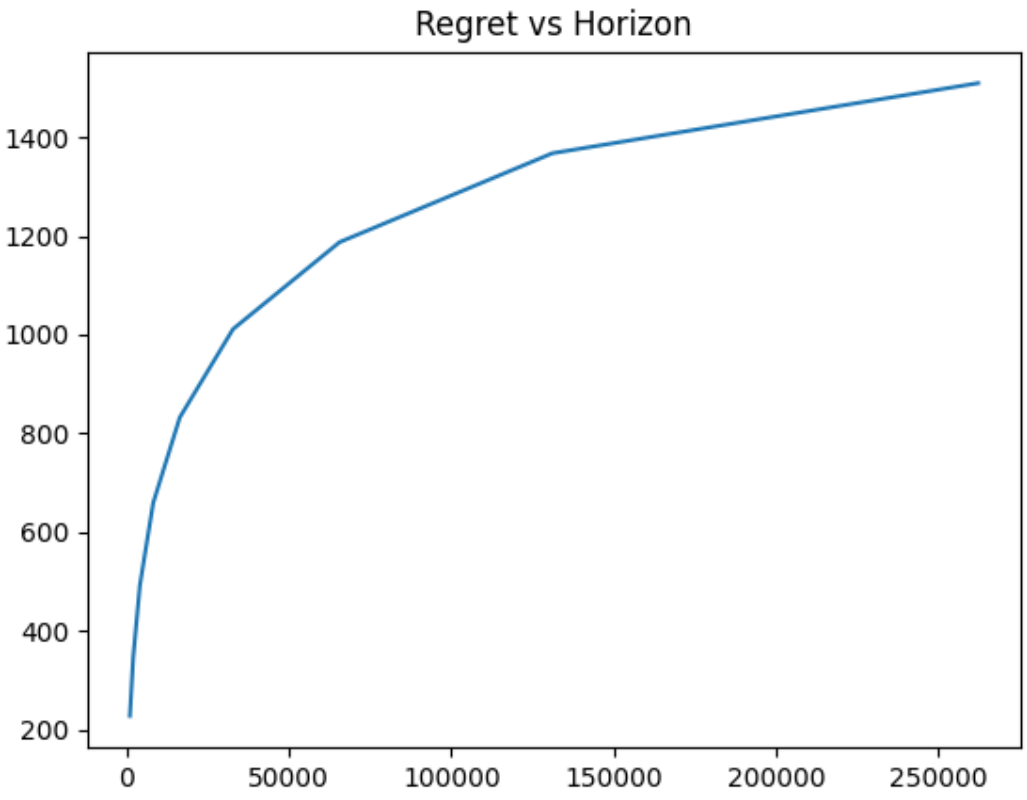
CS 747: Programming Assignment 1 – Report

Task 1:

UCB :-

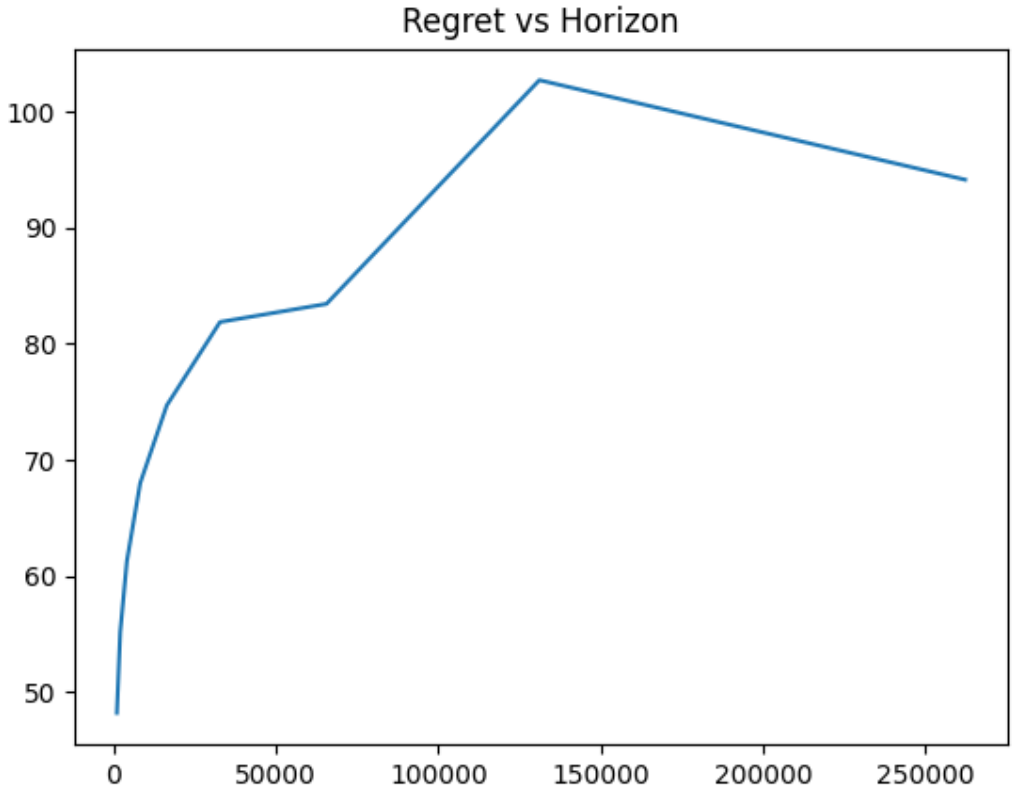


In the UCB algorithm, I have defined three arrays – ucb, u and p, representing the upper confidence bounds, the number of pulls, and the empirical means for each arm. For the initial condition, the algorithm pulls each arm once and returns the time instance as the index of the arm to be pulled when the time instance is less than the number of arms. Once the time instance passes this threshold, it returns the index of the maximum element in the ucb array, thus giving us the arm to be pulled. When it receives the index of the arm pulled and the corresponding reward, the algorithm increases the number of times this arm has been sampled (u) by 1, and then updates the empirical mean (p). Lastly, we update the ucb values for all the arms by using the given formula.

KL-UCB :-

In the UCB algorithm, I have defined three arrays – ucb_kl, u and p, representing the upper confidence bounds, the number of pulls, and the empirical means for each arm. For the initial condition, the algorithm pulls each arm once and returns the time instance as the index of the arm to be pulled when the time instance is less than the number of arms. Once the time instance passes this threshold, it returns the index of the maximum element in the ucb_kl array, thus giving us the arm to be pulled. When it receives the index of the arm pulled and the corresponding reward, the algorithm increases the number of times this arm has been sampled (u) by 1, and then updates the empirical mean (p). Lastly, we perform binary search to find the maximum q for which the inequality defined in the slides holds true (have assumed the value of c as 3). We set the new ucb_kl for each arm as the q we have obtained. I have also defined a helper function KL which returns the value of the KL divergence of the two input numbers.

Thompson Sampling :-



In the Thompson Sampling algorithm, I have defined three arrays – s, f and x, representing the number of successes, the number of failures, and the random sample from the beta distribution of Beta(s + 1, f + 1) for each arm. For the initial condition, when the sums of successes for all the arms and the sums of failures for all the arms are both equal to 0, the algorithm randomly pulls an arm from the entire set of arms. When it receives the index of the arm pulled and the corresponding reward, the algorithm increases the number of failures of that arm by 1 if the reward is 0, or increases the number of successes of that arm by 1 if the reward is 1. Then, the algorithm updates the sample x obtained from the beta distribution for all the arms.

Conclusion for Task 1:-

From the plots, we can see that the Thompson Sampling algorithm gives the lowest regret over the horizon. The highest regret is given by UCB, which justifies its suboptimal behaviour. Both Thompson Sampling and KL-UCB are optimal algorithms and give low regret, approximately an order of magnitude less than UCB.

Task 2:

I tried to implement KL-UCB algorithm for this task, and this was the graph I got. Though it is not right, I would like to explain my approach. I have defined a helper function KL which returns the value of the KL divergence of the two input numbers. I have defined three arrays – ucb_kl, u and p, representing the upper confidence bounds, the number of pulls, and the empirical means for each arm. For the initial condition, that is the first time we make the batch pull, I have shuffled the arms and considered 2 cases – one when batch size is less than the number of arms, where the algorithm just pulls the first few arms from the shuffled array, the size of the array of arms pulled corresponding to the batch size, only once. Hence the pulls_list array would just be an array containing 1s, the size of the array of arms pulled corresponding to the batch size. If the batch size is more than the number of arms, we pull all the arms at least once. Depending on how much the batch size exceeds the number of arms by, we keep on increasing the elements inside pulls_list by 1, till the total number of pulls becomes equal to the batch size. If this is not the first batch pull we are doing and we have ucb_kl values from the previous pulls, we choose the first m arms having the highest values of ucb_kl, where m is a random integer chosen based on the relation between batch size and the number of arms. To decide the number of times the chosen arms have been sampled, I have used the ratio of the ucb_kl for each arm. Thus, the number of pull for each would be the ucb_kl of that arm divided by the sum of all the ucb_kl's of the chosen m arms, multiplied by the batch size. In the get_reward function, I have separated the arm_indexes and the rewards. Then, I update the number of times each arm has been sampled and update the empirical means for each arm. Then, I have performed binary search to find the maximum q for which the inequality defined in the slides holds true (have assumed the value of c as 3). We set the new ucb_kl for each arm as the q we have obtained.

Conclusion for Task 2:-

I have chosen KL-UCB algorithm because it gives optimal regret. It is the most accurate algorithm for estimating the regret for our problem.