Laboratorio 1

Daniel González Carrillo 20293
Sebastian Aristondo 20880
Pablo Daniel González 20362

Task 1

1. ¿Qué pasa si algunas acciones tienen probabilidades de cero?

R: En general, en cualquier algoritmo de bandido, si una opción tiene probabilidad 0 de ser elegida, esa opción nunca se explorará ni se explotará, lo que puede llevar a que el algoritmo no aprenda una estrategia óptima si esa opción era en realidad la mejor. Esto resalta la importancia de garantizar que todas las opciones tengan alguna probabilidad no nula de ser seleccionadas, al menos durante la fase de exploración, para asegurar que el algoritmo pueda descubrir la verdadera distribución de recompensas de cada opción.

2. ¿Qué pasa si la póliza es determinística?
   a. π1(a) = 1 para algún a

R: Si la póliza (o política) es determinística y se define como $\pi(a)$=1 para alguna acción a, significa que el agente siempre elegirá esa acción a en particular.La política determinística elimina completamente la exploración. El agente nunca probará ninguna otra acción aparte de la acción a. Si la acción a a no es la mejor acción en términos de recompensa esperada, el agente terminará obteniendo una recompensa subóptima en el largo plazo.

3. Investigue y defina a qué se le conoce como cada uno de los siguientes términos, asegúrese de definir en qué consiste cada una de estas variaciones y cómo difieren de los k-armed bandits

a. Contextual bandits

Los bandidos multi armados contextuales (Contextual Multi-Armed Bandits) son una extensión del problema clásico de los bandidos multi armados (Multi-Armed Bandit). Esta extensión incorpora la idea de que hay contextos o características adicionales que pueden influir en la decisión de cuál "brazo" (acción) elegir.

Contexto (Xt): Antes de cada ronda t, el jugador observa un contexto Xt, que puede ser un vector de características. Este contexto proporciona información adicional que puede ayudar a predecir las recompensas de los diferentes brazos.

Brazos (A): Hay un conjunto finito de brazos A={1,2,...,K}. En cada ronda, el jugador selecciona uno de estos brazos.

Recompensa ($Rt(a)$): Cada brazo aa tiene una recompensa asociada $Rt(a)$, que puede depender del contexto Xt. La recompensa es generalmente estocástica y tiene una distribución desconocida.

Estrategia de Decisión ($\pi(Xt)$): La estrategia del jugador es una función que mapea el contexto observado Xt a una elección de brazo at . El objetivo es diseñar una estrategia que maximice la recompensa acumulada a lo largo del tiempo.

b. Dueling bandits

Los dueling bandits (bandidos en duelo) son una variante del problema de los bandidos multi-armados (Multi-Armed Bandit) que aborda situaciones donde las comparaciones entre pares de acciones son más naturales o fáciles de obtener que las recompensas absolutas. En lugar de recibir recompensas numéricas directas, el jugador compara dos acciones (brazos) y observa cuál es mejor según un criterio dado. Esta configuración es útil en aplicaciones donde es más fácil evaluar las preferencias relativas que medir recompensas absolutas.

Brazos (A): Hay un conjunto finito de brazos A={1,2,...,K}. En cada ronda, el jugador selecciona dos brazos para comparar.

Preferencias ($P(a,b)$): Las preferencias entre dos brazos a y b están representadas por la probabilidad $P(a,b)$ de que el brazo a sea preferido sobre el brazo b. Estas preferencias son generalmente estocásticas y desconocidas.

Estrategia de Decisión (π): La estrategia del jugador es una función que mapea la información recolectada hasta el momento a una elección de par de brazos para comparar.

c. Combination bandits

Los Combination Bandits (bandidos combinatorios) son una extensión del problema de los bandidos multi-armados (Multi-Armed Bandit) que se aplican en situaciones donde se deben seleccionar combinaciones de acciones en lugar de una sola acción. Este tipo de problema surge en contextos donde las decisiones no son independientes y se deben tomar en conjunto para maximizar la recompensa.

Elementos (A): Hay un conjunto finito de elementos A={1,2,...,N} que pueden ser combinados. En cada ronda, el jugador selecciona una combinación de estos elementos. Conjunto de Decisiones (S): Un conjunto $S \subseteq A$ representa una combinación de elementos. Las posibles combinaciones están definidas por un conjunto de restricciones que especifican cuáles combinaciones son válidas.

Recompensa ($R(S)$): Cada combinación S tiene una recompensa asociada $R(S)$. La recompensa es generalmente estocástica y desconocida.

Estrategia de Decisión (π): La estrategia del jugador es una función que mapea la información recolectada hasta el momento a una elección de combinación S.

Task 2

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
np.random.seed(1234)
```

In [ ]:
```
class Bandit:
    def __init__(self, n_arms):
        self.p_true = np.random.uniform(0, 1, n_arms)
        self.n_arms = n_arms

    def pull(self, arm):
        random_value = np.random.uniform(0, 1)
        if random_value < self.p_true[arm]:
            return 1
        else:
            return 0

    def get_arm_count(self):
        return self.n_arms

    def get_probabilities(self):
        return self.p_true
```

In [ ]:
```
class Agent:
    def __init__(self, bandit, epsilon):
        self.bandit = bandit
        self.epsilon = epsilon
        self.arm_counts = np.zeros(bandit.get_arm_count())
        self.rewards = np.zeros(bandit.get_arm_count())

    def update_estimates(self, arm, reward):
        self.arm_counts[arm] += 1
        self.rewards[arm] += (reward - self.rewards[arm]) / self.arm_counts[arm]

    def select_arm(self):
        probability = np.random.uniform(0, 1)
        if probability < self.epsilon:
            return np.random.randint(0, self.bandit.get_arm_count())
        else:
            return np.argmax(self.rewards)

    def get_rewards(self):
        return self.rewards
```

In [ ]:
```
def plot_data(data, title):
    plt.plot(data)
```

```
    plt.title(title)
    plt.show()
```

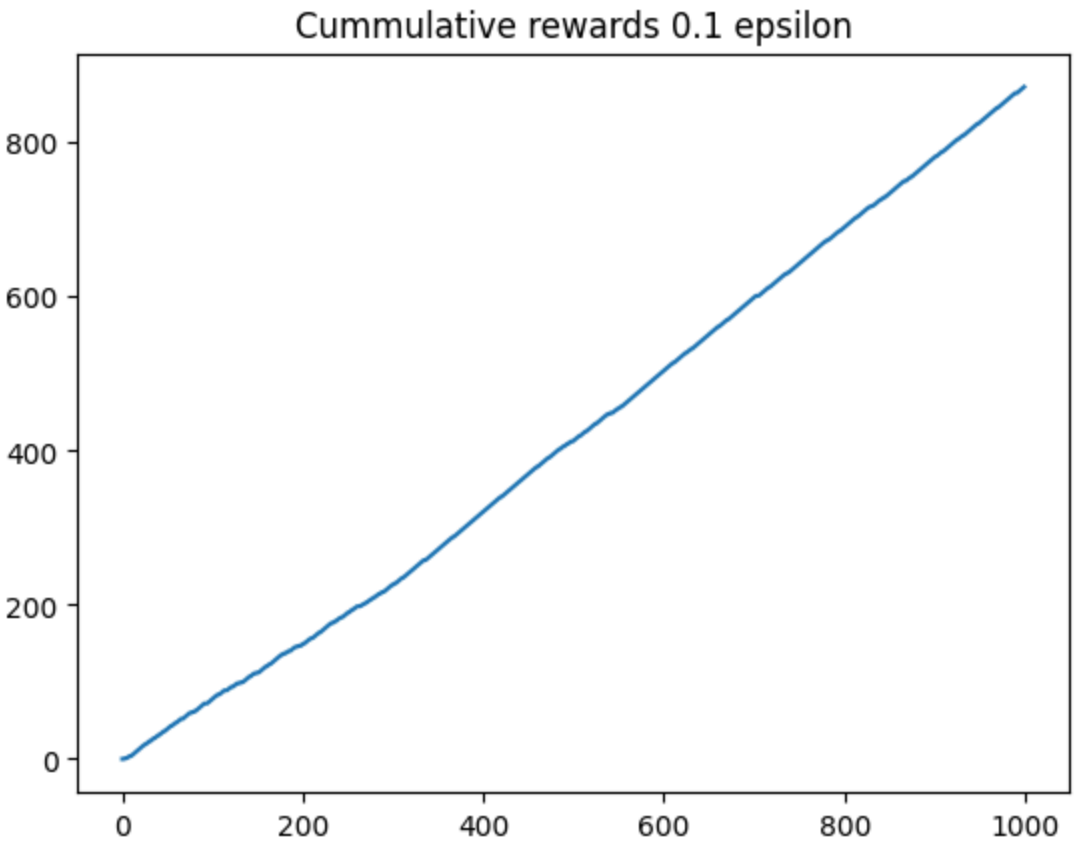
In [ ]:
```
def plot_probabilities(agent, bandit, title):
    plt.plot(agent.get_rewards(), label='Agent', marker='o')
    plt.plot(bandit.get_probabilities(), label='Bandit', marker='x')
    plt.title(title)
    plt.legend()
    plt.show()
```

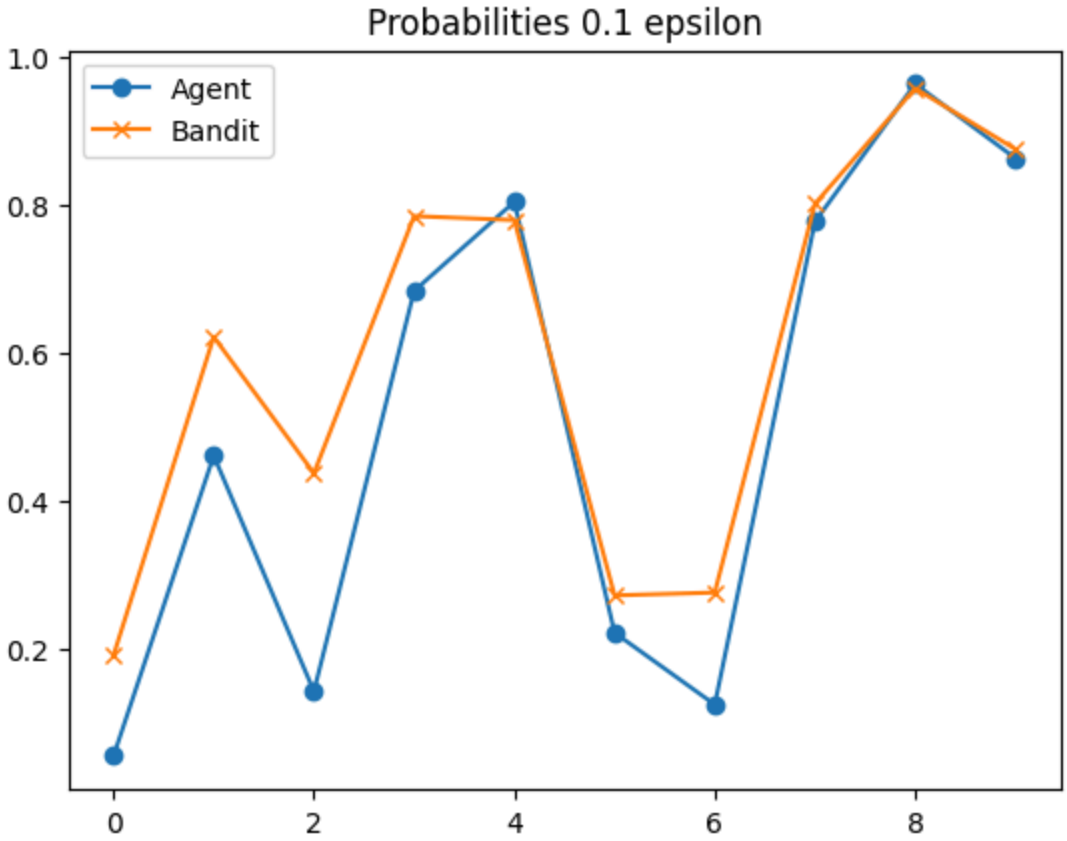
In [ ]:
```
def run_simulation(arms, epsilon, iter, title_rewards, title_probabilities):
    bandit = Bandit(arms)
    agent = Agent(bandit, epsilon)
    rewards = []
    cummulative_rewards = []
    for _ in range(iter):
        arm = agent.select_arm()
        reward = bandit.pull(arm)
        agent.update_estimates(arm, reward)
        rewards.append(reward)
        cummulative_rewards.append(sum(rewards))

    print('Total accumulated reward:', sum(rewards))
    plot_data(cummulative_rewards, title_rewards)
    plot_probabilities(agent, bandit, title_probabilities)
```

In [ ]:
```
run_simulation(10, 0.1, 1000, 'Cummulative rewards 0.1 epsilon', 'Probabilities 0.1
```

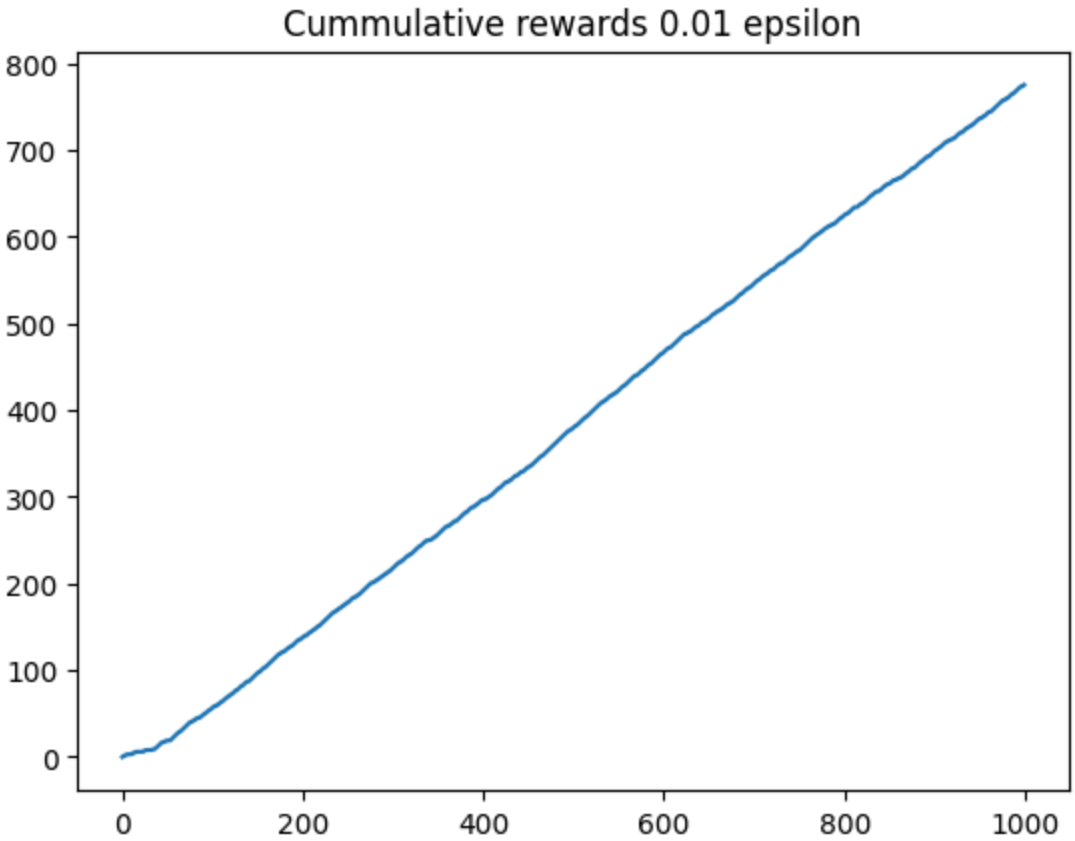
```
Total accumulated reward: 872
```

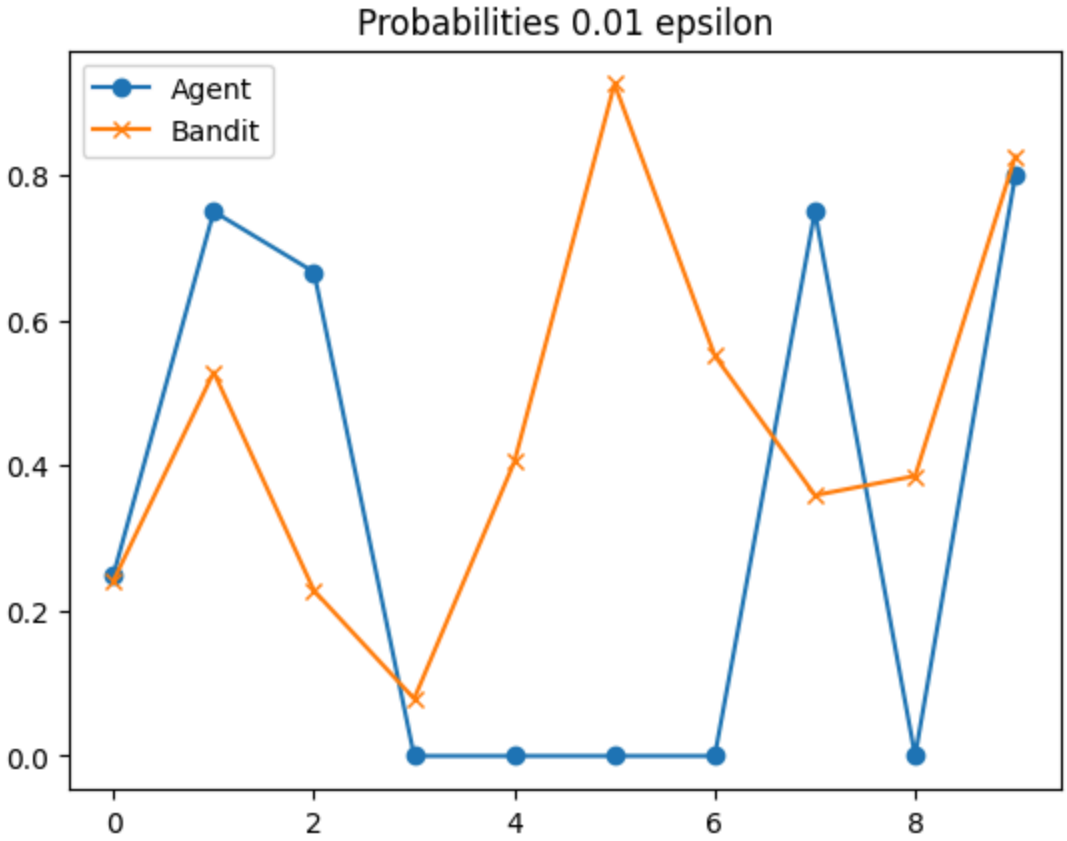




In [ ]:
```
run_simulation(10, 0.01, 1000, 'Cummulative rewards 0.01 epsilon', 'Probabilities 0
```

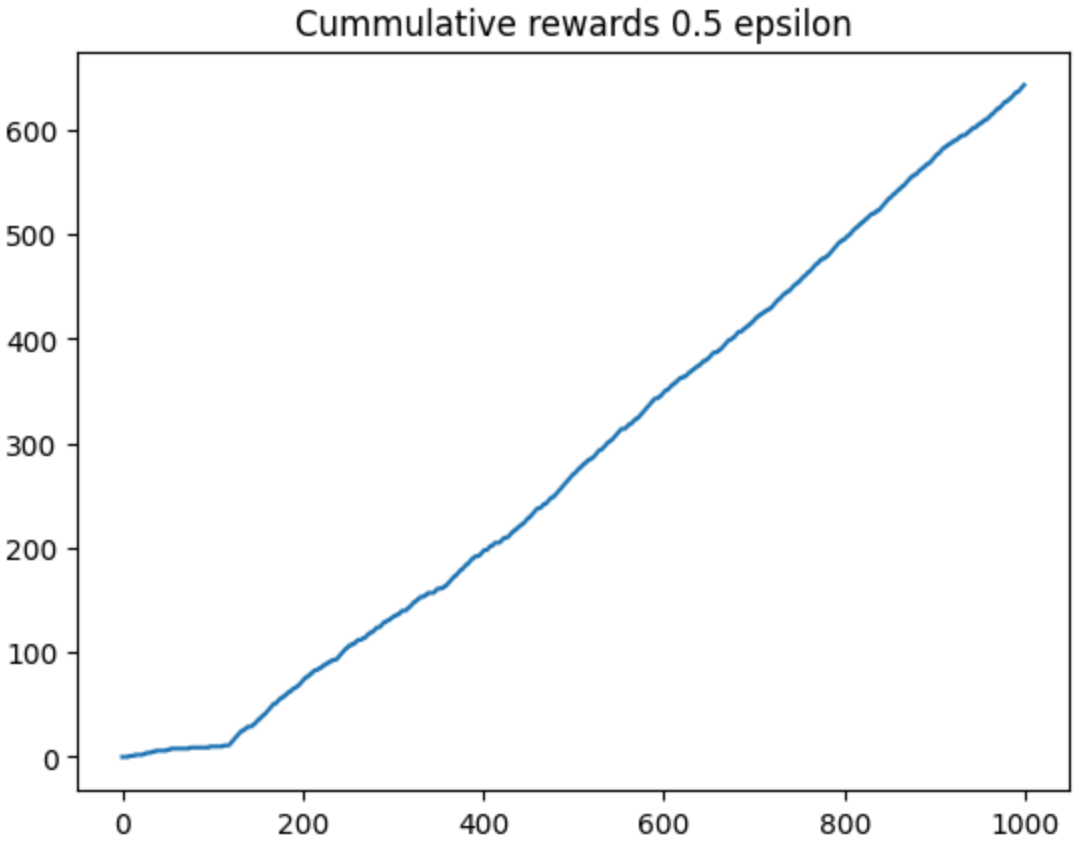
```
Total accumulated reward: 775
```

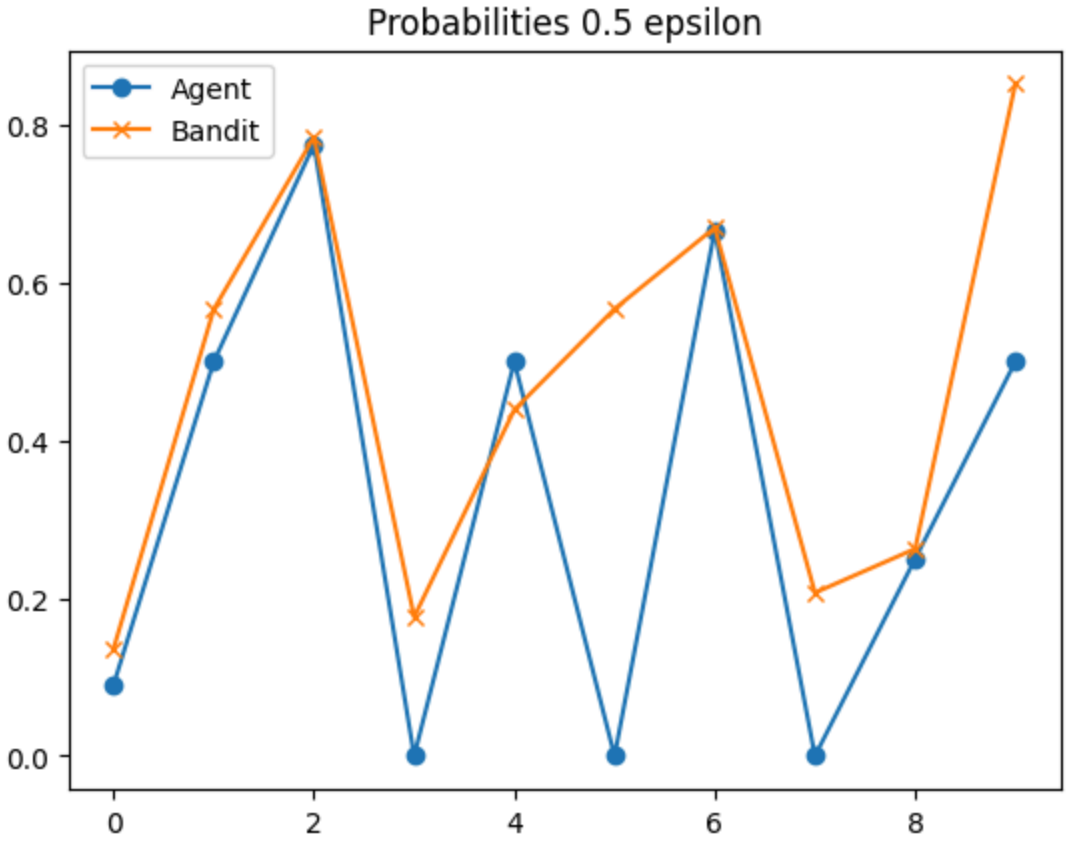




In [ ]:
```
run_simulation(10, 0.05, 1000, 'Cummulative rewards 0.5 epsilon', 'Probabilities 0.
```

```
Total accumulated reward: 643
```




Probabilities 0.5 epsilon
Agent
Bandit
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8