02180875, 0.02994216, 0.0234925 , 0.02023764, 0.01302789,
       0.04241142, 0.03536303, 0.02535294, 0.03216247, 0.01639951,
       0.01965901, 0.0513758 , 0.10217224, 0.02017778, 0.02124298,
       0.03131307, 0.07050133, 0.08362803, 0.05162414, 0.01804102,
       0.05382912, 0.06143372, 0.08667532, 0.05355925, 0.01456888])
```

[115]:
```
(stationary2-stationary_distribution) < 0.01
```

[115]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

[49]:
```

```

3.0.1 Question 5

[157]:
```
min_matrix=np.where(transition_matrix<transition_matrix.
↪T,transition_matrix,transition_matrix.T)
proposal_stationary=get_stationary_distribution(normalize_matrix(min_matrix))
proposal_stationary
```

[157]:
```
array([0.03623193, 0.05314085, 0.04338589, 0.039074  , 0.04035088,
       0.05016627, 0.0429694 , 0.02380878, 0.04097378, 0.04377247,
       0.0249732 , 0.03266443, 0.05314085, 0.02670919, 0.03966445,
       0.03096373, 0.04686786, 0.04565732, 0.03981049, 0.04184281,
       0.04970957, 0.04558968, 0.05314085, 0.03842504, 0.01696628])
```

Yes, It is logical,

4 Extra Section for testing purposes

[191]:
```
import numpy as np
transition = np.array([[0.1,0.2,0.7],
                                        [0.3,0.6,0.1],
                                        [0.3,0.4,0.3] ])
```

[192]:
```
np.linalg.matrix_power(transition,10000)
```

[192]:
```
array([[0.25  , 0.4375, 0.3125],
       [0.25  , 0.4375, 0.3125],
       [0.25  , 0.4375, 0.3125]])
```

[193]:
```
stationary_dist=get_stationary_distribution(transition)
stationary_dist
```

[193]:
```
array([0.25  , 0.4375, 0.3125])
```

[205]:
```
def metropolis_hastings(sample_items, transition, stationary_dist):
    sample = np.zeros(sample_items,dtype=np.int32)
    for i in range(sample_items-1):
            x = sample[i]
            y = w_choice(transition[x])

            r_nominator = stationary_dist[y]*transition[y,x]
            r_denominator = stationary_dist[x]*transition[x,y]

            a = 0
            if r_denominator == 0:
                a = 1
```

```
            else:
                a = min(r_nominator / r_denominator,1)

            if random.uniform(0,1) <= a:
                sample[i+1] = y
            else:
                sample[i+1] = x
    return sample
print(metropolis_hastings(10,transition,stationary_dist))
```

```
[0 0 0 1 1 2 0 0 2 0]
```

[213]:
```
import pandas as pd
number_of_iterations = 100000
samples_per_iteration = 100
samples=np.zeros((number_of_iterations,samples_per_iteration))
for i in range(number_of_iterations):
    samples[i]=
 ↪metropolis_hastings(samples_per_iteration,transition,stationary_dist)

samples=samples.reshape(-1)
imperical_stationary=pd.Series(samples).value_counts().sort_index()/len(samples)
imperical_stationary
```

[213]:
```
0.0    0.260744
1.0    0.427493
2.0    0.311763
dtype: float64
```

[ ]:
```
```