CS747 Assignment 2

Q1.
Value Iteration:
Q[a]+=prob[s][a][j]*(reward[s][a][j]+gamma*V[nxt_state[s][a][j]])
V_new[s] = max(Q)
pi[s]=np.argmax(Q)

Howards Policy Iteration:
Policy evaluation
Policy Improvement

Linear Programming:
problem+=v[s]-gamma*pulp.lpSum([prob[s][a][j]*v[nxt_state[s][a][j]] for j in range(len(prob[s][a]))])>=sum([prob[s][a][j]*reward[s][a][j] for j in range(len(prob[s][a]))])
pi_star[s] = np.argmax(vduales[s, :])

Q2.MDP formulation-
i)Number of states=2*states+2 for losing and winning states

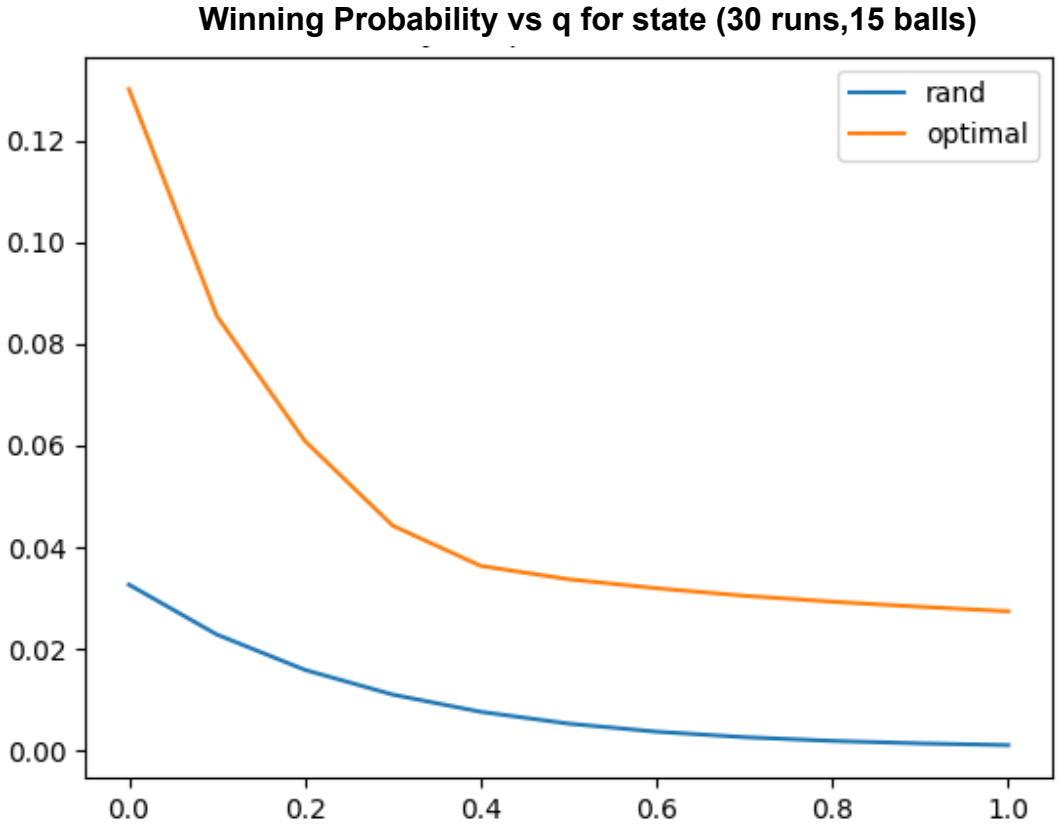
ii)Used encoding similar to hexadecimal but base 30 like 1530,1(15 balls 30 runs and player 1 on strike)=15*30+30=480 and 1530,2 (15 balls 30 runs and player 2 on strike)=15*30+30+500=980 to avoid any overlap therefore used 1000 states.

ii)Used 8 actions ie 0,1,2,4,6,7(for player 2 ie plays 1 action and has 3 outcomes with probability -1,0,1)(0-7=8actions)

iv)Transition to winning state if runs=0
Transition to losing state if runs>0 and balls=0
Strike change
v)Reward =1 if transition to winning state else 0.
vi)Gamma=1 since episodic.

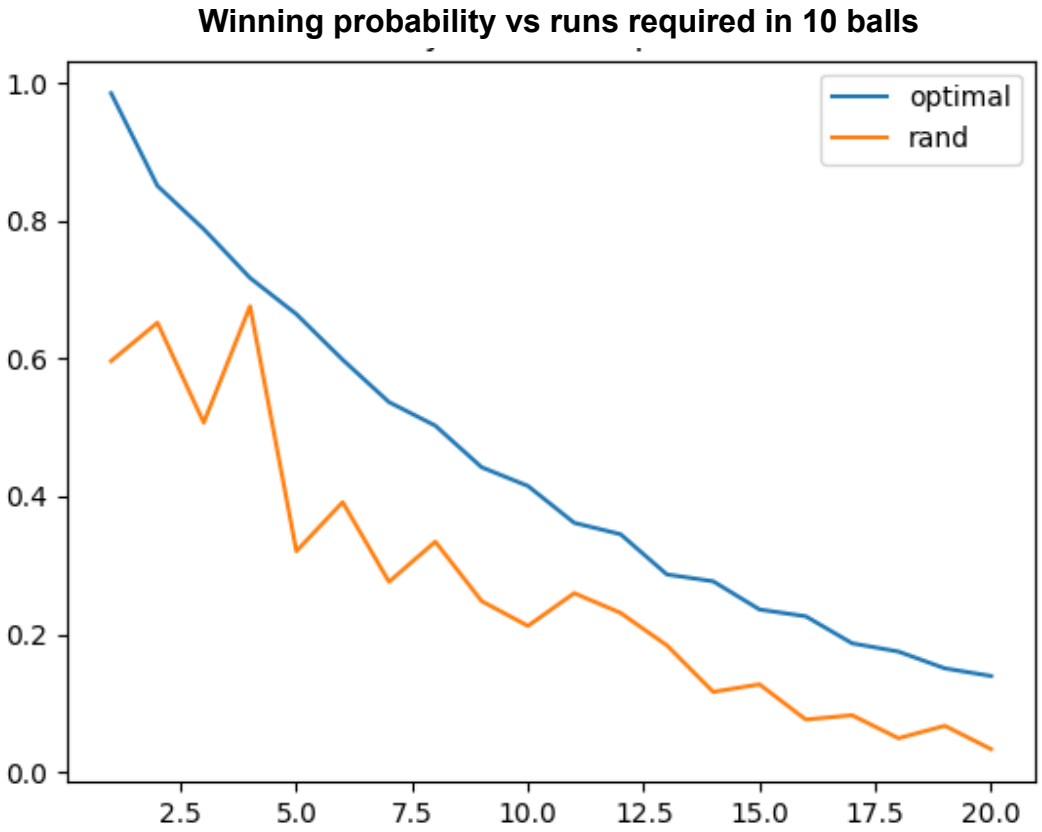
Graphs-

i)



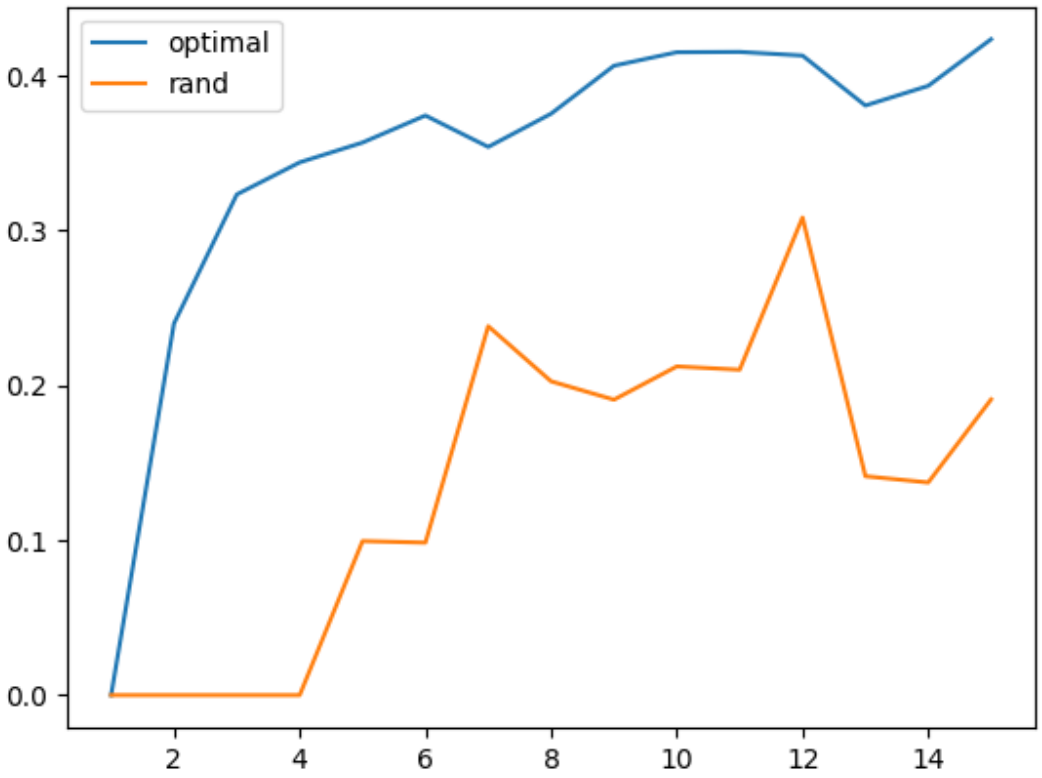
Optimal policy has a higher winning probability which is intuitively correct and the random policy is significantly worse.As B's strength decreases probability of winning decreases.

ii)



Optimal policy is better and as runs required increases winning probability decreases for optimal policy but there is a slight increase in random policy since it is not aware of the optimal action it needs to play.

iii) **Winning probability vs balls remaining to score 10 runs**



Optimal policy is better and winning probability increases as balls increase but there is a slight decrease from 6 to 7 as player A has to take a single which has higher probability of getting out than taking any other action. Similarly for random policy there are wide variations since the player A does not know optimal policy and has higher probability of losing.