CS 747 (Autumn 2025)
Mid-semester Examination and Weeks 5–6 Test

6.30 p.m. – 8.30 p.m., September 14, 2025, LA 001 and LA 002

Name: _____ Roll number: _____

Note. There are four questions in this test. Questions 1, 2, and 3 count towards your mid-semester examination (15 marks total), and Question 4 counts towards your Weeks 5–6 Test (3 marks).

Provide your answer to each question in the space following the question (and *before* the next question if one exists). You can use any blank space in the paper for rough work by drawing a line (either vertical or horizontal), writing "Rough work" on one side of it, and using the demarcated space for rough work.

Question	Marks	
1		/5
2		/5
3		/5
Total:		/15

Question	Marks	
4		/3

Question 1. Let $\mathcal{I}$ denote the set of all 3-armed bandit instances in which each arm yields Bernoulli rewards. Each instance $I \in \mathcal{I}$ is of the form $I = (p_1, p_2, p_3)$, where $p_1, p_2, p_3 \in [0, 1]$ are the mean rewards of arms a_1, a_2, and a_3, respectively. Recall that a *history* records the "arm pulled, reward obtained" sequence when an algorithm interacts with the bandit.

1a. Consider the following history h after 7 pulls, which is generated by a deterministic algorithm L.

$$h \stackrel{\text{def}}{=} (a_3, 1, a_2, 1, a_2, 0, a_3, 0, a_1, 1, a_3, 1, a_2, 0).$$

Which bandit instance $\bar{I} \in \mathcal{I}$ has the highest probability of generating history h while executing L? If there are multiple such instances, specify any one. [2 marks]

1b. What is the probability that $\bar{I}$ (from part 1a) generates h if L is executed? [1 mark]

1c. Suppose we execute a randomised algorithm L', which at each step selects an arm to pull uniformly at random. If L' is executed for 7 steps on instance $I = (p_1, p_2, p_3)$, which history h' has the highest probability of being generated? Describe h' in terms of variables p_1, p_2, and p_3. If there is a tie, you can break it arbitrarily. [2 marks]

Answer 1a. Since L is deterministic and it can generate h, we infer that its first action is always a_3; from history $a_3, 1$ it selects a_2; from history $a_3, 1, a_2, 1$ it selects a_2, and so on. Hence, the probability that instance $I = (p_1, p_2, p_3)$ generates h when executing L is

$$p_3 \times p_2 \times (1-p_2) \times (1-p_3) \times p_1 \times p_3 \times (1-p_2) = p_1 \times \left(p_2 \times (1-p_2)^2\right) \times \left((p_3)^2 \times (1-p_3)\right).$$

This probability is maximised for the instance having $p_1 = 1$, $p_2 = \frac{1}{3}$, $p_3 = \frac{2}{3}$.

Answer 1b. The probability of generating h while runining L on the maximising instance from part 1a is

$$1 \times \frac{1}{3} \times \frac{4}{9} \times \frac{4}{9} \times \frac{1}{3} = \frac{16}{729}.$$

Answer 1c. Since L' selects arms uniformly at random at each step, the probability of the subsequence of arms within any history is identical for all histories—in fact exactly equal to $\left(\frac{1}{3}\right)^7$. The difference in the probabilities of different histories arises from the probability of $0's$ and/or $1's$ being generated as rewards by the respective arms. If in history h, each arm a generates s_a 1's and f_a 0's, then the probability of h is

$$q = \left(\frac{1}{3}\right)^7 (p_1)^{s_1}(1-p_1)^{f_1}(p_2)^{s_2}(1-p_2)^{f_2}(p_3)^{s_3}(1-p_3)^{f_3}.$$

In this question, we have (p_1, p_2, p_3) fixed; we are asked for which history (or histories) the probability q is largest. Since we have the constraint that $s_1 + f_1 + s_2 + f_2 + s_3 + f_3 = 7$, we have to work out how to split 7 into portions to allot as exponents to the factors p_1, $1-p_1$, p_2, $1-p_2$, p_3, $1-p_3$ such that the product is maximised. Clearly we should allocate all 7 units to the largest among these, breaking ties arbitrarily. If allotting to s_a for some arm a, the rewards in h' would all be 1; if allotting to f_a for some arm a, the rewards in h' would all be 0. Formally, let

$$\bar{a} = \operatorname*{argmax}_{a \in \{a_1, a_2, a_3\}} \left| p_a - \frac{1}{2} \right|.$$

Then we can take

$$h' = \begin{cases} (\bar{a}, 1, \bar{a}, 1, \bar{a}, 1, \bar{a}, 1, \bar{a}, 1, \bar{a}, 1, \bar{a}, 1) & \text{if } p_{\bar{a}} \geq \frac{1}{2}, \\ (\bar{a}, 0, \bar{a}, 0, \bar{a}, 0, \bar{a}, 0, \bar{a}, 0, \bar{a}, 0, \bar{a}, 0) & \text{otherwise.} \end{cases}$$

Question 2. Consider an n-armed bandit setup, $n \geq 2$, in which each arm yields Bernoulli rewards. The arms are $a_1, a_2, \ldots, a_n$, and their respective mean rewards are $p_1, p_2, \ldots, p_n \in [0, 1]$.

We examine a setting in which algorithms for interacting with this bandit must specify a *pair of distinct arms* to pull on each time step. Thus, on time step $t \geq 0$, the algorithm specifies (x^t, y^t) such that x^t and y^t are both from the set $\{a_1, a_2, \ldots, a_n\}$, and $x^t \neq y^t$. Now, the environment selects one of x^t and y^t uniformly at random (that is, each with probability $1/2$), pulls the selected arm, and returns the 0 or 1 reward obtained. For example if (a_5, a_8) is the pair specified by the algorithm at some time step, then the environment selects one of these arms uniformly at random. Suppose a_5 is selected. Then a_5 is pulled, generating a reward of 1 with probability p_{a_5}, and a reward of 0 with probability $1 - p_{a_5}$. Only the reward (0 or 1) is communicated back to the algorithm—it is *not* disclosed which arm from the pair was pulled to obtain the reward. This process continues until T pulls have been performed, where $T \geq 2$ is the horizon.

Let $\mathcal{L}$ be the set of all algorithms from the family described above (that is, those specifying a pair of arms to pull at each time step). For algorithm $L \in \mathcal{L}$, bandit instance I, and horizon T, let $\text{rew}(L, I, T)$ denote the expected cumulative reward obtained by L on I over horizon T. For a given bandit instance I and horizon T, let $\text{rew}^\star(I, T)$ be the maximum expected cumulative reward achievable by any algorithm: that is,

$$\text{rew}^\star(I, T) = \max_{L \in \mathcal{L}} \text{rew}(L, I, T).$$

Accordingly, for algorithm $L \in \mathcal{L}$, instance I, and horizon $T \geq 2$, we define

$$\text{regret}(L, I, T) = \text{rew}^\star(I, T) - \text{rew}(L, I, T).$$

Describe an algorithm $L \in \mathcal{L}$ such that for every instance I and horizon $T \geq 2$,

$$\text{regret}(L, I, T) \leq C_I \ln(T),$$

where C_I is a constant depending on I. Prove that your algorithm achieves this regret upper bound, and comment on the form taken by C_I (an exact expression is not required). You are free to reuse results presented in class. [5 marks]

Answer 2. For $i, j \in \{1, 2, \ldots, n\}$, $i < j$, we treat the pair (i, j) as an arm in a new bandit setup I'. There are $\binom{n}{2}$ arms in I'. Playing (i, j) on I' is accomplished by playing i and j as a pair on the original instance I. Clearly the reward obtained is Bernoulli, with mean

$$\frac{1}{2}\{p_i(1) + (1 - p_i)(0)\} + \frac{1}{2}\{p_j(1) + (1 - p_j)(0)\} = \frac{1}{2}(p_i + p_j).$$

All we have to do is attain logarithmic regret on I', which is accomplished by implementing, say, UCB or Thompson Sampling on I'. In particular, we have to maintain statistics (such as successes and failures) for each pair (i, j), rather than separately for each arm in I.

The regret upper bound for this algorithm would be $C_I \ln(t)$ where C_I is the product of a constant with the sum of the "gap terms" of non-optimal pairs of arms. Define

$$r_{\max} \stackrel{\text{def}}{=} \max_{1 \le i < j \le n} (p_i + p_j).$$

Then we would have a gap term of the form $\frac{1}{r_{\max} - p_i - p_j}$ for each (i, j) pair, with $1 \le i < j \le n$, satisfying $p_i + p_j < r_{\max}$.

Question 3. Consider the `python` program shown below right.

Recall that `numpy.random.random()` returns a number drawn uniformly at random from $[0, 1)$. Also recall that `elif` is `python` syntax for "else if", so at most one of the lines marked L1, L2, and L3 can be executed within any iteration of the `while` loop.

Answer the questions below about the output of the program: that is, the number t that is printed out. Assume that the environment computes with infinite precision and no overflows. For all three parts, show steps/reasoning to arrive at your answers.

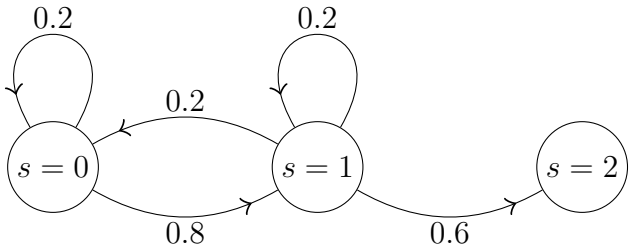
3a. What is the set of all possible values that t can take? [1 mark]

3b. What is $\mathbb{E}[t]$—that is, the expectation of t? Provide as a number. [3 marks]

3c. What is $\mathbb{P}\{t < \mathbb{E}[t]\}$—that is, the probability that t is smaller than its expectation? Provide as a number. [1 mark]

```
########## START ##############
import numpy

s = 0
t = 0

while s < 2:
    increment = 0
    x = numpy.random.random()

    if(s == 0 and x < 0.8):
        increment = 1 #L1
    elif(s == 1 and x < 0.2):
        increment = -1 #L2
    elif(s == 1 and x < 0.8):
        increment = 1 #L3

    s = s + increment
    t = t + 1

print(t)
########### END ###############
```

Answer 3. It is convenient to first visualise the flow of the program as a Markov Chain (that is, an MDP with a single action from each state); the figure below shows the states and transition probabilities.



t is nothing but the number of steps taken from state "$s = 0$" to reach state "$s = 2$", which is terminal.

Answer 3a. Clearly t takes values from the set $\{2, 3, 4, \ldots, \infty\}$.

Answer 3b. The answer is $V(s = 0)$, which is obtained by solving these Bellman equations:

$$\begin{aligned}
V(s = 0) &= 0.2\{1 + V(s = 0)\} + 0.8\{1 + V(s = 1)\}, \\
V(s = 1) &= 0.2\{1 + V(s = 0)\} + 0.2\{1 + V(s = 1)\} + 0.6\{1 + V(s = 2)\}, \\
V(s = 2) &= 0.
\end{aligned}$$

The answer works out to $\mathbb{E}[t] = V(s = 0) = \frac{10}{3}$.

Answer 3c. This answer can be worked out from the answers to parts 3a and 3b.

$$\begin{aligned}
&\mathbb{P}\{t < \mathbb{E}[t]\} \\
&= \mathbb{P}\{t = 2\} + \mathbb{P}\{t = 3\} \\
&= \mathbb{P}\{s = 0 \to s = 1 \to s = 2\} + \mathbb{P}\{s = 0 \to s = 0 \to s = 1 \to s = 2\} + \mathbb{P}\{s = 0 \to s = 1 \to s = 1 \to s = 2\} \\
&= 0.8 \times 0.6 + 0.2 \times 0.8 \times 0.6 + 0.8 \times 0.2 \times 0.6 \\
&= 0.672.
\end{aligned}$$

Question 4. Consider an arbitrary MDP (S, A, T, R, γ), notation as usual, which encodes a continuing task with $\gamma < 1$. Suppose there are $n \geq 2$ states and $k \geq 2$ actions. Let Π be the set of all deterministic, Markovian, stationary policies $\pi : S \to A$ for the MDP. Hence, $|\Pi| = k^n$. Now consider the following claim, denoted C.

> Claim C. The k^n distinct policies in Π can necessarily be arranged in a sequence such that for any two consecutive policies in the sequence, the latter policy's value function dominates or equals the former policy's value function. In other words, we can name the $|\Pi|$ policies $\pi_1, \pi_2, \ldots, \pi_{|\Pi|}$ such that for each $i \in \{1, 2, \ldots, |\Pi| - 1\}$, $\pi_{i+1} \succeq \pi_i$.

Is Claim C correct? If your answer is yes, provide a proof. If your answer is no, provide a counterexample. [3 marks]

Answer 4. Claim C is not correct. If the claim was correct, it would imply that for every pair of policies π and π', either "$\pi \succeq \pi'$" or "$\pi' \succeq \pi$" must be true. On the other hand, we know that there are MDPs with pairs of *incomparable* policies—such that each policy in the pair has a strictly larger value for some state than the other policy.

Concretely, consider a 2-state, 2-action MDP, in which both actions a_1 and a_2 have self-loops at each state; let the states be s_1 and s_2. The reward is 0 for every transition of a_1, and 1 for every transition of a_2. Take $\gamma = \frac{1}{2}$. And let $\pi_{i,j}$ denote the policy taking a_i from s_1 and a_j from s_2 for $i, j \in \{1, 2\}$. The value functions of these policies are given below.

Policy π	$V^\pi(s_1)$	$V^\pi(s_2)$
π_{11}	0	0
π_{12}	0	2
π_{21}	2	0
π_{22}	2	2

Policies π_{12} and π_{21} are incomparable—hence no sequence that contains both of them can have the property required in Claim C.