CS 747: Programming Assignment 1

# REPORT

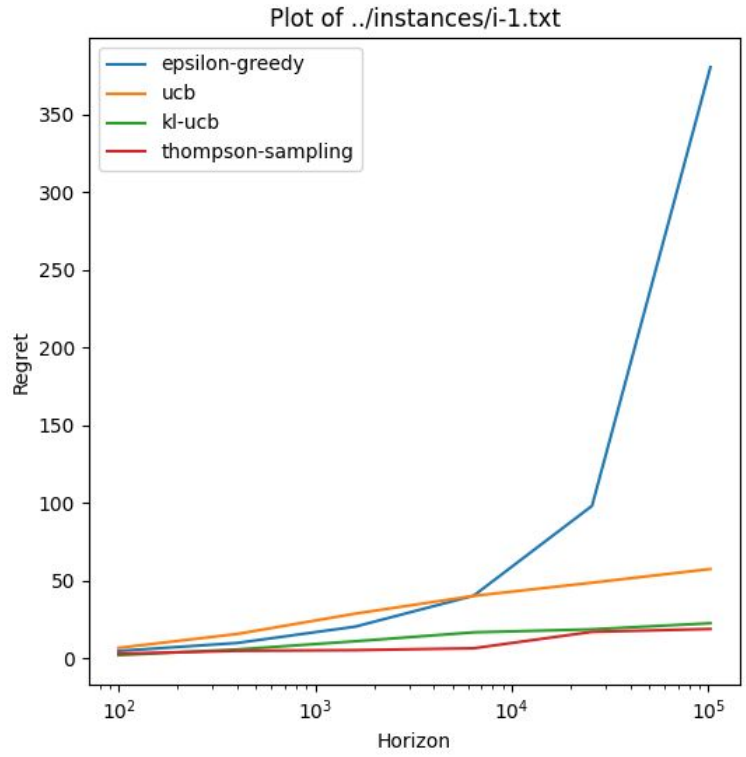
## Task 1

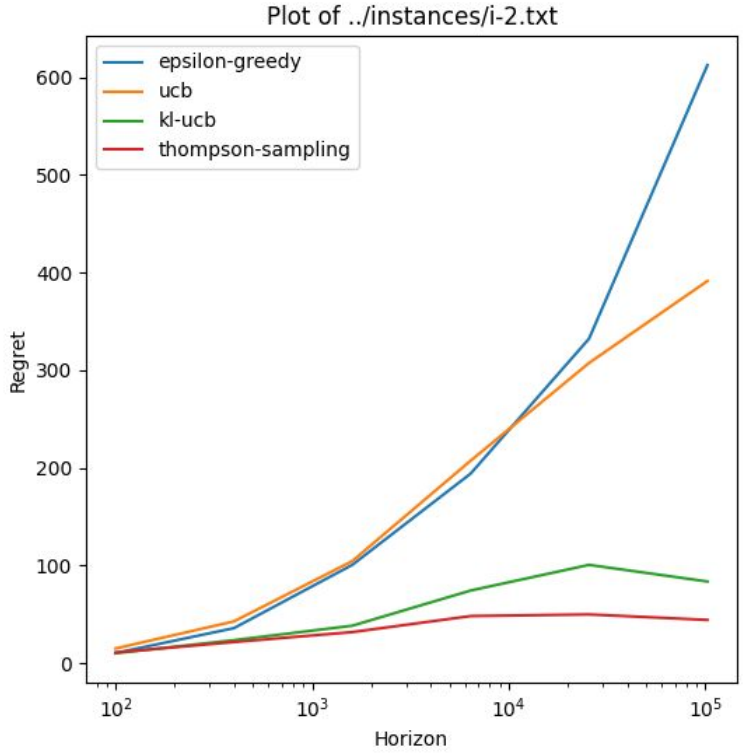
Please find the plots attached below.

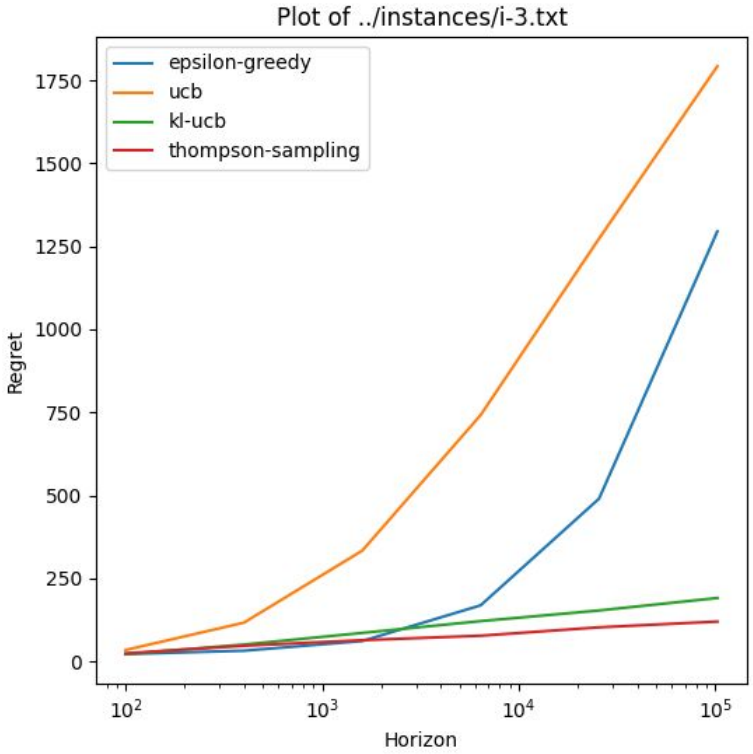
The implementation details are as follows.

1. Epsilon-greedy: An initial round-robin to prevent division by 0 issues. Then, with probability $\varepsilon$ pick an arm uniformly at random, and with probability $1-\varepsilon$ pick the arm with the highest empirical mean reward (splitting ties uniformly). Pull the arm picked and update its empirical mean.
2. UCB: An initial round-robin to prevent division by 0 issues. Thereafter at each time step compute the ucb value (as defined in the lectures) of each arm and pick the arm with the highest ucb value (splitting ties uniformly). Pull the arm picked and update its empirical mean.
3. KL-UCB: An initial round-robin to prevent division by 0 issues. Thereafter at each time step compute the kl-ucb value (computed by performing a binary search as suggested in class) of each arm and pick the arm with the highest kl-ucb value (splitting ties uniformly). Pull the arm picked and update its empirical mean.
4. Thompson: Maintain a belief PDF (a beta distribution) over true means for each arm. At every time step, draw a sample from the distribution and pick the arm with the maximum sampled value (splitting ties uniformly). Pull the arm picked and update its empirical mean.

Interpretations:
The graphs are scaled logarithmically in the x-axis, so the regret of epsilon-greedy, which is a linear, blows up with an increase in the horizon. It performs better for low values of the horizon but soon is outperformed by the others for the same reason. The rest all give logarithmic regret so roughly form straight lines in the plots. UCB has a greater slope than KL-UCB and Thompson because they are optimal wrt Lai and Robbins' bound while UCB is not. An increase in the number of arms can be visualized as reducing the horizon, thus epsilon-greedy outperforms UCB for most of instance-3, but if we were to generate data for higher horizon we would observe similar behavior to the graphs' of the other instances.


Plot of ../instances/i-1.txt
epsilon-greedy
ucb
kl-ucb
thompson-sampling
Regret
Horizon
0
50
100
150
200
250
300
350
$10^2$
$10^3$
$10^4$
$10^5$

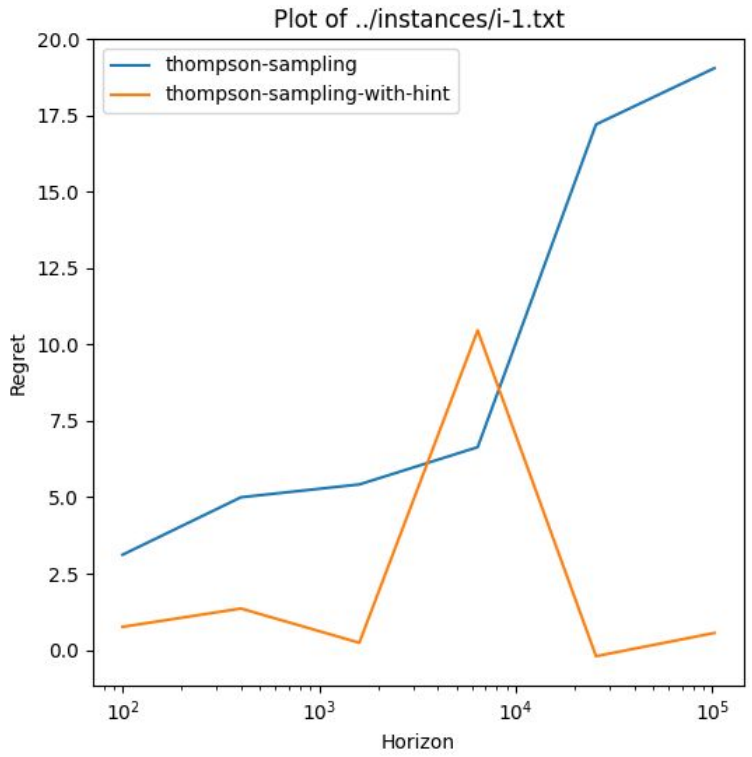

Plot of ../instances/i-2.txt
epsilon-greedy
ucb
kl-ucb
thompson-sampling
Regret
Horizon
0
100
200
300
400
500
600
$10^2$
$10^3$
$10^4$
$10^5$

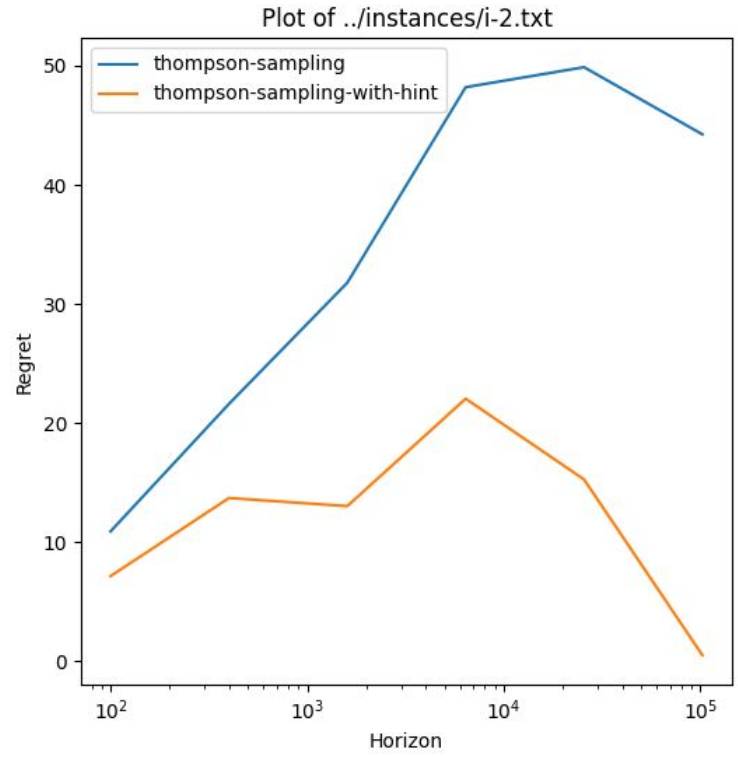

Plot of ../instances/i-3.txt
epsilon-greedy
ucb
kl-ucb
thompson-sampling
Regret
Horizon
0
250
500
750
1000
1250
1500
1750
$10^2$
$10^3$
$10^4$
$10^5$

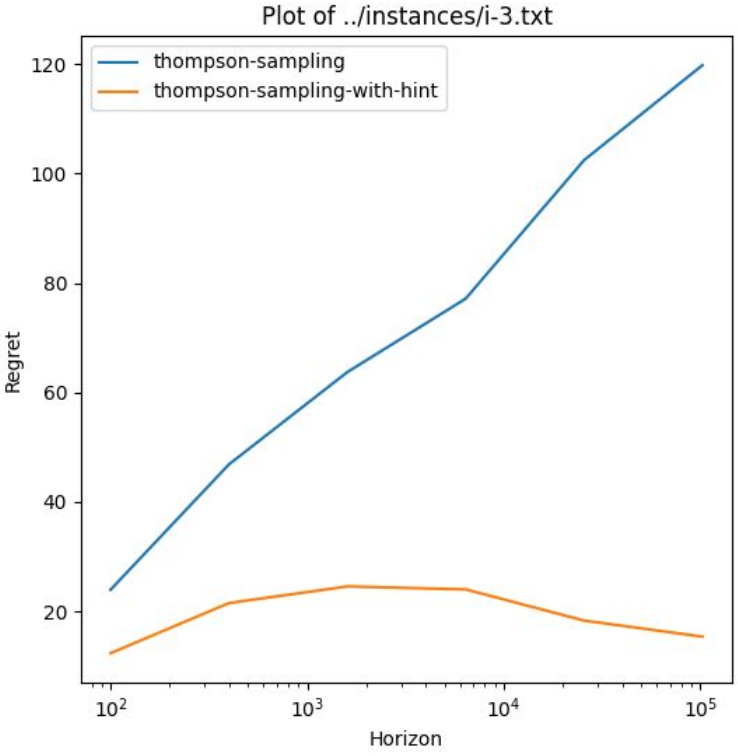
# Task 2

Please find the plots attached below.

The idea here was to work with a belief distribution over the true means which are given as a hint. The algorithm maintains a PMF for every arm and picks the arm which has the highest probability for the highest true mean. I initially tried sampling from the PMFs, akin to what Thompson did to pick an arm, but that gave erratic results which were worse than those of Thompson sampling on average. After picking an arm, the algorithm pulls it and updates the PMF of that arm based on the (Bernoulli) outcome of the pull. The motivation to do this was the Bayesian analysis of Thompson. Knowledge of the true means reduces the search space of true means from a continuous interval to a discrete set. Thus there is an improvement in expected cumulative regret as compared to Thompson sampling.




Plot of ../instances/i-2.txt
thompson-sampling
thompson-sampling-with-hint
Regret
Horizon
0
10
20
30
40
50
$10^2$
$10^3$
$10^4$
$10^5$


Plot of ../instances/i-3.txt
thompson-sampling
thompson-sampling-with-hint
Regret
Horizon
20
40
60
80
100
120
$10^2$
$10^3$
$10^4$
$10^5$

# Task 3

I did indeed find eps_1 > eps_2 > eps_3 such that REG(eps_1) > REG(eps_2) < REG(eps_3). The following results in the brackets are averaged REG of 50 random seeds (0 through 49) for the horizon of 102400.

1. `../instances/i-1.txt: eps_1 = **0.01** (232.18), eps_2 = **0.001** (92.38), eps_3 = **0.0001** (1107.18)`
2. `../instances/i-2.txt: eps_1 = **0.1** (2076.82), eps_2 = **0.003** (1341.84), eps_3 = **0.0003** (4395.92)`
3. `../instances/i-3.txt: eps_1 = **0.1** (4307.6), eps_2 = **0.01** (1054.06), eps_3 = **0.001** (2030.0)`