Task 1:
Assumptions made -

1) UCB and KL-UCB are initialized by pulling each bandit once and then starting the algorithm
2) Epsilon greedy and Thompson sampling algorithms are directly started with no history.
3) Ties are broken by the candid of lower index

Epsilon greedy
A function is made of name algoout which takes meanarray and epsilon.
Meanarray - empirical means of all bandits
Epsilon - exploration coefficient
First a random number is generated between 0 and 1 and if the value is less than epsilon we do exploration.
For exploration - we randomly choose any bandit
If value is more, we would do explitation
For exploitation- we directly choose the bandit with highest empirical mean

UCB
A function is made of name algoucb which takes meanarray, tosstimes, totaltoss and scale=2
Meanarray - empirical means of all bandits
Tosstimes - number of times each bandit got pulled
Totaltoss = number of tosses till now
Scale - for task 2 as value of c in exploration term

We compute value for each bandit and then report the bandit with highest value

$$\text{ucb\_arm} = \text{empirical\_mean\_arm} + \text{exploration\_bonus},$$

$$\mathbf{c} \times \ln(\text{total\_pulls}) / \text{pulls\_arm}.$$

ucbarray= meanarray + np.sqrt(scale*np.log(totaltoss)/tosstimes)

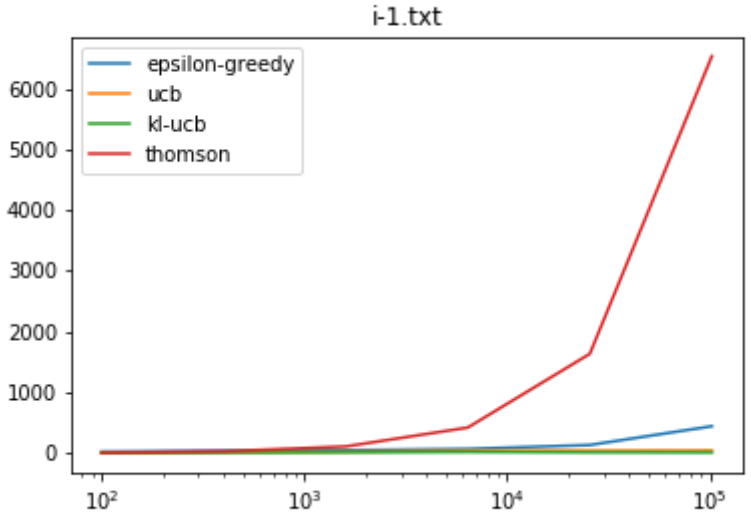
KL-UCB
A function is made of name algoklusb which takes meanarray, tosstimes, totaltoss
Meanarray - empirical means of all bandits
Tosstimes - number of times each bandit got pulled
Totaltoss = number of tosses till now

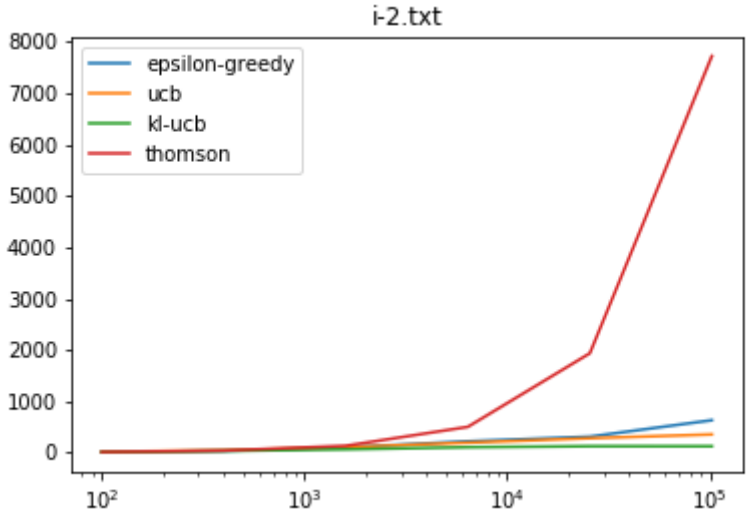
First limitvalue (the value from which ua x kl(pa,q) value should not exceed using totaltoss
Then iterating over all bandits

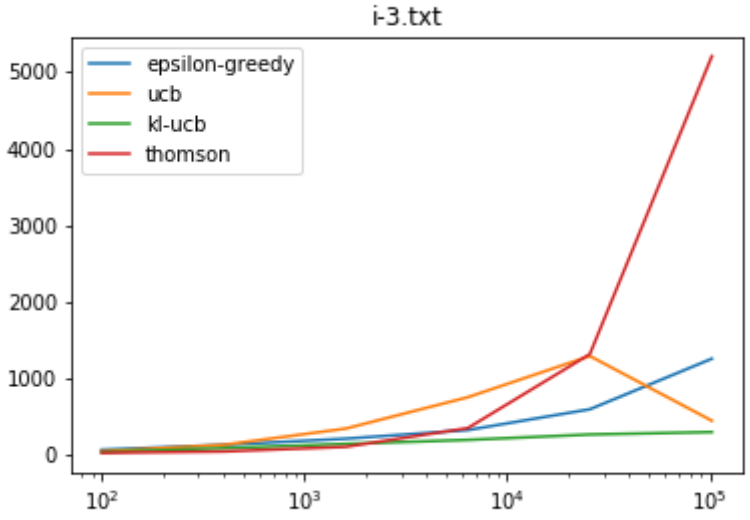
We calculate the value of ua x kl(pa,q) for each value of q ranging from pa to 1 in the difference of 0.01
As the value of this (rhs) exceeds the limit value we break the loop and used the value of q0 of previous run
A function of name KL_div is formed to calculate kl divergence

A function is made of name algothom which takes headtimes,tosstimes,randomSeed
headtimes - number of success of each arm
Tosstimes - total tosses of each arm
randomSeed - seed for beta distribution

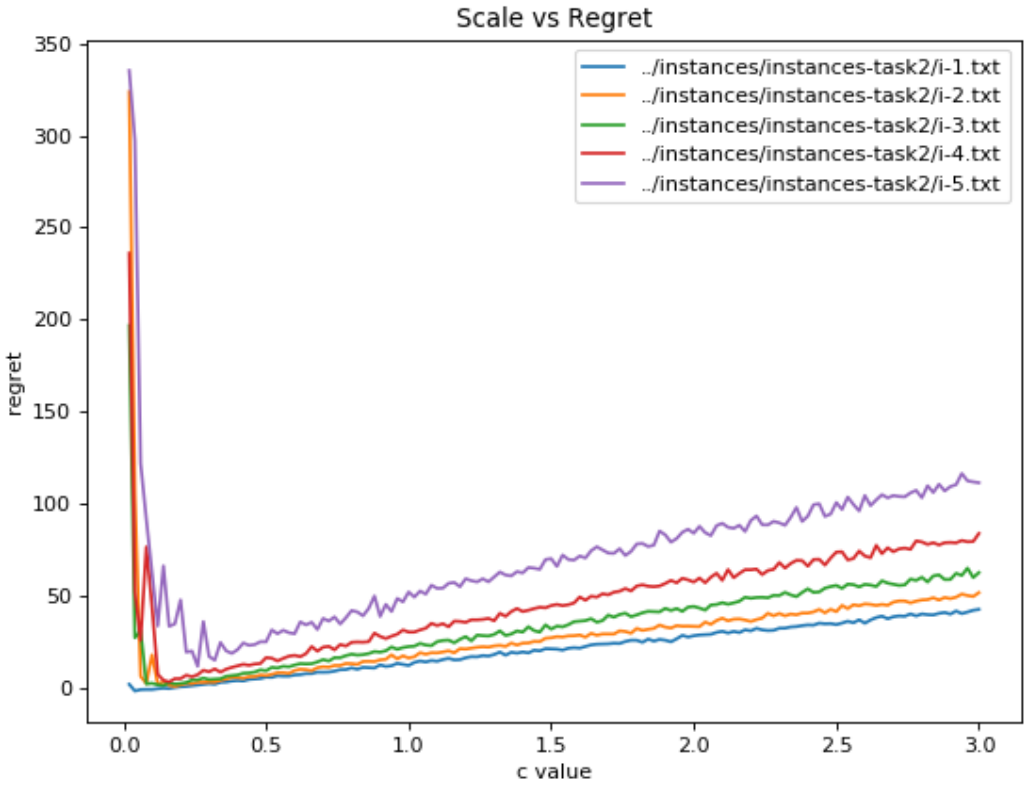
From this failures is calculated by subtracting headtimes from Tosstimes
We compute beta distribution for each bandit upon value of success and failures
And then draw a sample fromeach using numpy function
Whose value is highest is choosen




i-2.txt
epsilon-greedy
ucb
kl-ucb
thomson
8000
7000
6000
5000
4000
3000
2000
1000
0
10^2
10^3
10^4
10^5


i-3.txt
epsilon-greedy
ucb
kl-ucb
thomson
5000
4000
3000
2000
1000
0
10^2
10^3
10^4
10^5

Task2:

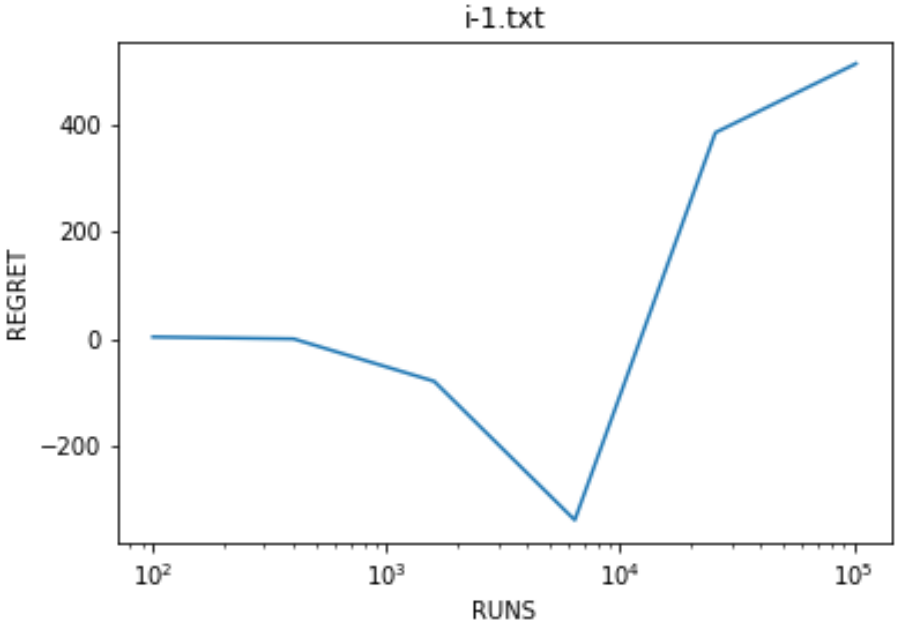


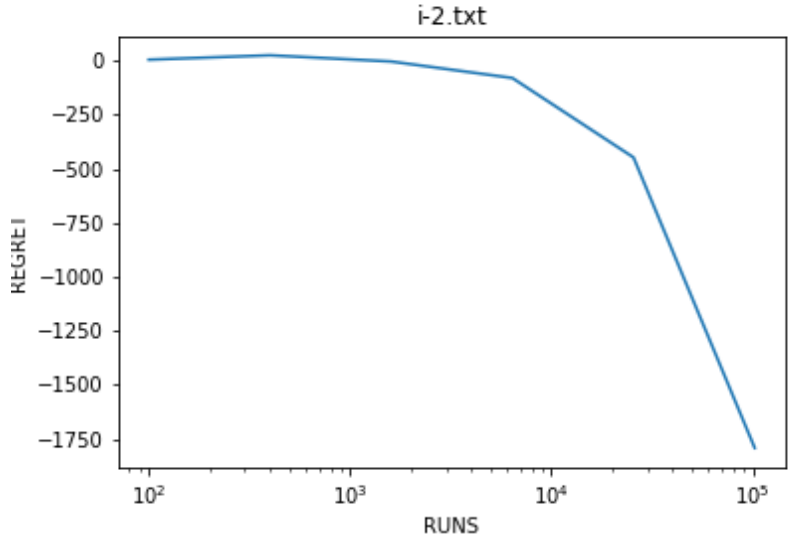
Sorry, by mistake i though we have to plot till 3.

(1) C values for min regret
array([0.04, 0.18, 0.14, 0.16, 0.26])

(2) At verysmall value of c the regret is high because the algorithm is exploring very less and then it decreases as value increasesand attains the minima. After minima the value of regret again increases as te algorithm is doing too much exploration and even after getting a good bandit it tries to use other bandits with low empirical mean

Task3:
Previously we defined coming heads or +1 in bandit as the addition in number of success till now in thomso, here we defined it as the reward to be added in sucess and 1-reward to be added in failure. Although we can get failure by also subtracting number of times that bandit is choosen- successs it has given.


i-1.txt
400
200
0
−200
REGRET
10^2
10^3
10^4
10^5
RUNS


i-2.txt
0
−250
−500
−750
−1000
−1250
−1500
−1750
REGRET
10^2
10^3
10^4
10^5
RUNS