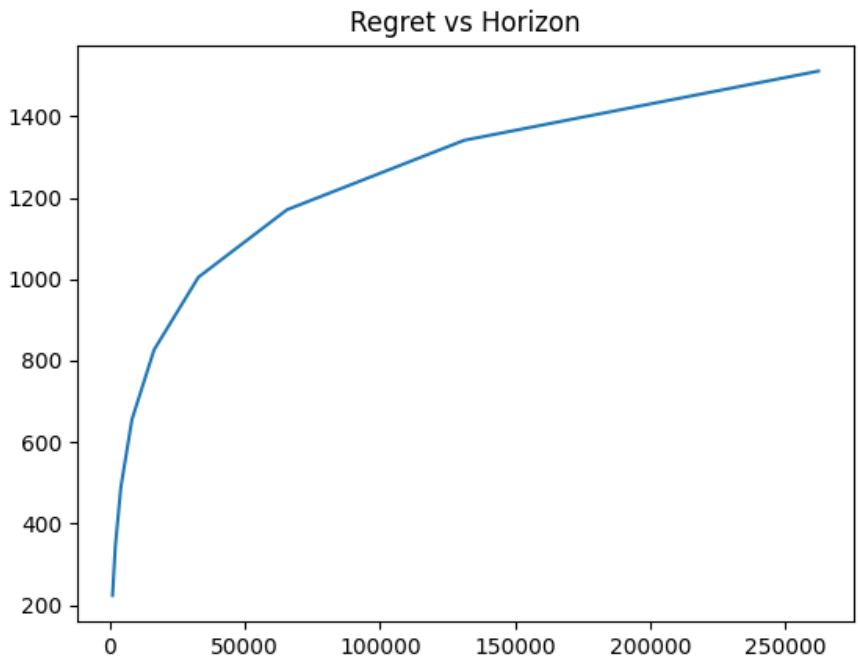
CS747: Assignment 1

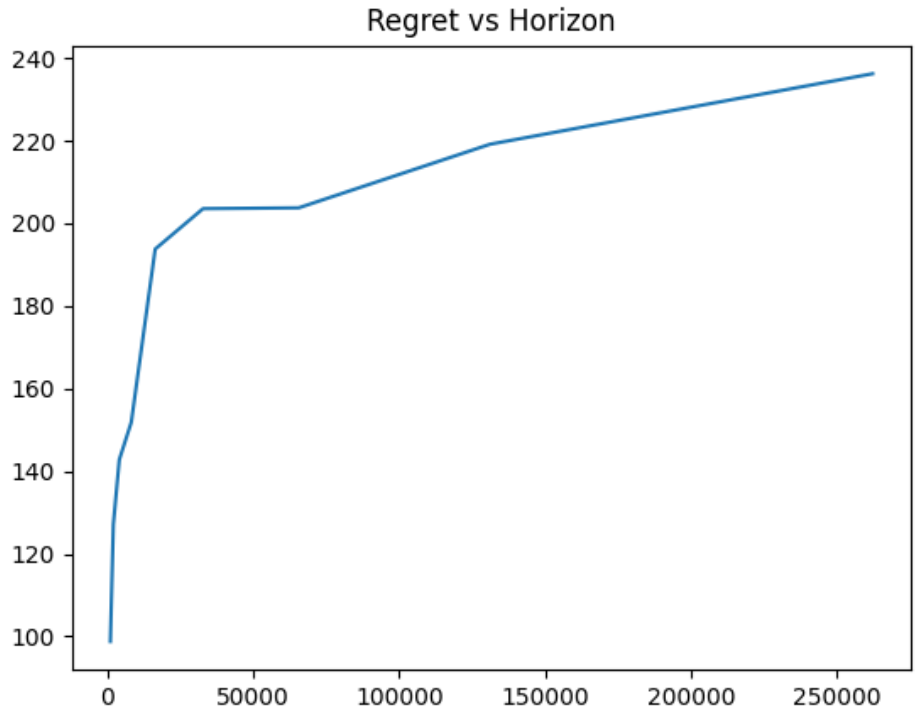
Task 1 :

- UCB:
  - In UCB algorithm, we first have to sample each arm once and maintain counts and values array (same way as in epsilon greedy). I am using round robin for this.
  - After each arm has been sampled once, we start the actual UCB, where in we maintain an array ucb and update ucb for each arm after each pull in the get_reward function according to the definition of ucb . Other than the updation step, everything in get_reward fn is same as it was in Epsilon greedy.
  - The give_pull function for the first num_arms time step outputs arm indices in round robin fashion. Later, it outputs index of the maximal element in ucb array.



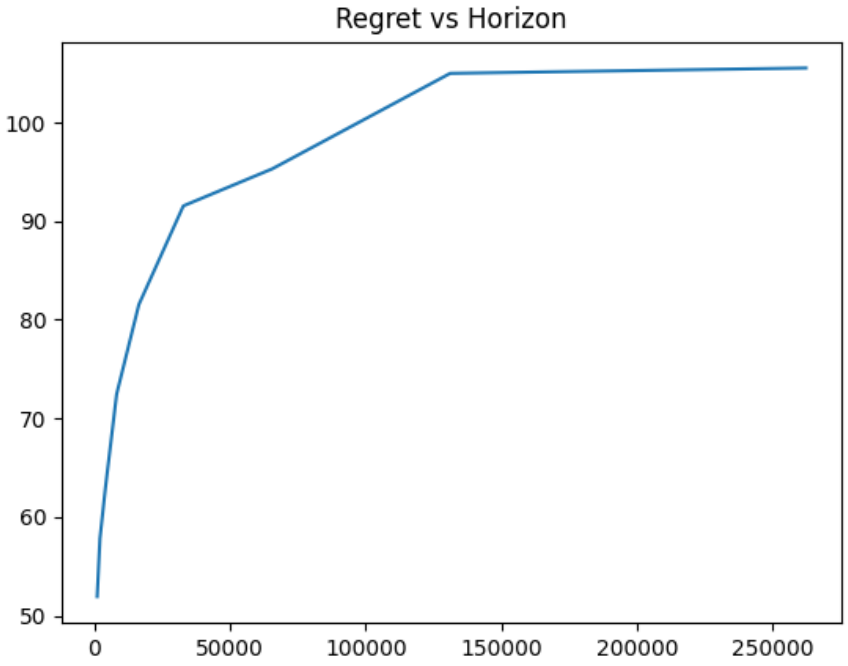
UCB: Regret vs Horizon

- KL UCB:
  - Before we start the actual algorithm, we need to sample each arm once the same way we did for UCB and maintain counts and values array (same way as in epsilon greedy).
  - After each arm has been sampled once, we start the actual KL-UCB where in we maintain a klucb array and update it after each pull according to its definition. Kl_ucb is defined as the root of the equation (shown in lecture). I have employed bisection method to find the root to the equation and thereby the Kl-ucb value for each arm.
  - The get reward function remains the same as in epsilon greedy algo but the change is in updating kl_ucb value using bisection method.
  - The give_pull function for the first num_arms time step outputs arm indices in round robin fashion. Later, it outputs index of the maximal element in klucb array.
  - The graph also shows as tighter regret bound than in UCB.



KL_UCB: Regret vs Horizon

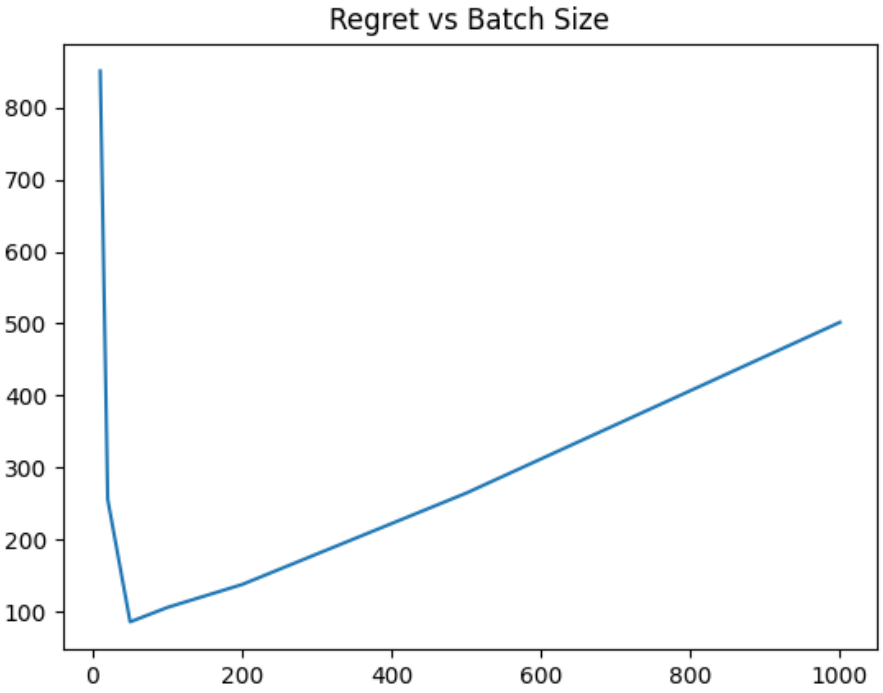
- Thompson Sampling:
  - By far the most elegant approach to MAB problem.
  - Here, we maintain a beta array storing samples from beta distribution beta(s+1,f+1) for each arm.
  - We maintain a success and failure array storing number of success and failures of each arm which we would use in sampling from beta distribution.
  - Also, in the first pull i am pulling an arm at random, this does not affect the result in anyway.
  - And after each pull, for the next pull, you sample from beta distribution after updating success and failure array. And pull the index of maximal element in beta array.
  - The graph also shows lower regret than Kl_ucb.



Thompson-Sampling: Regret vs Horizon

- Task 2:
  - My approach to this question is very intuitive based on the observation of beta distribution specifically the mean and how the mean varies based on the 2 parameters.

- Here the get reward function is the same as in normal Thompson sampling algo, the only difference is that here i am also updating mean expected value of reward for each arm in values array.
- For each batch, i am setting a tolerance level. And all arms in that bandwidth will be selected for the next step that is “deciding how many pulls for each arm”.
- For every batch i am using the success and failure values determined prior to the current batch, and just varying the parameters of beta distribution of each arm while sampling such that the arms that are not pulled get a beta distribution with higher mean, thus helping in exploring.
- Once the batch has been decided it is returned and give pull fn updates the empirical values of variables. And this we use for the next batch’s sampling.
- My approach is based on the fact that a beta distribution with a higher mean will have higher probability of getting a higher sampled value.



As can be seen from the graph, the regret is lowest for batch size of around 50. And rises linealry with batch-size.