CS747 Assignment-1 Report

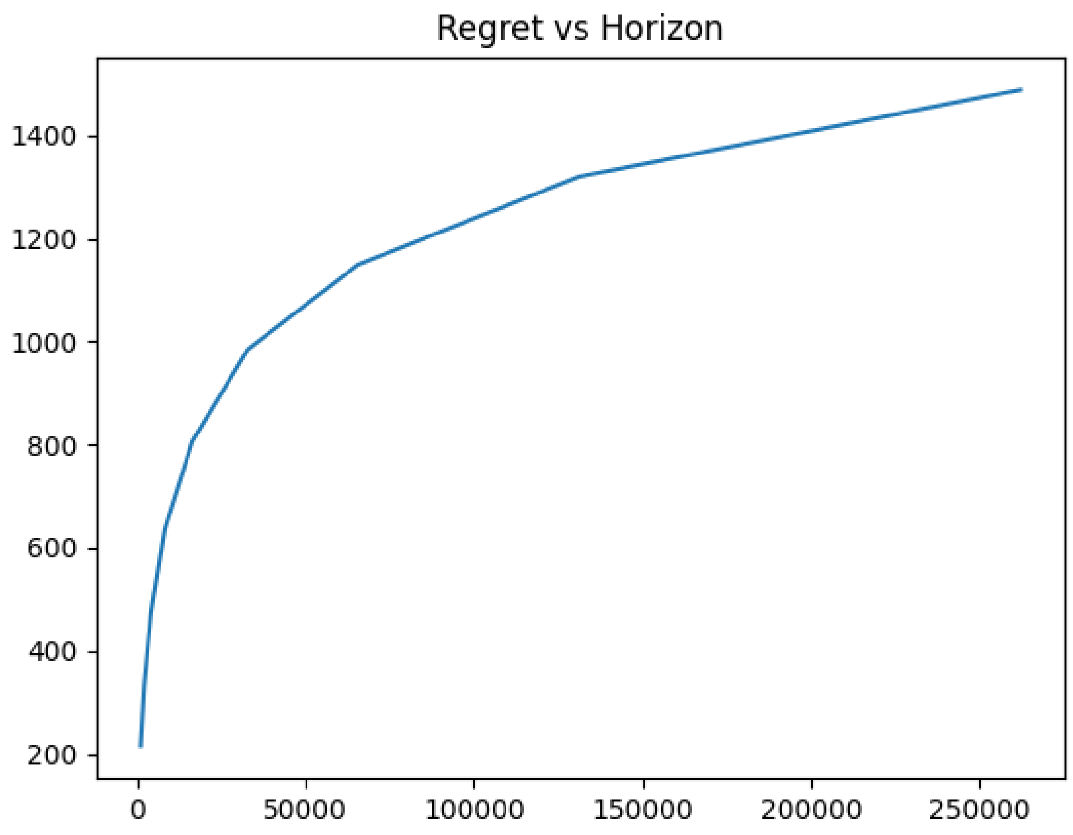
Task1:

a)UCB-

Implemented give_pull function: which calculates the upper confidence bound(ucb) for each arm and then chooses the arm with highest ucb.

ucb= $\hat{p}_{a,t} + \sqrt{\frac{2ln(t)}{u_{a,t}}}$

get reward function:which updates the reward received for each arm based on if it gets a 1 or 0.



b)KL-UCB-

Implemented kl function which implements KL divergence using

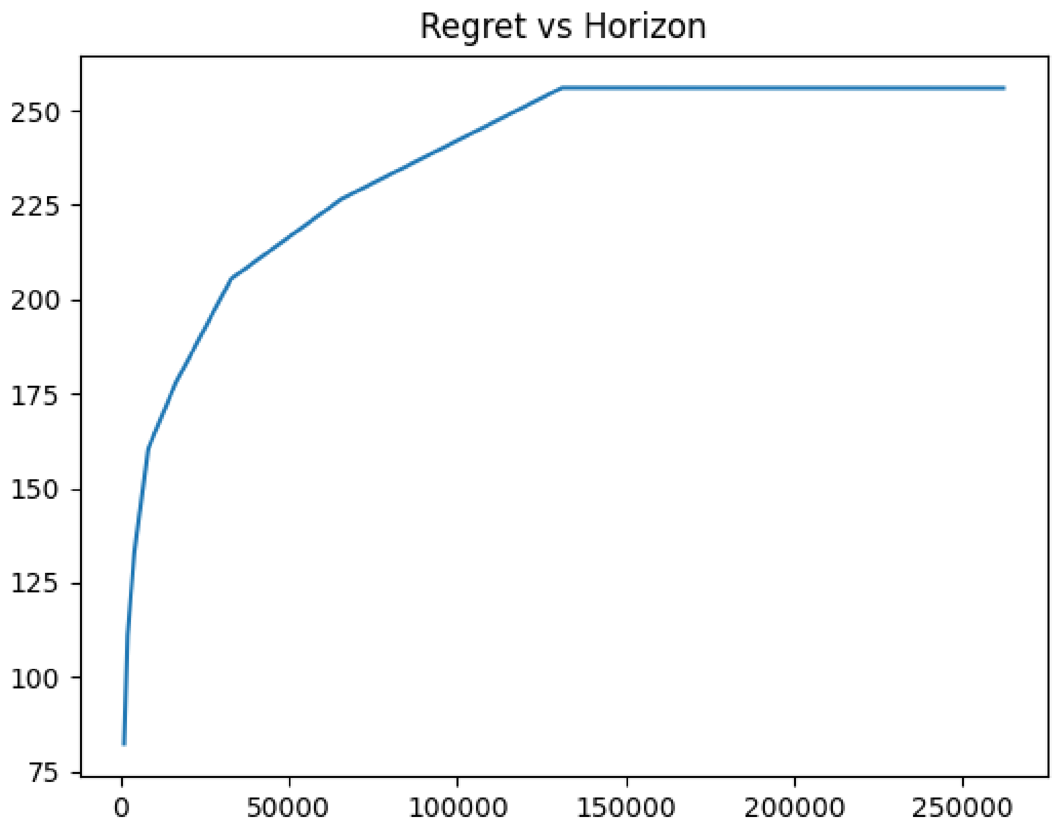
$kl(x, y) = x.log(\frac{x}{y}) + (1 - x).log(\frac{1-x}{1-y})$

Implemented klucb function:which

$ucb - kl_a^t = max\{q \in [\hat{p}_a^t, 1]\ s.\ t.\ u_a^t KL(\hat{p}_a^t, q) \leq ln(t) + 4\ ln(ln(t))\}$

To find the q used binary search function as kl divergence is a monotonic increasing function .If KL(p,q)>value($ln(t) + 4\,ln(ln(t))$) then we have to decrease q and vice versa if inequality is reverse.We use a precision metric to observe the equality constraint since we are dealing with floats.

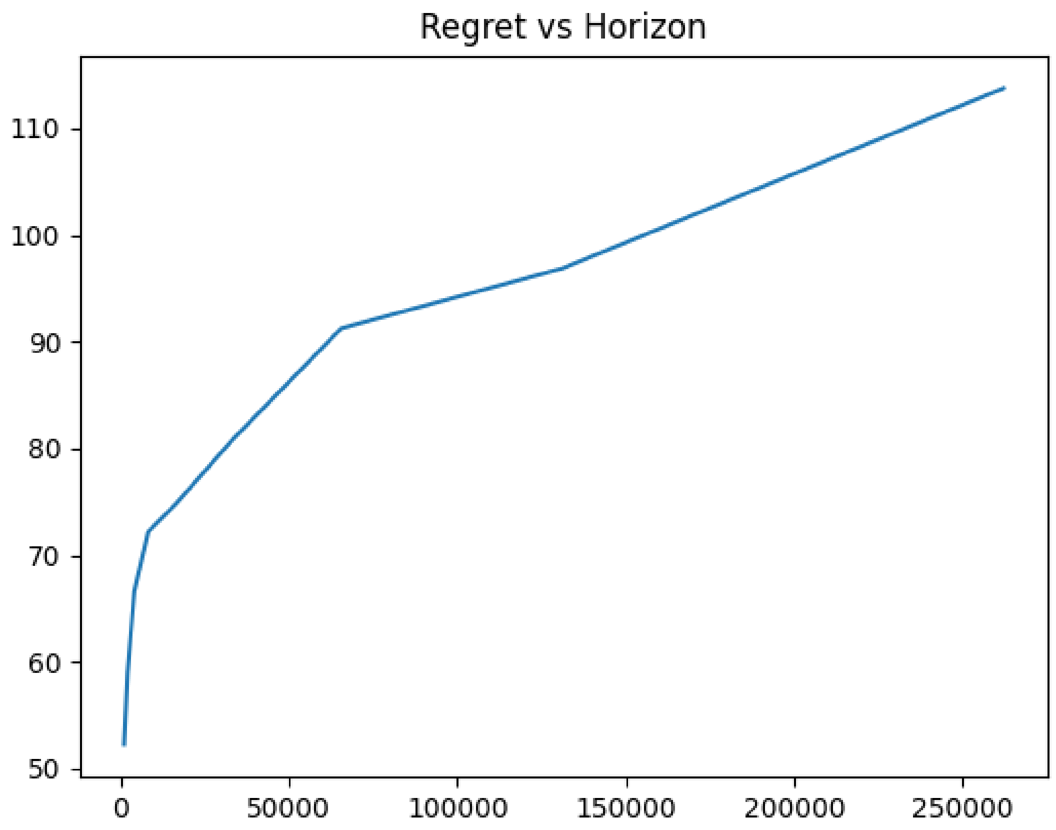
Rest of the functions are similar to UCB.



c)Thompson Sampling-
Implemented give_pull function: which the arm to pull based on the beta beliefs ie one with the highest value is pulled.

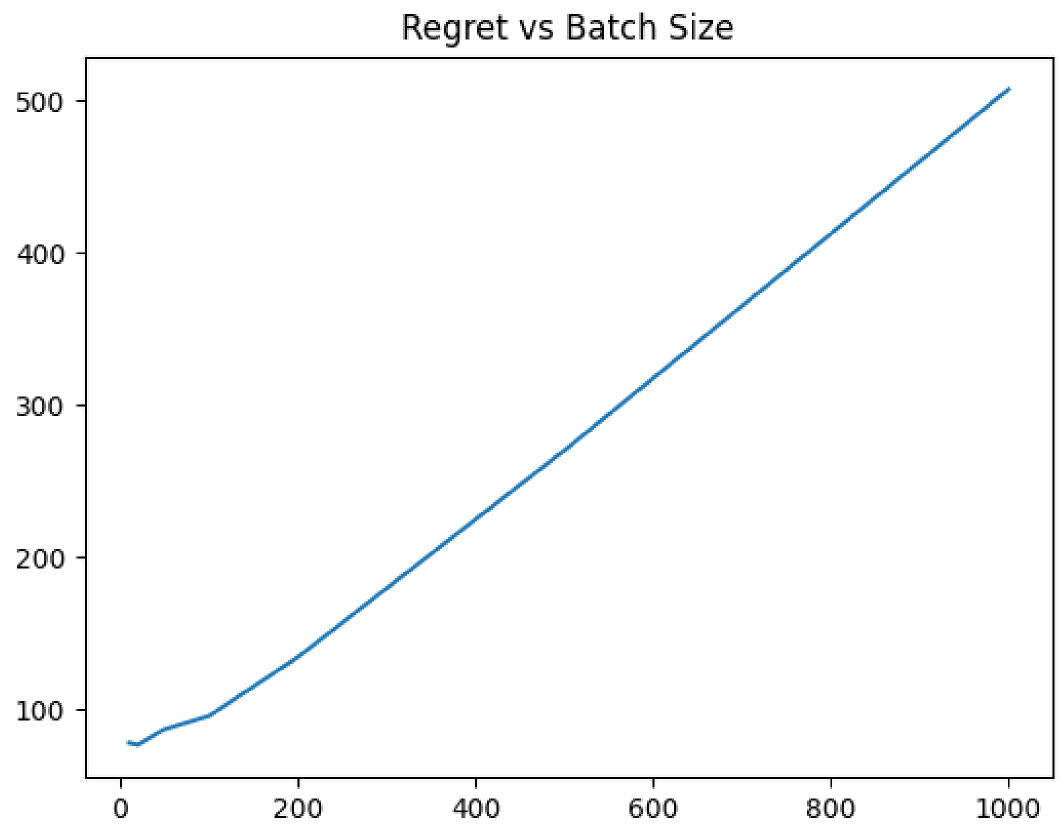
Get_reward function:which is similar to ucb but also keeps track of the success and failures for every arm based on the reward they received.

Regret comparison: Thompson < kl ucb < ucb



Regret vs Horizon
110
100
90
80
70
60
50
0
50000
100000
150000
200000
250000

Task2:
To determine the distribution of arms in a batch I used Thompson sampling batch_size number of times.Each time assign the argmax a pull.The reward function is similar to the Thompson sampling where I keep track of success and failures.



Regret is linear with respect to batch size.Plausible reason could be that we are not able to ascertain our beliefs based on our beta prior.We have to wait until batch size pulls are finished to receive the reward.Therefore smaller the batch size faster we are able to update our beliefs hence smaller regret.

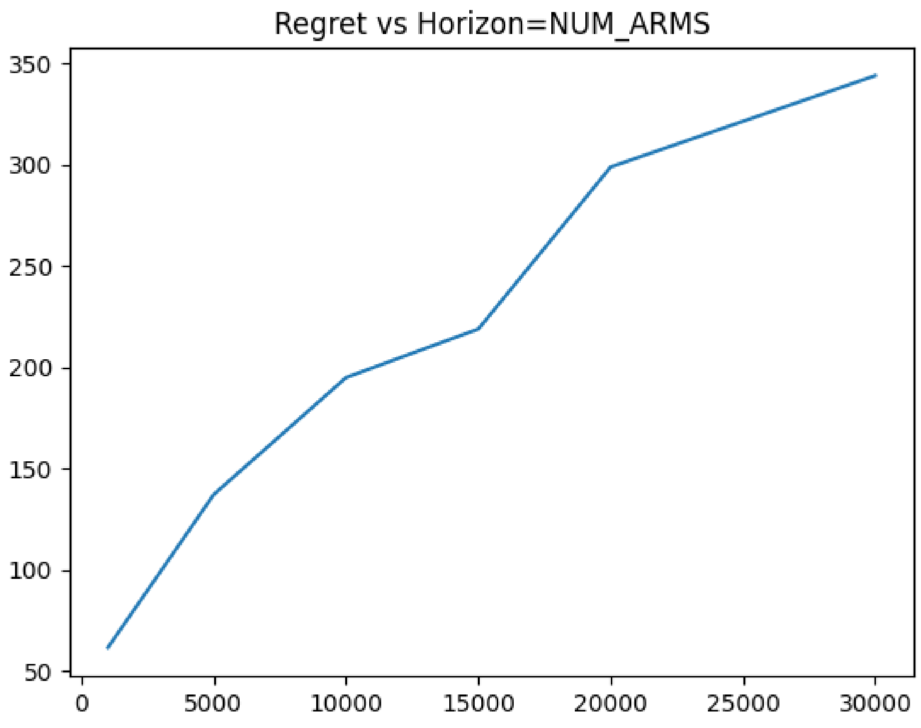
Task 3:
Randomly pick arm from the active permute set:
If it gives us $c.\sqrt{n}$,(c<1)consecutive 1's we identify it as the best arm and continue to pull it for rest of the turns.

If not then store the number of 1's,0's it has produced and pick another arm randomly from the remaining active set. Repeat until the set becomes={}.

If the active set has become empty, pick the arm with the highest number of 1's for the remaining number of turns.

It is noticed that the regret is bounded by O($\sqrt{n}$).As seen from the graph ,the regret is almost of the order O($\sqrt{n}$).In total each arm will be pulled less than $\sqrt{n}$ times and Probability of getting a good arm amongst the $\sqrt{n}$ arms is high.
Take n=1000 arms probability that an arm with greater than 0.9 will be picked in first $\sqrt{n}$ picks is 1- C(900,32)/C(1000,32)=1-4*1e-5 >0.999 therefore quite high.

Therefore Expected regret=number of times arms are switched(i.e we get a zero after consecutive 1's)+Expected regret of one of the best arms=$\sqrt{n}$+$\approx$ 0=O($\sqrt{n}$)



Reference-https://faculty.wharton.upenn.edu/wp-content/uploads/2012/04/Bandit-problems-with-infinitely-many-arms.pdf
https://github.com/Naereen/Kullback-Leibler-divergences-and-kl-UCB-indexes/blob/master/Kullback_Leibler_divergences_in_native_Python__Cython_and_Numba.py