analysis

April 25, 2025

[48]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[49]:
```python
q_table_df = pd.read_csv("q_table.csv")
q_table_df.head()
```

[49]:

	x	y	p1	p2	p3	p4
0	0	0	0.00000	0.0	0.0	0.000000
1	0	1	0.00000	0.0	0.0	0.002195
2	0	2	0.00000	0.0	0.0	0.025200
3	0	3	0.00000	0.0	0.0	0.041209
4	0	4	0.00244	0.0	0.0	0.000000

[50]:
```python
episode_times_df = pd.read_csv("episode_times.csv")
episode_times_df.head()
```

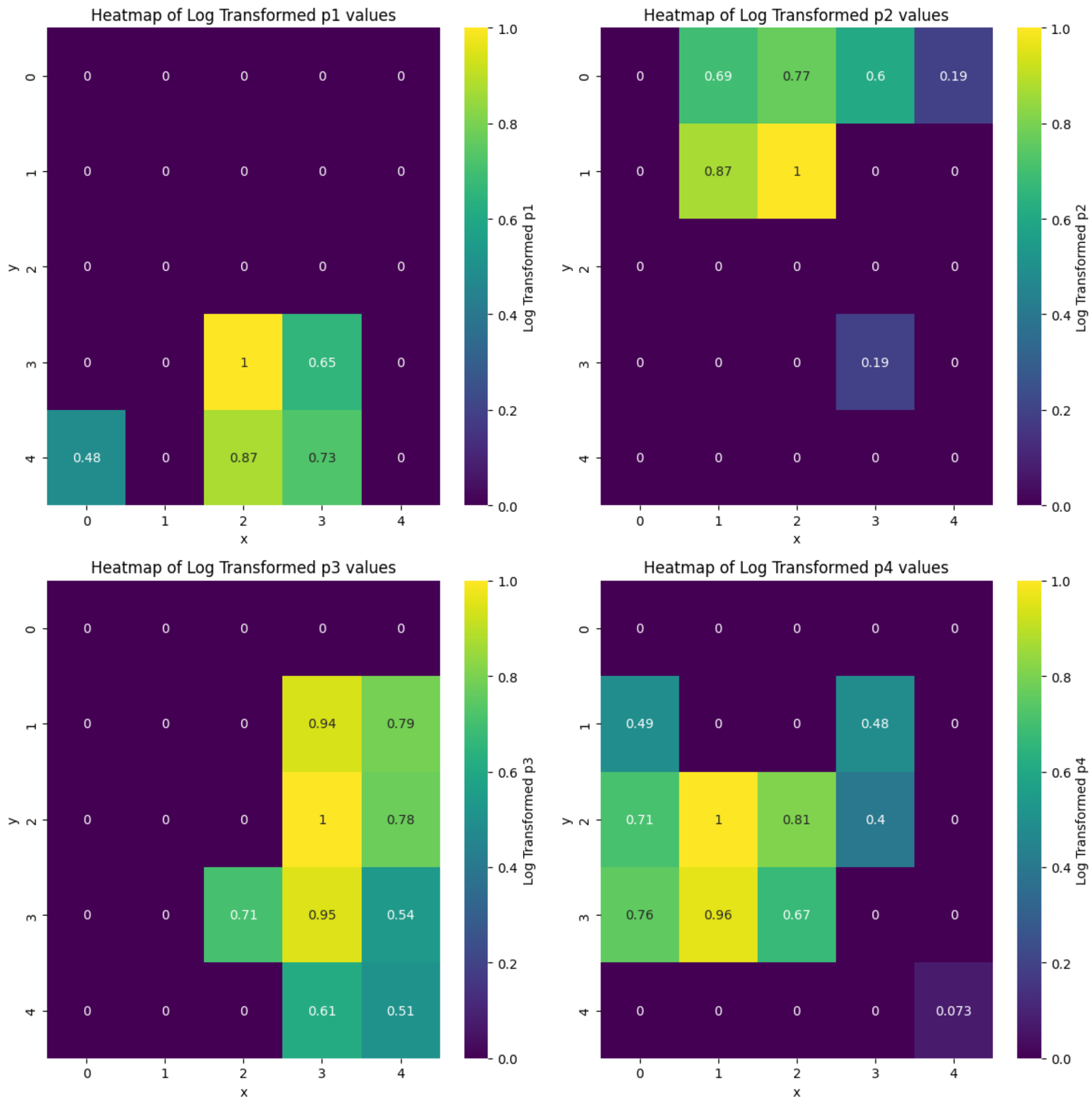
[50]:

	episode	duration_seconds
0	1	1.348670
1	2	0.434523
2	3	0.603118
3	4	0.970803
4	5	0.067901

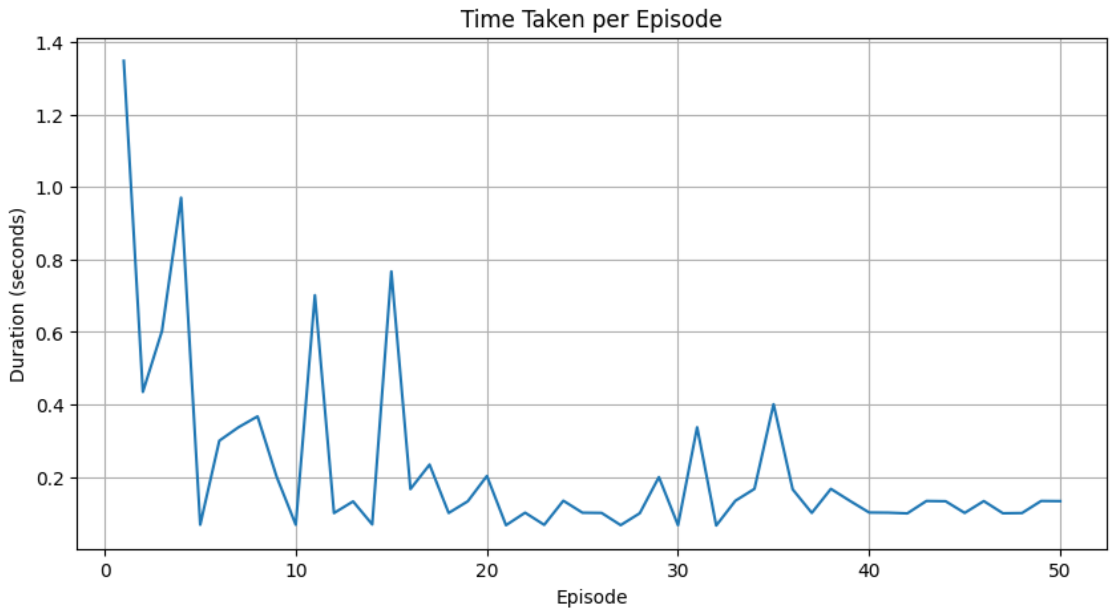
[51]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming q_table_df is already loaded and has columns 'x', 'y', 'p1', 'p2',
↪'p3', 'p4'

# List of columns to create heatmaps for
p_columns = ['p1', 'p2', 'p3', 'p4']

# Create a 2x2 grid for the heatmaps (2 rows, 2 columns)
fig, axes = plt.subplots(2, 2, figsize=(12, 12))
```

```python
# Loop through each column and create a heatmap
for idx, p_col in enumerate(p_columns):
    # Extract relevant columns using .copy() to avoid modifying a slice
    ptt = q_table_df[["x", "y", p_col]].copy()

    # Apply log transformation to the 'p' column to deal with skewed data
    # We add a small constant (e.g., 1e-5) to avoid issues with log(0)
    ptt.loc[:, p_col] = np.log(ptt[p_col] + 1e-5)

    # Normalize the transformed 'p' column to [0, 1] (optional, but might help)
    ptt.loc[:, p_col] = (ptt[p_col] - ptt[p_col].min()) / (ptt[p_col].max() -␣
 ↪ptt[p_col].min())

    # Get unique x and y values for grid creation
    x_unique = np.sort(ptt['x'].unique())
    y_unique = np.sort(ptt['y'].unique())

    # Create meshgrid for the heatmap
    grid_x, grid_y = np.meshgrid(x_unique, y_unique)

    # Create an empty matrix for the 'p' values on the grid
    grid_p = np.zeros_like(grid_x, dtype=float)

    # Assign normalized p values to the grid based on (x, y)
    for i in range(len(ptt)):
        x_idx = np.where(x_unique == ptt['x'].iloc[i])[0][0]
        y_idx = np.where(y_unique == ptt['y'].iloc[i])[0][0]
        grid_p[y_idx, x_idx] = ptt[p_col].iloc[i]

    # Determine the row and column for the current heatmap
    row = idx // 2  # Row index (0 or 1)
    col = idx % 2   # Column index (0 or 1)

    # Plotting the heatmap with log-transformed values
    sns.heatmap(grid_p, xticklabels=x_unique, yticklabels=y_unique,␣
 ↪cmap='viridis', cbar_kws={'label': f'Log Transformed {p_col}'}, annot=True,␣
 ↪ax=axes[row, col])

    # Adding labels and title to each subplot
    axes[row, col].set_xlabel('x')
    axes[row, col].set_ylabel('y')
    axes[row, col].set_title(f'Heatmap of Log Transformed {p_col} values')

# Adjust layout for better spacing between subplots
plt.tight_layout()
```

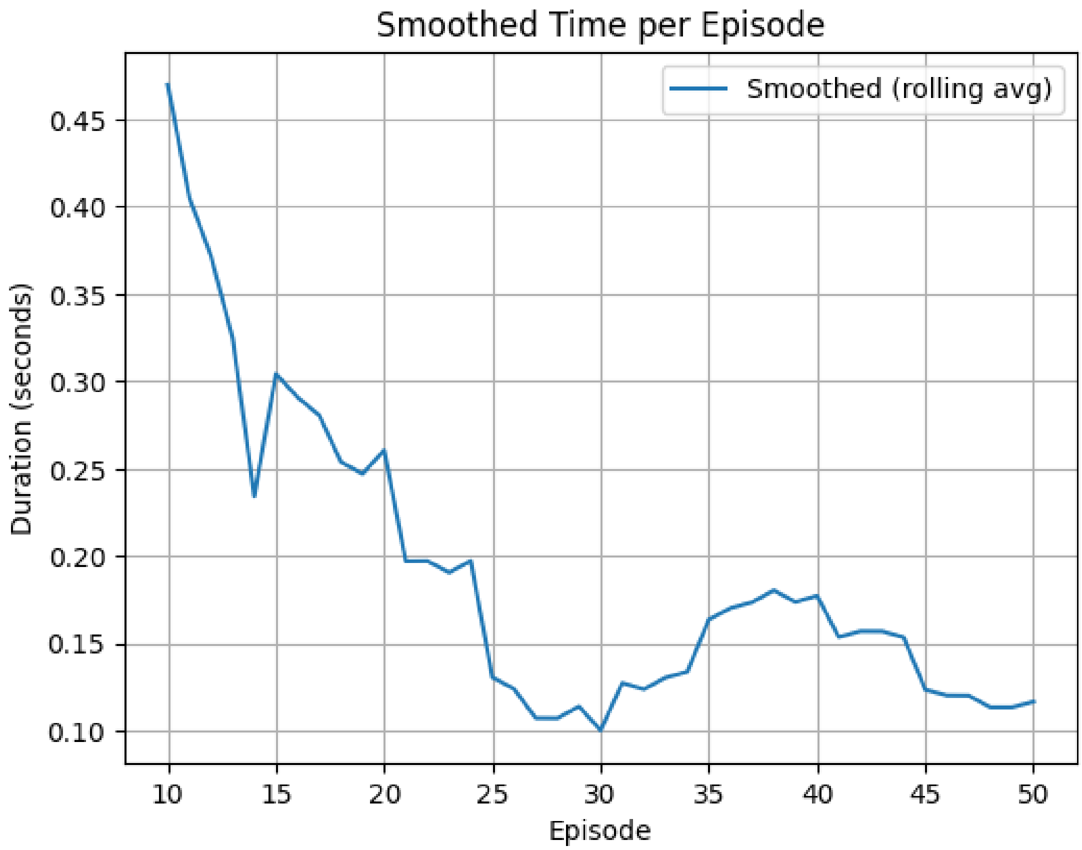
```python
# Show the plot
plt.show()
```



[52]:
```python
plt.figure(figsize=(10, 5))
plt.plot(episode_times_df["episode"], episode_times_df["duration_seconds"])
plt.xlabel("Episode")
plt.ylabel("Duration (seconds)")
plt.title("Time Taken per Episode")
plt.grid(True)
plt.show()
```



[53]:
```
episode_times_df["smooth"] = episode_times_df["duration_seconds"].
 ↪rolling(window=10).mean()

plt.plot(episode_times_df["episode"], episode_times_df["smooth"],␣
 ↪label="Smoothed (rolling avg)")
plt.xlabel("Episode")
plt.ylabel("Duration (seconds)")
plt.title("Smoothed Time per Episode")
plt.legend()
plt.grid(True)
plt.show()
```


Smoothed Time per Episode
Smoothed (rolling avg)
Duration (seconds)
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
10
15
20
25
30
35
40
45
50
Episode

[ ]: