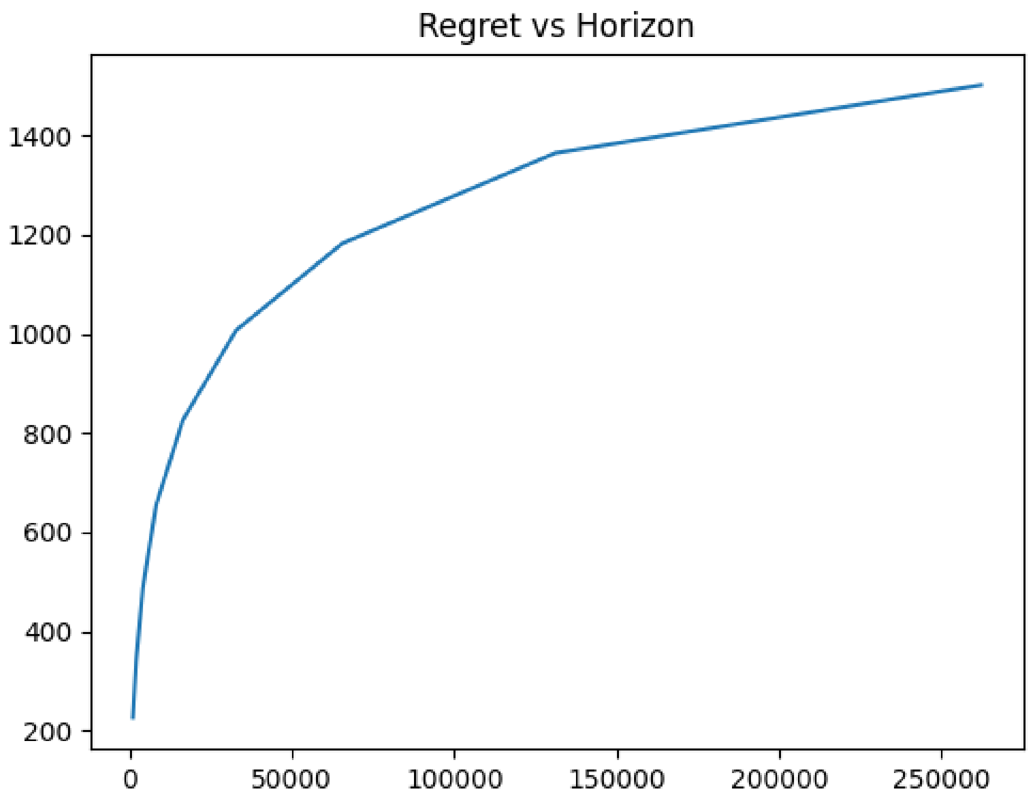
ASSIGNMENT_1 REPORT

Task1

UCB_Algorithm:

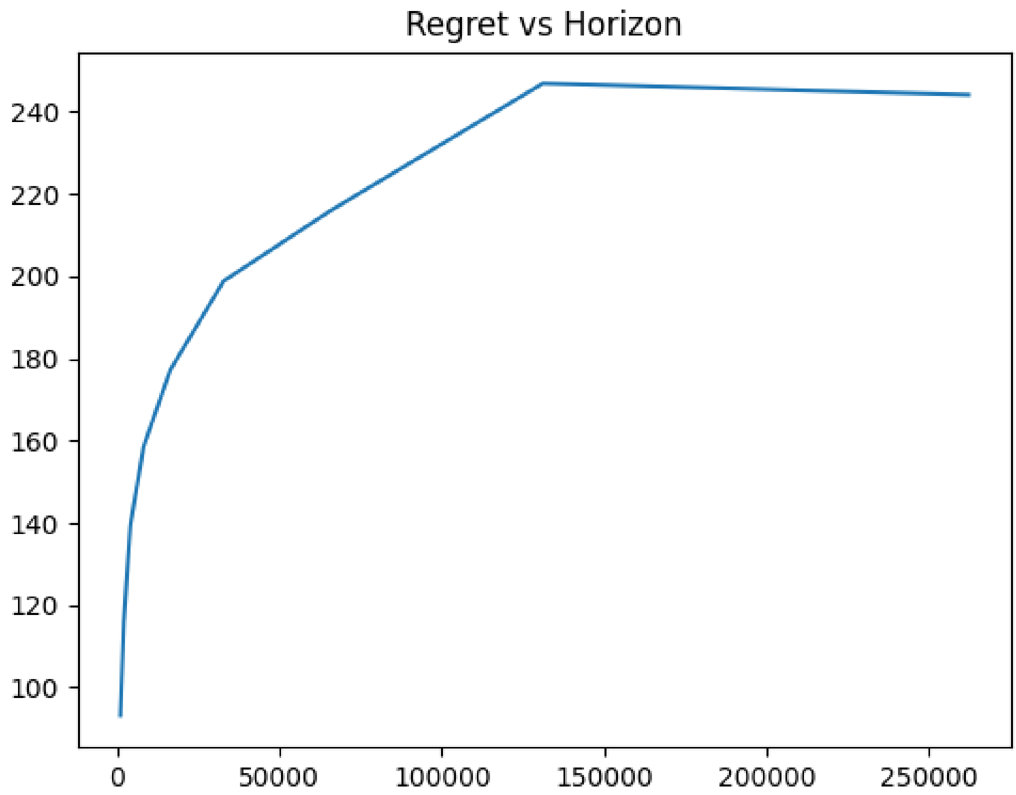
1) give pull performs a Round Robbin for first n pulls and then pulls arms according to ucb algorithm
2)
```
ucb = self.values + np.sqrt((2*np.log(self.time))*np.reciprocal(self.counts))
```
3) attain max valued index from above vector using argmax
4) get reward function updates counts and values , given the arm pulled and reward attained
5) **UCB algorithm grows with log(Horizon) which can be seen in the picture**



6)

KL_UCB:

1) give pull performs a Round Robbin for first n pulls and then pulls arms according to kl_ucb algorithm
2) ucb-kl(t,a) = max{q ∈ [p(a,t) , 1] s. t. u(a,t)*KL(p(a,t) , q) ≤ ln(t) + c ln(ln(t))}, c>=3
3) **performed a binary search to compute q with a precision of 0.001**
4) for cases where p(a,t) = 1 , took q=1
5) kl_ucb has a tighter bound than ucb , which can be seen from the results of autograder.py and the picture



6)

Thompson:

1) Give pull draws a sample from beta distribution on every arm and returns the arm that has maximum sample value, optimal in practice
2) `return np.argmax(np.random.beta(self.success+1,self.failures+1))`
3) get reward updates reward for drawn arm whether it is success or failure

4)
5) Results of autograder.py

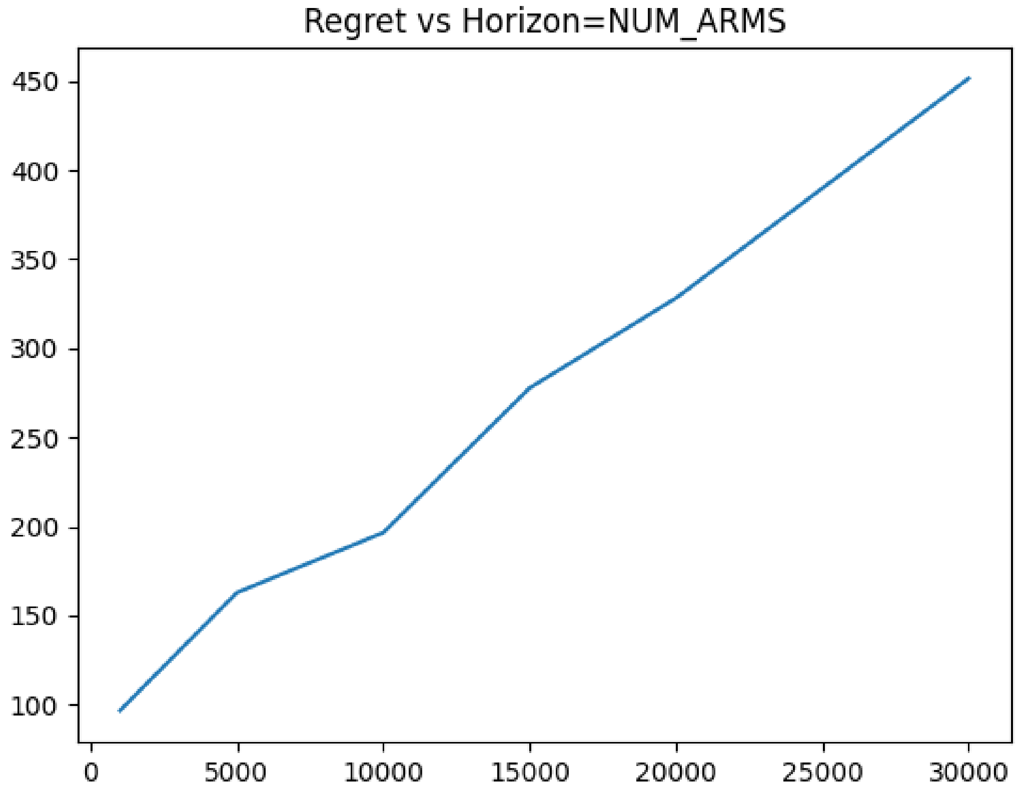
Task2

1) Used Thompson model for pulling k pulls at a time
2) I thought using ucb or kl_ucb is not optimal because without updating reward after each pull, we get the same arm for all k pulls
3) Instead Thompson model gives us different arms to pull even if it has same values and counts because we are drawing random samples from beta distribution.

4)

Task3:

1) Used epsilon greedy algorithm to achieve the results
2) Tried using Thompson model but it is performing worst in this case
3) Started with epsilon = 0.1 and slowly decreased its value to improve the results and finally obtained epsilon = 0.012


Regret vs Horizon=NUM_ARMS
450
400
350
300
250
200
150
100
0
5000
10000
15000
20000
25000
30000

4)