(1)   LaTeX

```
\textbf{Thompson Sampling Algorithm (Gaussian Rewards)}

\begin{align*}
& \text{Initialize: } \hat{\mu}_a = 0, \; n_a = 0 \quad \forall a \in \{1, \dots, k\} \\
& \text{For each time step } t = 1 \text{ to } T: \\
& \quad \theta_a \sim \mathcal{N}(\hat{\mu}_a, 1 / (n_a + 1)) \quad \forall a \\
& \quad a_t = \arg\max_a \theta_a \\
& \quad \text{Receive reward } r_t \sim \mathcal{N}(\mu_{a_t}, 1) \\
& \quad n_{a_t} \leftarrow n_{a_t} + 1 \\
& \quad \hat{\mu}_{a_t} \leftarrow \hat{\mu}_{a_t} + \frac{1}{n_{a_t}}(r_t - \hat{\mu}_{a_t})
\end{align*}
```

(2) ChatGPT

       Gaussian reward   Thompson Sampling   arm

(3) Python

```python
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)
k = 10
steps = 10000
true_rewards = np.random.normal(0, 1, k)

mu_hat = np.zeros(k)
N = np.zeros(k)
rewards = np.zeros(steps)

for t in range(steps):
    samples = np.random.normal(mu_hat, 1 / (np.sqrt(N + 1)))
    a = np.argmax(samples)
    reward = np.random.normal(true_rewards[a], 1)
    N[a] += 1
    mu_hat[a] += (reward - mu_hat[a]) / N[a]
    rewards[t] = reward

cumulative_average = np.cumsum(rewards) / (np.arange(steps) + 1)

plt.figure(figsize=(10, 6))
plt.plot(cumulative_average, label="Thompson Sampling")
plt.xlabel("Steps")
plt.ylabel("Average Reward")
plt.title("Thompson Sampling Algorithm Performance")
plt.legend()
plt.grid(True)
plt.savefig("thompson_result.png", dpi=300)
plt.show()
```
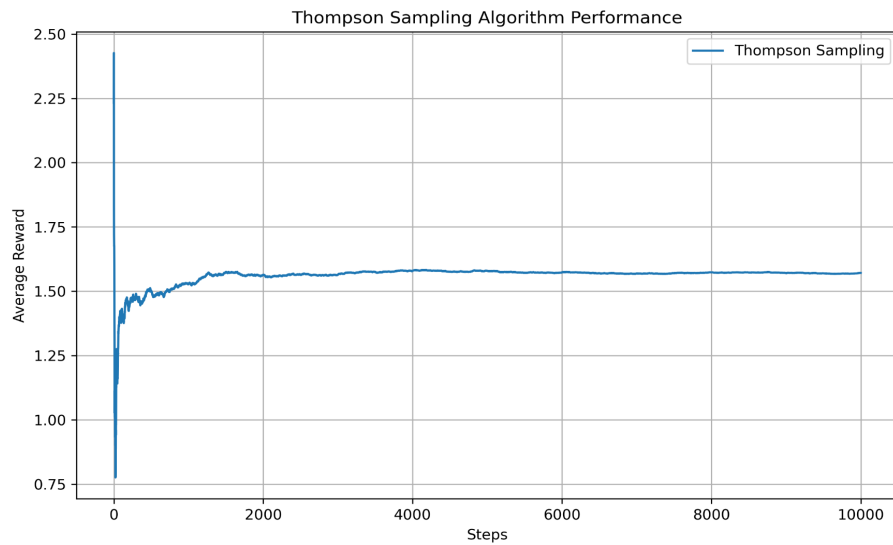
(3-1)

Thompson Sampling Algorithm Performance

(4)

O(T * k)

O(k + T)

-

-                 epsilon   tau

-