Task 1

Write out a proof that P is reversible with respect to μ. Conclude that μ is an invariant probability distribution for P.

To show that P is reversible with respect to μ, we must show that

$$\mu_i P_{ij} = \mu_j P_{ji}$$

In the case $i = j$ this is trivial. Otherwise if $i! = j$,

$$\mu_i P_{ij} = \mu_i q_{ij} \min\left\{1, \frac{\mu_j}{\mu_i}\right\} = q_{ij} \min\left\{\mu_i, \mu_j\right\}$$

and similiarly

$$\mu_j P_{ji} = \mu_j q_{ji} \min\left\{1, \frac{\mu_i}{\mu_j}\right\} = q_{ji} \min\left\{\mu_j, \mu_i\right\}$$

If a probability mass function μ_i satisfies the above, then it is invariant, because $\forall j, \pi_j = \sum_i \pi_i P_{ij}$

$$\sum_i \pi_i P_{ij} = \sum_i \pi_j P_{ji} = \pi_j \sum_i P_{ji} = \pi_j \quad \square.$$

Task 2

Write out a proof that P is irreducible, and that if μ isn't perfectly uniform, then P is aperiodic. *[Hint: show that if $i \longrightarrow j$ under Q, then $i \longrightarrow j$ under P. For aperiodicity, consider a site where a transition could be rejected.]*

For the chain P to be irreducible, we must show that $\forall i, j \in \mathcal{S}$, $i \longrightarrow j$. It is sufficient to show that if $i \longrightarrow j$ under Q, then $i \longrightarrow j$ under P. Suppose $i \longrightarrow j$ under Q. There are two cases to consider.

Case 1: If $\mu_j \geq \mu_i$ then we always accept the move and $p_{ij} = q_{ij}$, and since Q is symmetric, $\exists n$ such $q_{ij}^{(n)} > 0$. Therefore $\forall i, j \in \mathcal{S}$ $i \longrightarrow j$.

Case 2: If $\mu_j < \mu_i$ then $p_{ij} = q_{ij} \frac{\mu_j}{\mu_i} > 0$, so $p_{ij} > 0$.

Suppose that μ is not perfectly uniform, then w.l.g. there exists an $l, i \in \mathcal{S}$ such that $\frac{\mu_l}{\mu_i} < 1$, and $p_{ii} > 0$. Then the period of state i is 1, but since P is irreducible, it has only one class, so the period of P must be $\gcd\{1, \ldots\} = 1$. So P is aperiodic $\square$.

Task 3

We propose the following method to define Q:

Given a current state which is the tour $x = \begin{pmatrix} x_0 & x_1 & \cdots & x_{19} \end{pmatrix}$, randomly pick $i, j \in \{0, \ldots 19\}$ such that $i' \neq j'$ and $i = \min(i', j'), j = \max(i, j)$. Define T to be the transformation which swaps cities i and j, ie.

$$\begin{pmatrix} x_0 & \cdots & x_i & \cdots & x_j & \cdots x_{19} \end{pmatrix} \longrightarrow \begin{pmatrix} x_0 & \cdots & x_j & \cdots & x_i & \cdots x_{19} \end{pmatrix}$$

If all resulting tours x' are valid, then there are $\binom{20}{2} = 190$ possible outcomes, so the probability q_{ij} for $i \neq j$ is

$$q_{ij} = \frac{1}{\binom{20}{2}} = \frac{1}{190}$$

In the event that state x_j is not valid due it containing a forbidden connection between two cities, we remain in state x_i. Suppose f denotes the number of forbidden states that neighbor state x_i. Then the probability of remaining in state x_i are

$$q_{ii} = \frac{f}{\binom{20}{2}}.$$

When a state x_i has more forbidden neighbors, the probability of remaining in state x_i increases proportionaly. The effect of this is that $q_{ij} = \frac{1}{190}$ for all $i, j \in \mathcal{S}$ where K is a constant, and that implies

$$q_{ij} = q_{ji} \quad \square.$$

1 Results

Table 1: Results of 10 runs of Metropolis-Hastings algorithm

Run	$E[f(x)]$ estimate
1	151.7582975592603
2	151.78521559558445
3	151.71995644022766
4	151.72257839116807
5	151.7989403512341
6	151.77048793781816
7	151.73434402770206
8	151.7712540561242
9	151.723880973142
10	151.7255789619957

Figure 1: $l(i,j) \sim \text{Gamma}(k = 7.5)$

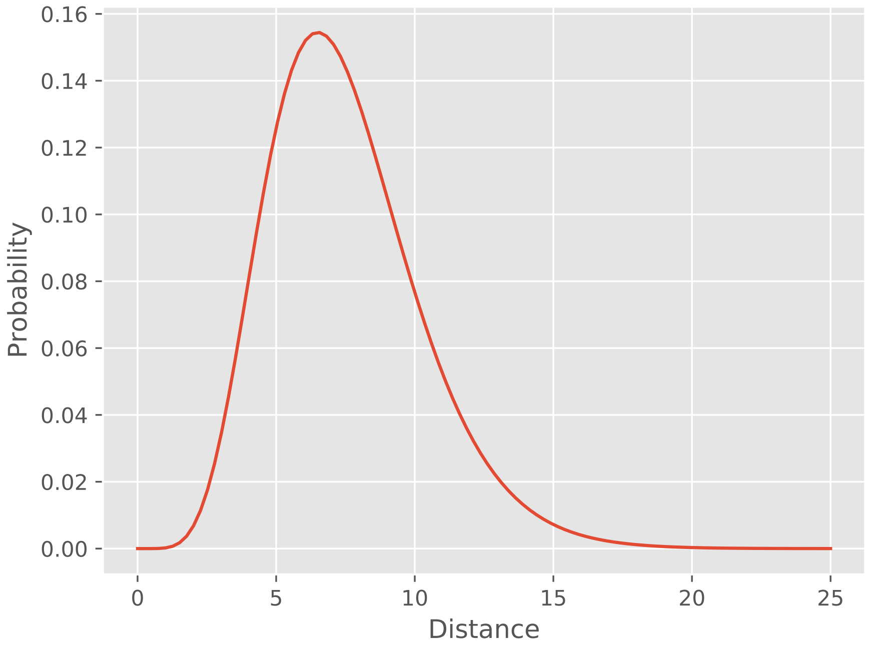

Figure 2: Several runs of Metropolis-Hastings algorithm $n = 10^6$ and $m = 10^4$

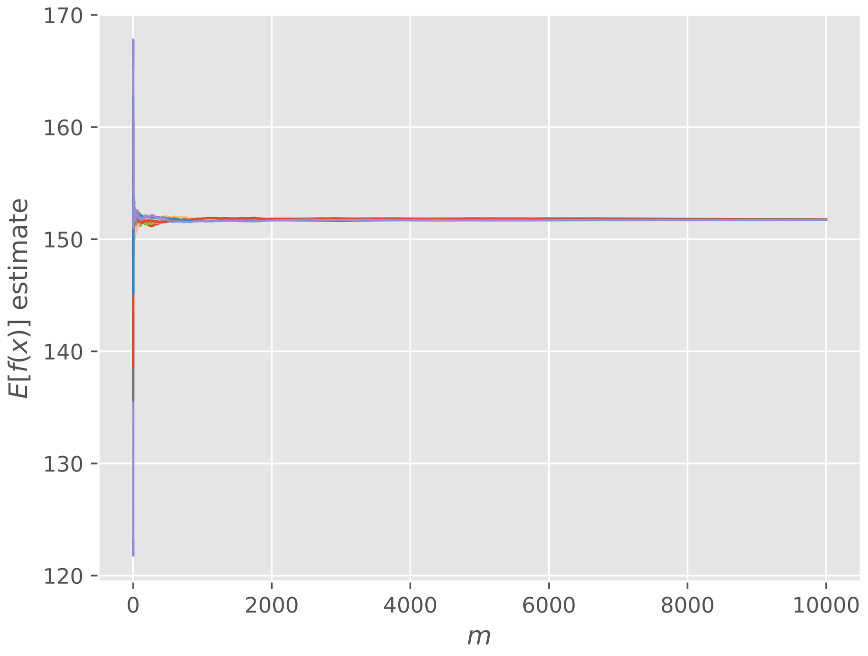

Table 1 shows the convergence of the MCMC. The burn-in phase is not depicted.

Appendix

Table 2: Distances between pairs of cities (i, j) for $i, j \in \{0, 9\}$ and $i \neq j$

i	j	$l(i,j)$	i	j	$l(i,j)$	i	j	$l(i,j)$	i	j	$l(i,j)$	i	j	$l(i,j)$
0	1	11.26	2	11	13.25	8	10	4.95	11	4	7.32	16	18	6.79
0	2	11.92	2	12	8.08	8	11	10.29	11	5	7.04	16	19	5.14
0	3	6.02	2	13	15.34	8	12	7.86	11	6	6.13	17	2	4.03
0	4	5.74	2	14	4.70	8	13	5.82	11	7	5.34	17	3	4.16
0	5	8.05	2	15	13.81	8	14	3.23	11	12	14.47	17	4	6.58
0	6	14.37	2	18	8.12	8	15	8.22	11	13	6.84	17	5	5.15
0	7	9.05	2	19	4.63	8	16	6.86	11	14	13.87	17	6	7.10
0	8	5.09	3	4	4.27	8	17	7.35	11	15	8.47	17	7	7.25
0	9	10.81	3	5	9.03	8	18	4.50	12	5	6.70	17	10	6.54
0	10	3.00	3	6	3.46	8	19	15.86	12	6	7.86	17	11	8.55
0	11	7.43	3	7	3.77	9	2	11.78	12	7	6.49	17	12	3.17
0	12	6.67	3	12	8.88	9	3	5.63	12	13	6.71	17	13	8.76
0	13	3.04	3	13	9.89	9	4	6.60	12	14	11.08	17	14	9.74
0	14	11.61	3	14	7.47	9	5	4.36	12	15	6.48	17	15	9.77
0	15	6.37	3	15	9.96	9	6	8.22	13	5	9.27	17	18	6.96
0	16	6.88	4	5	14.08	9	7	9.63	13	6	4.28	17	19	6.61
0	17	7.25	4	6	4.28	9	10	7.84	13	7	13.30	18	3	3.94
0	19	6.64	4	7	5.54	9	11	6.62	13	14	9.35	18	4	10.56
1	2	6.89	4	12	6.72	9	12	6.46	13	15	5.58	18	5	13.23
1	3	5.27	4	13	12.53	9	13	5.10	14	7	10.12	18	6	13.52
1	4	5.11	4	14	9.28	9	14	6.23	14	15	5.13	18	7	10.42
1	5	7.55	4	15	4.94	9	15	8.94	15	7	9.11	18	11	6.20
1	6	6.03	5	6	4.72	9	18	10.98	16	1	7.09	18	12	8.71
1	9	7.93	5	7	5.91	9	19	5.59	16	2	9.97	18	13	6.36
1	11	11.22	5	14	8.21	10	3	7.85	16	3	6.82	18	14	5.02
1	12	5.48	5	15	8.39	10	4	9.40	16	4	5.74	18	15	14.79
1	13	8.99	6	7	5.08	10	5	7.12	16	5	6.00	18	19	5.46
1	14	2.97	6	14	10.78	10	6	7.28	16	6	4.76	19	3	8.73
1	15	5.40	6	15	5.37	10	7	6.48	16	7	9.43	19	4	6.78
1	17	9.07	8	1	7.00	10	11	12.31	16	9	8.66	19	5	10.94
1	18	5.72	8	2	6.02	10	12	4.34	16	10	4.08	19	6	8.01
2	3	8.26	8	3	10.55	10	13	3.35	16	11	11.61	19	7	13.82
2	4	4.77	8	4	5.51	10	14	2.68	16	12	9.64	19	11	4.46
2	5	3.99	8	5	3.72	10	15	9.40	16	13	8.25	19	12	4.87
2	6	6.67	8	6	6.77	10	18	4.39	16	14	10.74	19	13	7.57
2	7	9.61	8	7	8.01	10	19	4.96	16	15	7.20	19	14	7.19
2	10	7.26	8	9	12.26	11	3	8.05	16	17	7.78	19	15	4.11

Table 2 shows distances on between the 185 pairs of 20 cities. Disntances are generated randomly from a gamma distribution. Refer to listing 1 for details.

Listing 1: Traveling Salesperson with MCMC

```
import random
from pprint import pprint
import time
import json
import pickle
import math
from pathlib import Path

import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.style.use('ggplot')
# mpl.use('module://matplotlib-backend-kitty')
mpl.use('Agg')
import scipy.stats.distributions as distributions

DATA_PATH = Path(__file__).absolute().parents[1] / 'data' / 'data.pkl'
# random.seed(0)
# np.random.seed(0)

random.seed(0)

n_cities = 20
n_ban = 5
pairs = {frozenset([i, j]) for i in range(n_cities) for j in range(i + 1, n_cities)
    if i != j}

gamma = distributions.gamma(a=7.5)
bans = random.sample(pairs, n_ban)

print(bans)

print({7, 13} in bans)
exit()

fig, ax = plt.subplots(1)

x = np.linspace(0, 25, 100)
z = gamma.pdf(x)
ax.plot(x, z)
ax.set_xlabel('Distance')
ax.set_ylabel('Probability')
fig.savefig('fig1.pdf', dpi=200)
# plt.show()
# exit()

distances = {p: gamma.rvs() for p in pairs.difference(bans)}

# pprint(distances)
# pprint(bans)

with open('table1-1.tex', 'wt') as fh:

    vals = list((i, j, d) for (i, j), d in distances.items())
    vals = sorted(vals, key=lambda e: (e[0], e[1]))

    row = ' & '.join(('{} & {} & {:.2f}',) * 5) + ' \\\\\n'

    for i in range(len(vals) // 5):
        row_s = row.format(*vals[i], *vals[i + 37], *vals[i + 2 * 37], *vals[i + 3 *
 37], *vals[i + 4 * 37])
        fh.write(row_s)

with open('table1-2.tex', 'wt') as fh:

    vals = list((i, j) for (i, j) in bans)
    vals = sorted(vals, key=lambda e: (e[0], e[1]))

    for e in vals:
        fh.write('{} & {} \\\\\n'.format(*e))


def legal(t):
    for i in range(n_cities):
        if {t[i], t[(i + 1) % n_cities]} in bans:
            return False
    return True


def dist(t):
    return sum(distances[frozenset((t[i], t[(i + 1) % n_cities]))] for i in range(
    n_cities))


def mcmc(n_burn, m_samples):

    # Pick random initial state; repeat until we have legal first tour.
    t0 = tuple(random.sample(range(n_cities), n_cities))
    while not legal(t0):
        t0 = tuple(random.sample(range(n_cities), n_cities))

    # Do burn-in.
    t = t0
    for k in range(n):

        i, j = random.sample(range(n_cities), 2)

        t_prime = list(t)
        t_prime[i], t_prime[j] = t_prime[j], t_prime[i]
        t_prime = tuple(t_prime)
        if legal(t_prime):
            t = t_prime

    # Sample MCMC.
    d = 0.
    d += dist(t)
    running_avg = [d, ]
    for k in range(m):

```

```python
        i, j = random.sample(range(n_cities), 2)

        t_prime = list(t)
        t_prime[i], t_prime[j] = t_prime[j], t_prime[i]
        t_prime = tuple(t_prime)
        if legal(t_prime):
            t = t_prime

        d += dist(t)

        if k % 100 == 0:
            running_avg.append(d / (k + 2))

    return d / (m + 1), running_avg


n = 100000
m = 1000000

num_runs = 10

fig, ax = plt.subplots(1, 1)

for i in range(num_runs):
    d_exp, plt_d_exp = mcmc(n, m)
    ax.plot(list(range(len(plt_d_exp))), plt_d_exp, linewidth=1.0)
    print(f'{i} -> E[f(t)] = {d_exp}')

ax.set_ylabel('$E[ f(x) ]$ estimate')
ax.set_xlabel('$m$')

fig.savefig('fig2.pdf', dpi=200)
```