CS747 Assignment 1

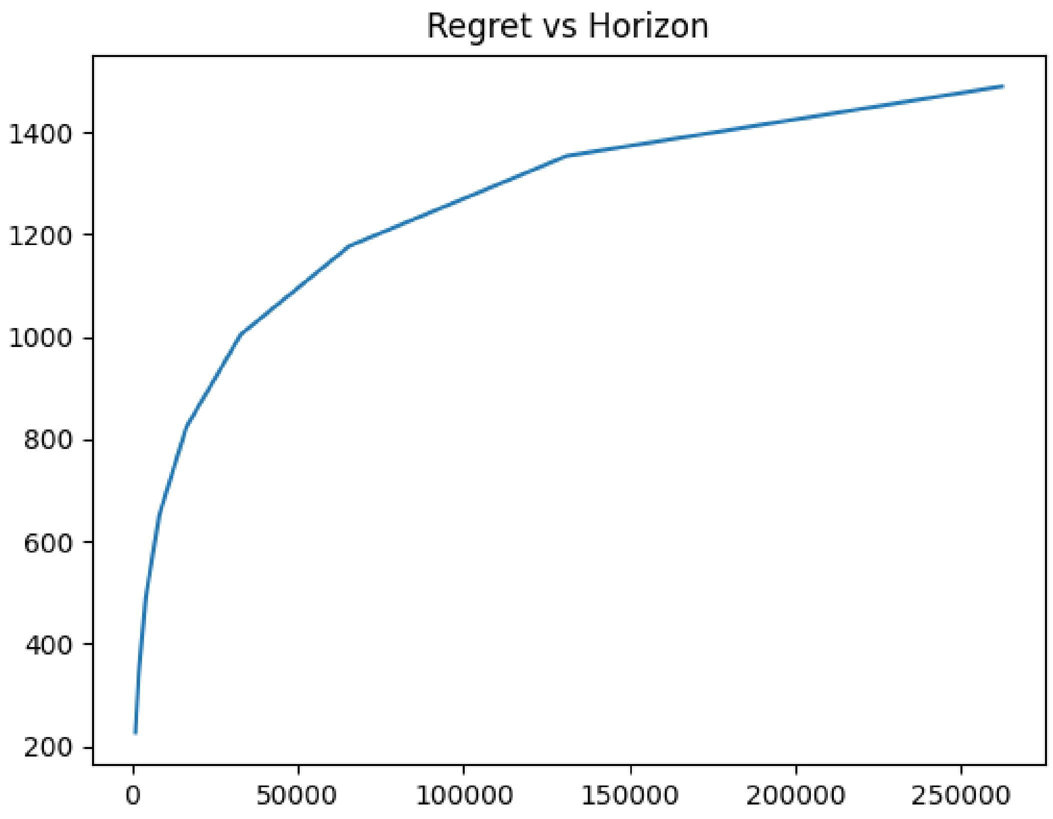
Gaurav Misra
200100063

Autumn 2022

Contents

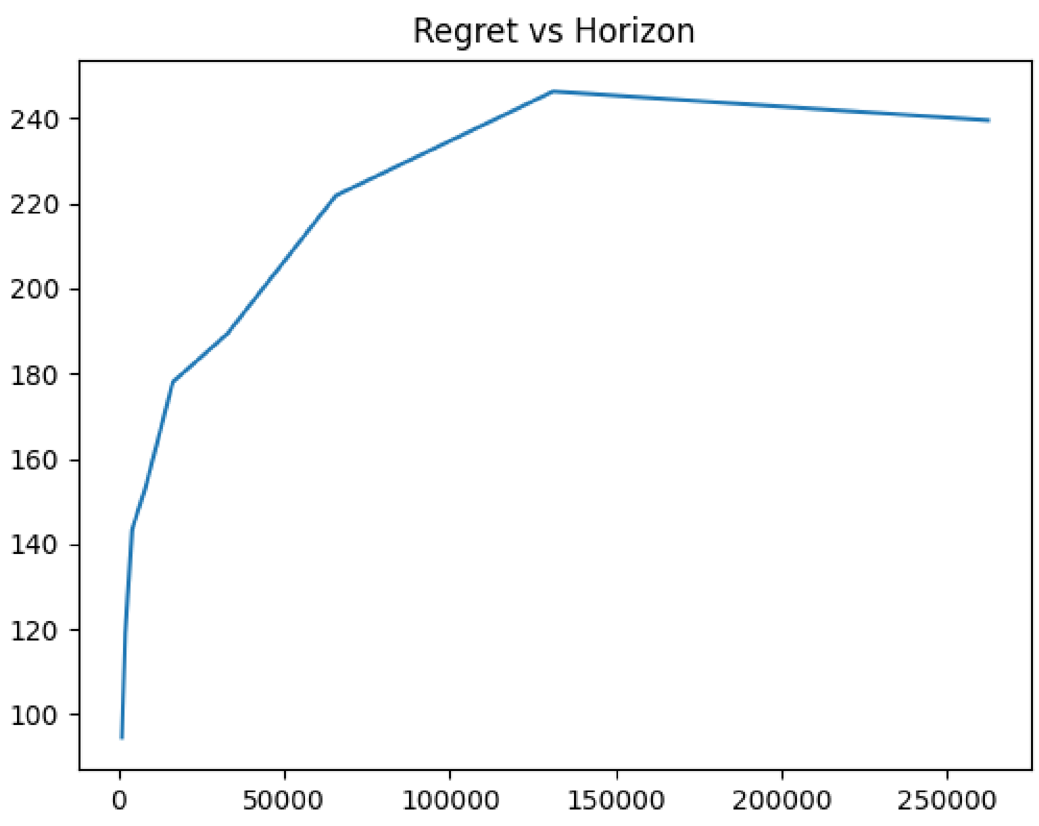
1 Task 1

1.1 UCB



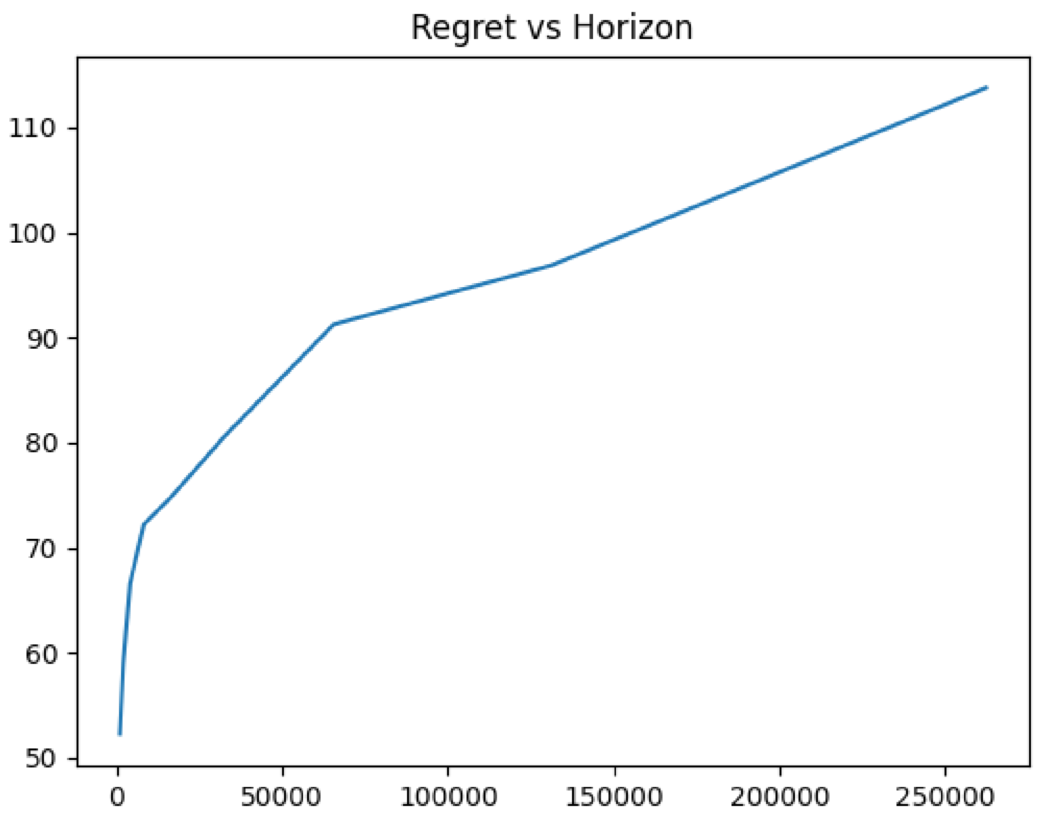
The UCB Algorithm regret is logarithmic, as is evident from the plot.
It's implementation caused no problems whatsoever.
The algorithm basically involves keeping a track of the empirical means for each arm, and then adding to those the exploration bonuses. At any timestep, the arm with the maximum (empirical mean + exploration bonus) value, that is, ucb value, is pulled.
Regret values generated by simulator.py are: [227.36, 344.39, 488.96, 651.74, 823.76, 1004.62, 1177.68, 1353.64, 1489.34]

1.2 KL-UCB



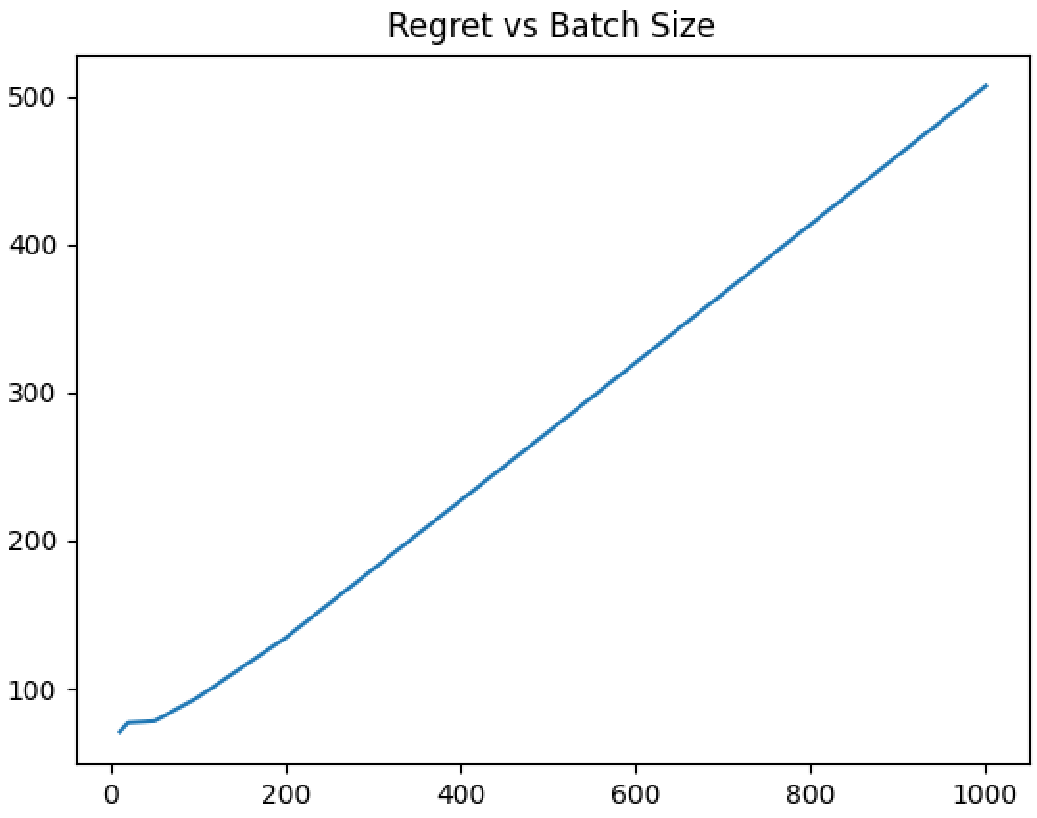
KL-UCB was a much more involved task, especially when it came to the part where bisection method had to be applied, so as to find the optimal arm to sample at each timestep.
An interesting detail is that the curve, though logarithmic overall, curves downward in the end. I believe it would cancel out when averaging is performed over several runs of the code.
The algorithm for KL-UCB involves solving an equation using the bisection method. The roots of said equation are compared for each arm. The arm which returns the maximum root is pulled.
Regret values generated by simulator.py are:[94.62, 119.04, 143.38, 153.42, 178.04, 189.32, 221.82, 246.28, 239.52]

1.3 Thompson Sampling



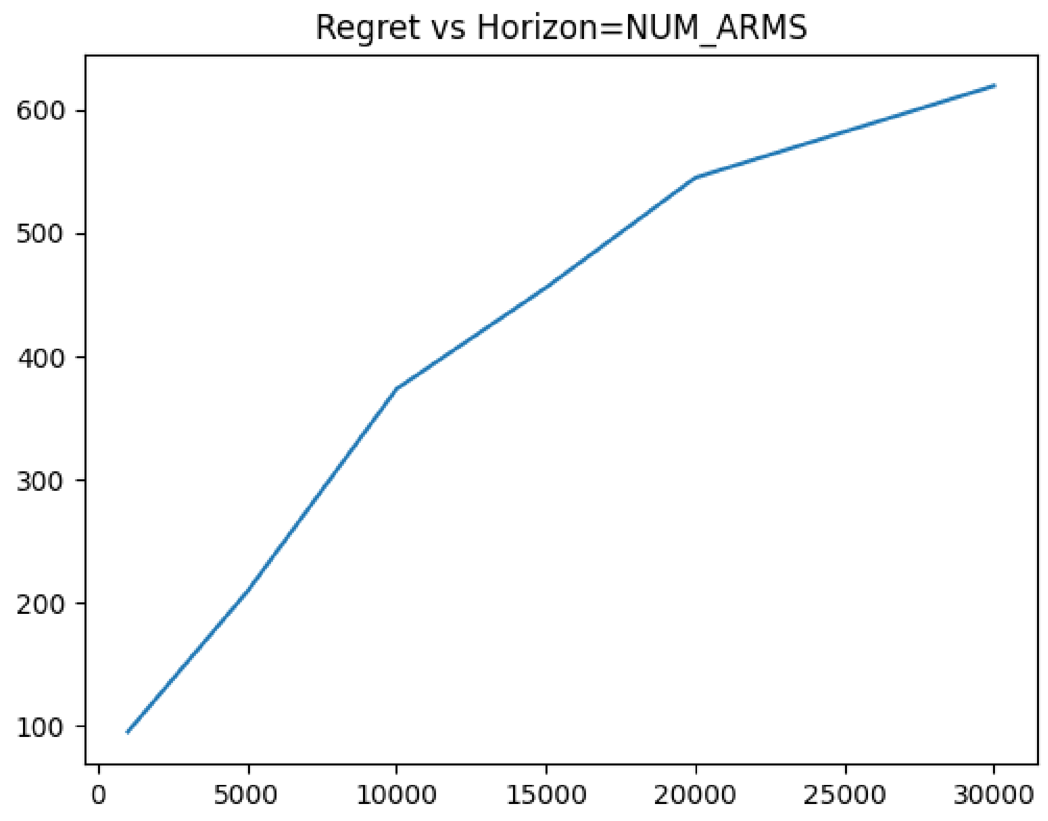
The regret for Thompson Sampling is also logarithmic.
The algorithm involves sampling random values from a list of beta distributions (belief distributions), one for each arm. The arm that returns the maximum value is pulled.
Regret values generated by simulator.py are: [52.28, 58.98, 66.58, 72.20, 74.78, 80.66, 91.26, 96.86, 113.76]

2 Task 2



Task 2 was a pretty involved one.
The curve for Regret vs Batch Size is roughly linear, with slight perturbations at the beginning.
The question that needs to be answered in order to understand why the chosen distribution (creating a list of samples pulled from beta distributions for each arm for each timestep in the batch) was applied is simply this: What policy must one use such that it requires updation only after a batch_size number of timesteps have elapsed?
Regret values generated by simulator.py are roughly: [70.96, 77.00, 78.12, 94.26, 134.44, 273.49, 506.98]

3 Task 3



Task 3 took quite some thinking before it could be implemented. In the end, however, it was simply an application of a small change to a code that was inspired from the idea of an exploiter that did the trick. The regret values come out to be just within the margin set by the testcases.
The idea for the algorithm, as mentioned before, stems from that of an exploiter, which samples an arm as long as it outputs a reward of 1, and switches to a random arm when it encounters a 0.
The change made was that the switch was implemented only when two consecutive zeros were encountered.
The curve for Regret vs Horizon is roughly a straight line.
Regret values generated by simulator.py are: [95.02, 208.89, 373.46, 455.62, 544.94, 619.46]