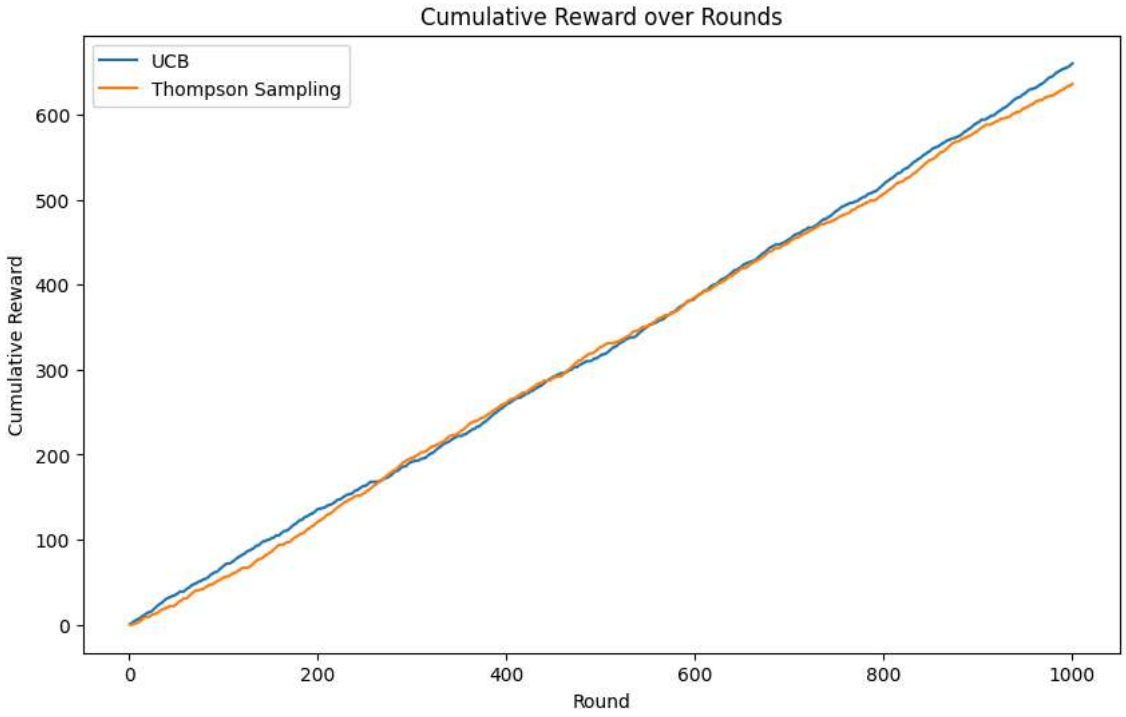
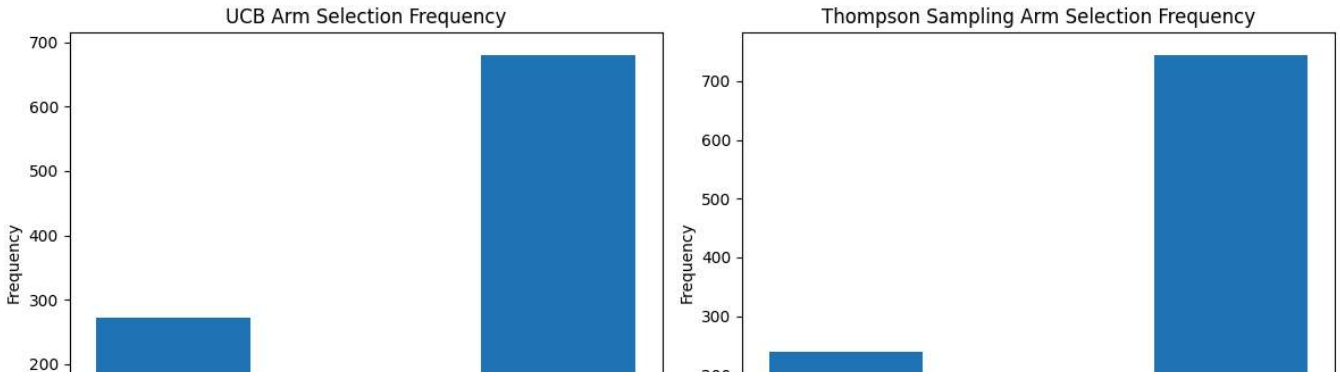
```
import random
def pull_arm(arm_index, true_probabilities):
  if random.random() <= true_probabilities[arm_index]:
    return 1
  else:
    return 0

true_probabilities = [0.6, 0.4, 0.7]


import numpy as np

def ucb_algorithm(num_rounds, true_probabilities):
  num_arms = len(true_probabilities)
  n_pulls = [0] * num_arms
  sum_rewards = [0] * num_arms
  arm_selections = []
  cumulative_rewards = []
  total_reward = 0

  # Initial pulls
  for arm in range(num_arms):
    reward = pull_arm(arm, true_probabilities)
    n_pulls[arm] += 1
    sum_rewards[arm] += reward
    arm_selections.append(arm)
    total_reward += reward
    cumulative_rewards.append(total_reward)

  # UCB logic for subsequent rounds
  for t in range(num_arms, num_rounds):
    ucb_values = []
    for arm in range(num_arms):
      if n_pulls[arm] > 0:
        average_reward = sum_rewards[arm] / n_pulls[arm]
        exploration_term = np.sqrt(2 * np.log(t + 1) / n_pulls[arm])
        ucb_values.append(average_reward + exploration_term)
      else:
        ucb_values.append(float('inf')) # Explore unpulled arms first

    selected_arm = np.argmax(ucb_values)
    reward = pull_arm(selected_arm, true_probabilities)

    n_pulls[selected_arm] += 1
    sum_rewards[selected_arm] += reward
    arm_selections.append(selected_arm)
    total_reward += reward
    cumulative_rewards.append(total_reward)

  return arm_selections, cumulative_rewards



import random
import numpy as np
from scipy.stats import beta

def thompson_sampling_algorithm(num_rounds, true_probabilities):
  num_arms = len(true_probabilities)
  # Initialize with a prior of 1 success and 1 failure for each arm
  successes = [1] * num_arms
  failures = [1] * num_arms

  arm_selections = []
  cumulative_rewards = []
  total_reward = 0

  for t in range(num_rounds):
    # Draw samples from Beta distributions for each arm
    beta_samples = [beta.rvs(successes[i], failures[i]) for i in range(num_arms)]

    # Select the arm with the highest sample
    selected_arm = np.argmax(beta_samples)

    # Simulate pulling the selected arm
    reward = pull_arm(selected_arm, true_probabilities)

    # Update success or failure counts
    if reward == 1:
      successes[selected_arm] += 1
    else:
```

```
          failures[selected_arm] += 1

        # Record arm selection and cumulative reward
        arm_selections.append(selected_arm)
        total_reward += reward
        cumulative_rewards.append(total_reward)

      return arm_selections, cumulative_rewards


num_rounds = 1000

ucb_arm_selections, ucb_cumulative_rewards = ucb_algorithm(num_rounds, true_probabilities)
ts_arm_selections, ts_cumulative_rewards = thompson_sampling_algorithm(num_rounds, true_probabilities)


import matplotlib.pyplot as plt
import numpy as np

# Plot cumulative reward
plt.figure(figsize=(10, 6))
plt.plot(range(1, num_rounds + 1), ucb_cumulative_rewards, label='UCB')
plt.plot(range(1, num_rounds + 1), ts_cumulative_rewards, label='Thompson Sampling')
plt.xlabel('Round')
plt.ylabel('Cumulative Reward')
plt.title('Cumulative Reward over Rounds')
plt.legend()
plt.show()

# Plot arm selection frequency
num_arms = len(true_probabilities)
ucb_arm_counts = [ucb_arm_selections.count(i) for i in range(num_arms)]
ts_arm_counts = [ts_arm_selections.count(i) for i in range(num_arms)]

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

axes[0].bar(range(num_arms), ucb_arm_counts)
axes[0].set_xticks(range(num_arms))
axes[0].set_xlabel('Arm Index')
axes[0].set_ylabel('Frequency')
axes[0].set_title('UCB Arm Selection Frequency')

axes[1].bar(range(num_arms), ts_arm_counts)
axes[1].set_xticks(range(num_arms))
axes[1].set_xlabel('Arm Index')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Thompson Sampling Arm Selection Frequency')

plt.tight_layout()
plt.show()
```


Cumulative Reward over Rounds
UCB
Thompson Sampling
Cumulative Reward
Round
0
100
200
300
400
500
600
0
200
400
600
800
1000


UCB Arm Selection Frequency
Frequency
200
300
400
500
600
700


Thompson Sampling Arm Selection Frequency
Frequency
300
400
500
600
700