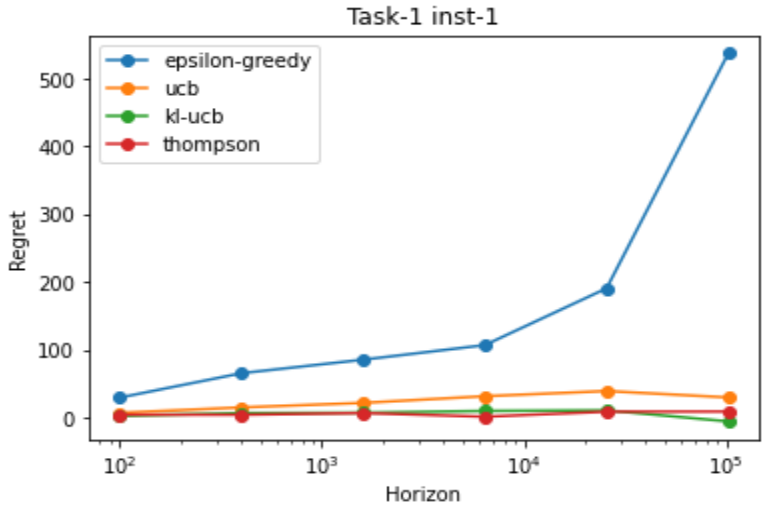
CS 747

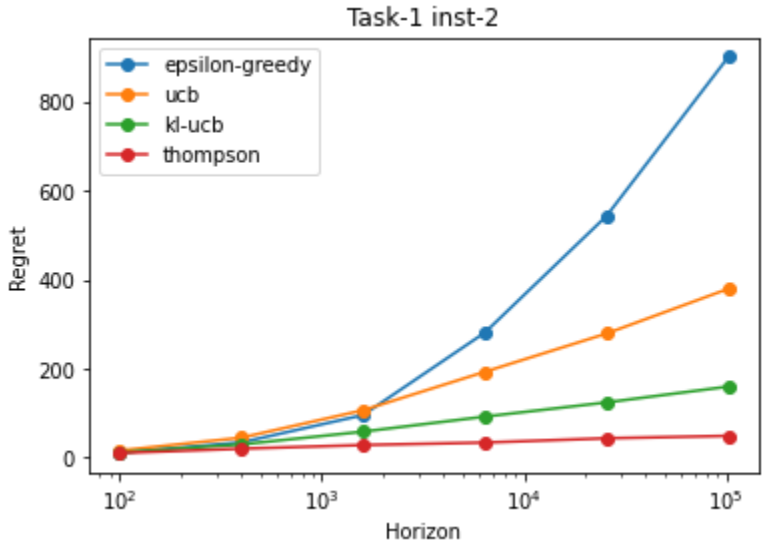
Programming Assignment 1

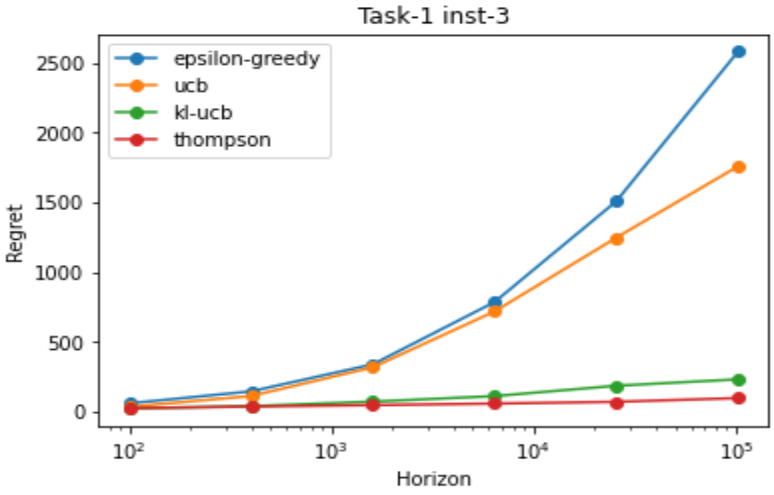
Rushikesh J. Metkar **19D070034**

CONTENTS :

TASK 1


Task-1 inst-1
epsilon-greedy
ucb
kl-ucb
thompson
Regret
Horizon
0
100
200
300
400
500
10^2
10^3
10^4
10^5


Task-1 inst-2
epsilon-greedy
ucb
kl-ucb
thompson
Regret
Horizon
0
200
400
600
800
10^2
10^3
10^4
10^5


Task-1 inst-3
epsilon-greedy
ucb
kl-ucb
thompson
Regret
Horizon
0
500
1000
1500
2000
2500
10^2
10^3
10^4
10^5

I have implemented the following sampling algorithms:
(1) epsilon-greedy, (2) UCB, (3) KL-UCB, and (4) Thompson Sampling

I've assumed that each arm is pulled at least once.

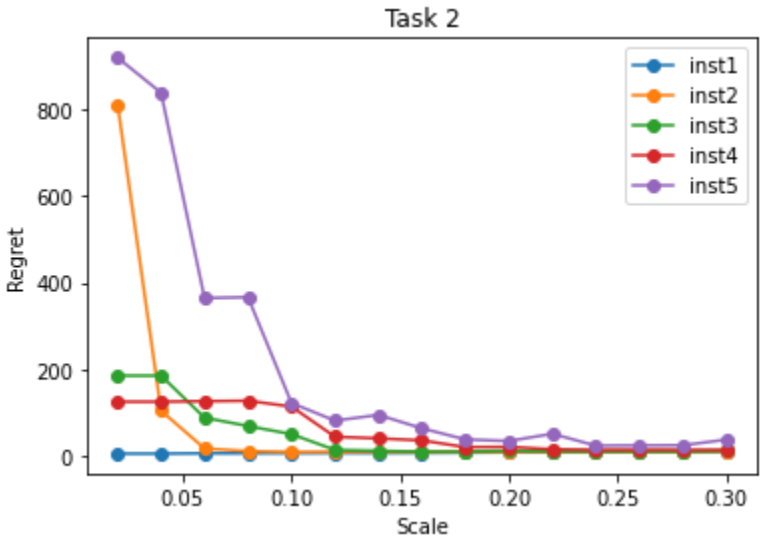
In Epsilon-greedy algorithm, I'm sampling an arm uniformly at random; with probability epsilon and sampling an arm with the highest empirical mean with probability 1 - epsilon
I know the most optimal arm and by using it's true mean (p_star) I've computed regret as REG = [hz*p_star - REW (cumulative reward)]
The Epsilon-greedy algorithm gives the highest regret for all instances.

In the UCB algorithm, I'm sampling an arm which has the highest ucb value.
ucb_arm = empirical_mean_arm + exploration_bonus where
exploration_bonus is defined as the square root of **2** × ln(total_pulls) / pulls_arm
REG is computed similarly to the first case.
The UCB algorithm gives the second highest regret for all instances.

In the KL-UCB algorithm, I'm sampling an arm which has the highest kl-ucb value.
ucb-klt(a)= max{q ∈ [pˆ t a , 1] such that ut(a)KL(pˆ t(a) , q) ≤ ln(t) + c ln(ln(t))},where c≥3
I've made two functions KL and solve_q where the first function finds the KL divergence between two inputs and the second function gives the value of q that follows KL inequality.
REG is computed similarly to the first case.
Regret is very low as compared to the first two cases.

In the Thompson sampling algorithm, for every arm a, I'm drawing a sample xt(a) ~ Beta(st(a) + 1, ft(a) + 1) where st(a) are successes and ft(a) are failures and choosing the arm with maximal xt(a).
In this algorithm I've pulled each arm for 1 time in the start and then selected an arm depending on the Beta distribution values.
REG is computed similarly to the first case.
Regret is very low as compared to the first two cases.

TASK 2

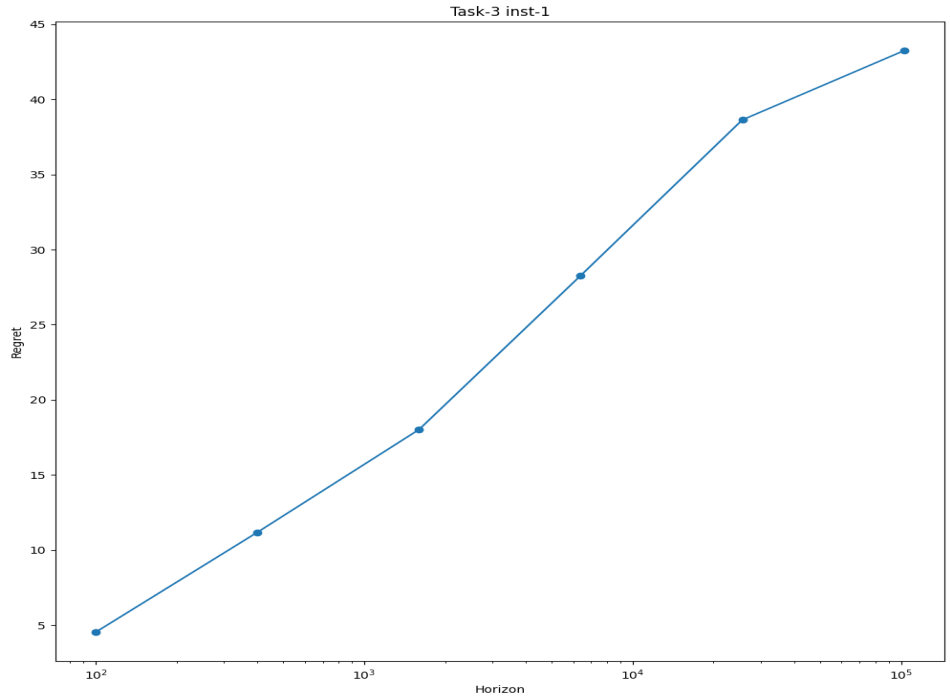


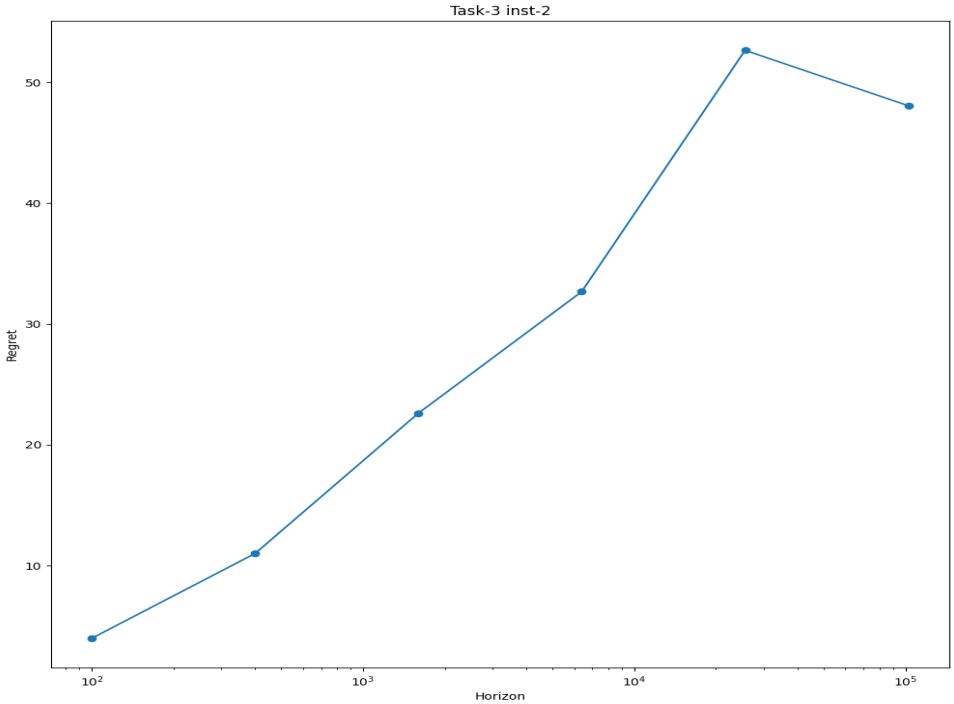
The value of c that gave the lowest regret for each of the five instances is **0.26**

For each instance as we increase the scale, regret value decreases.
Exploration_bonus is defined as the square root of **c** × ln(total_pulls) / pulls_arm.
Exploration_bonus increases with c and thus ucb for each arm will increase accordingly.

If scale is too low then empirical mean factor will dominate and if scale is too high then Exploration_bonus factor dominates.
Thus, choosing a proper scale value will help us to choose the optimum arm.

TASK 3


Task-3 inst-1
Regret
Horizon
45
40
35
30
25
20
15
10
5
10^2
10^3
10^4
10^5


Task-3 inst-2
Regret
Horizon
50
40
30
20
10
10^2
10^3
10^4
10^5

I have used Thompson sampling to implement alg-t3.

For both instances the average regret is less than 50 for a horizon upto 10^5.

For every arm a, I'm drawing a sample xt(a) ~ Beta(st(a) + 1, ft(a) + 1) where st(a) are successes and ft(a) are failures and choosing the arm with maximal xt(a).

In this algorithm I've pulled each arm for 1 time in the start and then selected an arm depending on the Beta distribution values.

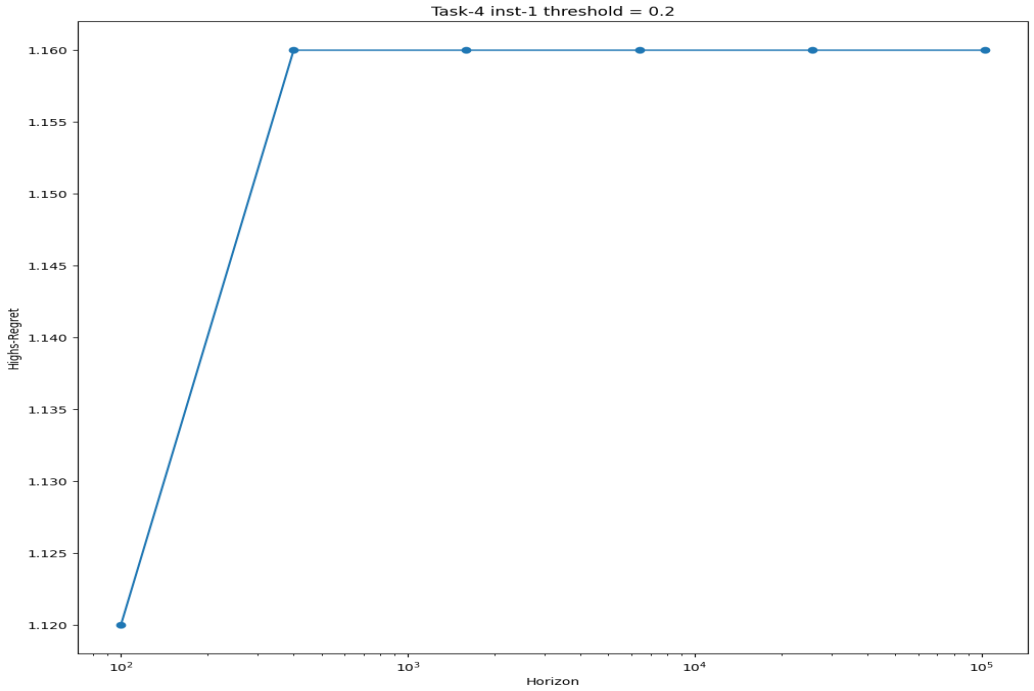
Using np.random.choice() I'm generating the reward for each pull.

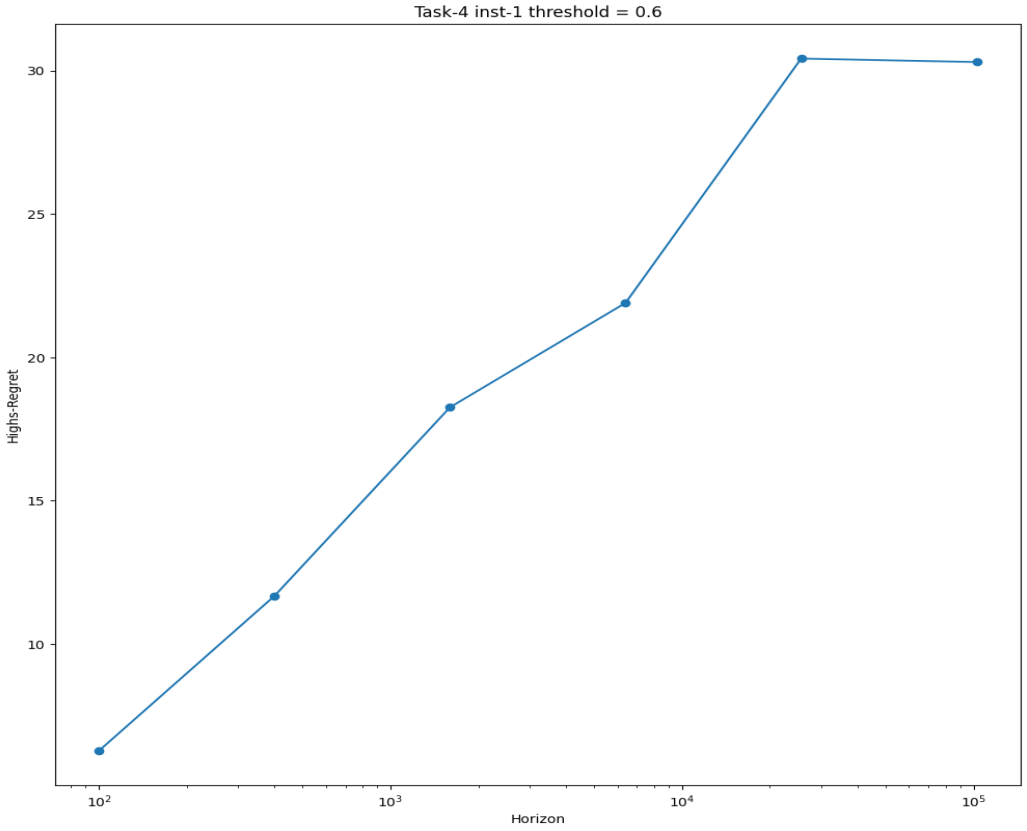
I've calculated regret as follows

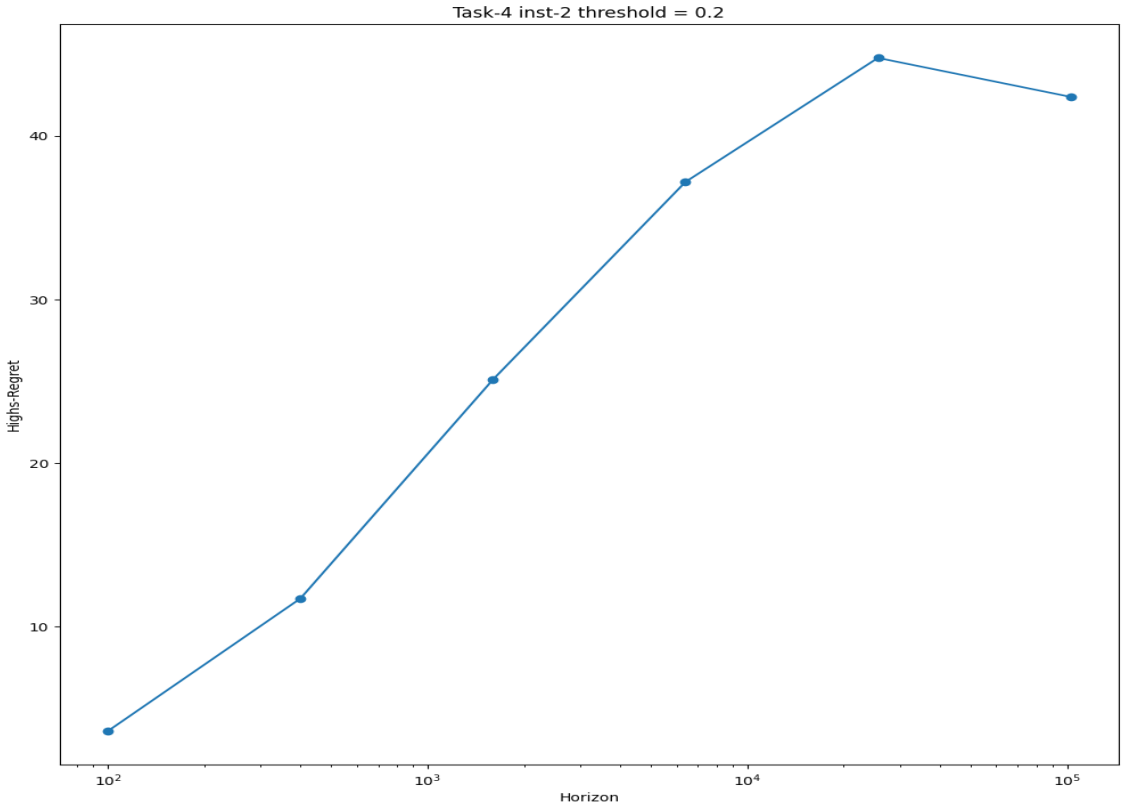
1. Getting the maximum of expected rewards obtained from each arm
2. Multiplying it with horizon to get max_reward = horizon x max(E(reward))
3. Calculating REG as REG = max_reward - REW (cumulative reward)

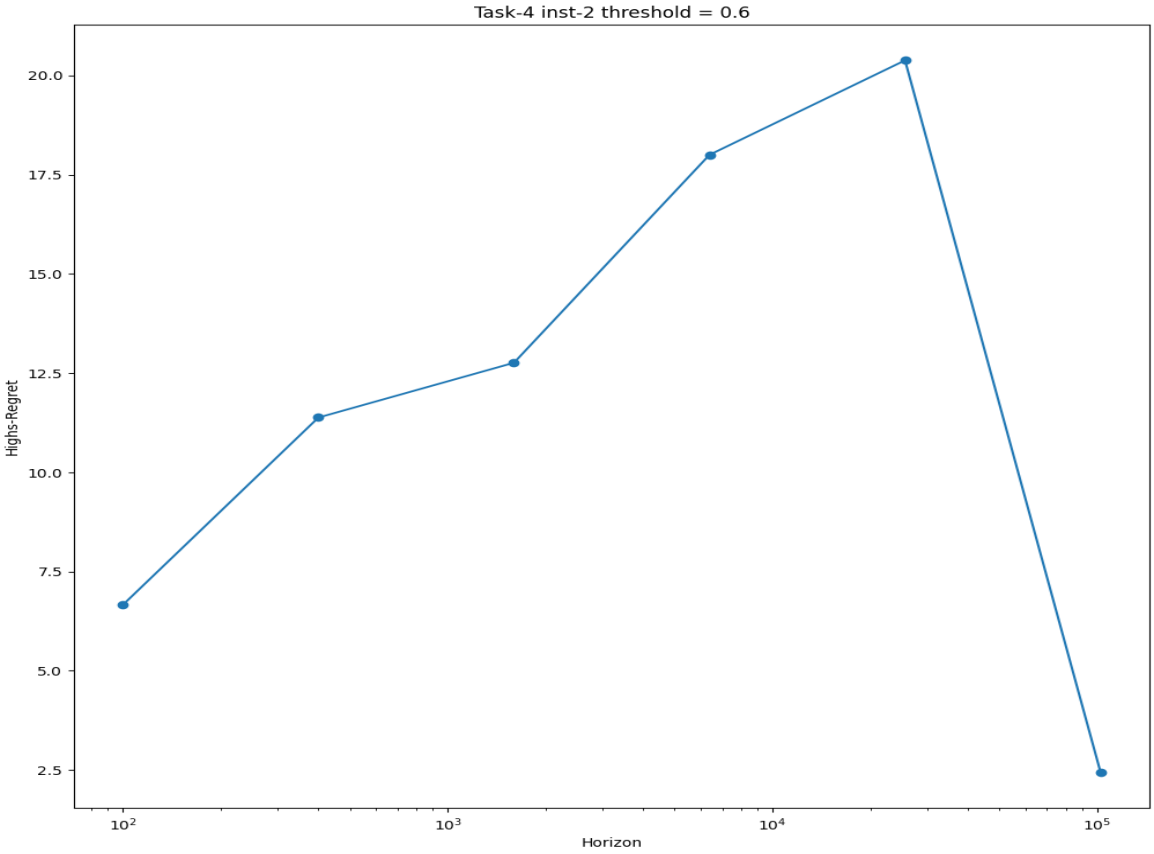
With the help of Thompson sampling algorithm I'm getting very low regret.

TASK 4


Task-4 inst-1 threshold = 0.2
Highs-Regret
1.160
1.155
1.150
1.145
1.140
1.135
1.130
1.125
1.120
10^2
10^3
10^4
10^5
Horizon


Task-4 inst-1 threshold = 0.6
Highs-Regret
30
25
20
15
10
10^2
10^3
10^4
10^5
Horizon


Task-4 inst-2 threshold = 0.2
Highs-Regret
40
30
20
10
10^2
10^3
10^4
10^5
Horizon


Task-4 inst-2 threshold = 0.6
Highs-Regret
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
10^2
10^3
10^4
10^5
Horizon

I have used Thompson sampling here as well to implement alg-t4.

In this algorithm I've pulled each arm for 1 time in the start and then selected an arm depending on the Beta distribution values.

Using np.random.choice() I'm generating the reward for each pull.

If the reward I get is greater than the threshold then it is considered as '1' else '0'.

I've calculated regret as follows

1. E(reward)(arm_x) = p_m + ….. + p_n where p_m upto p_n are the probabilities of getting a reward greater than threshold .
2. Getting the maximum of expected rewards obtained from each arm
3. Multiplying it with horizon to get max_reward = horizon x max(E(reward))
4. Calculating regret as REG = max_reward - REW (cumulative reward)

We can see from the plots that for very large horizons we are getting very less regret.