Assignment #1

Due date: October 11th, 2023 at 11:59pm

- Please write your name here Bradley Scott
- Please write your UID here 119775028
- Please write your solutions in the free space after problem statements.
- Your grades depend on the correctness and clarity of your answers.
- Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.
- Write your answers in the spaces provided. If needed, attach other pages.

(1)

1. A standard 52-card deck includes thirteen *ranks* of {2, 3, ... ,10, *J, Q, K, A*} in each of the four *suits* {♦, ♣, ♥, ♠}. In Poker, any subset of 5 cards is called a *hand*. A hand is called a *full house* if it includes exactly three cards of the same rank and two cards of another rank.

(a) Suppose that we draw 5 cards out of the 52 cards randomly. What is the probability that these 5 cards form a full house?

(b) Suppose that we draw 7 cards out of the 52 cards randomly. What is the probability that there exists at least one full-house hand within these 7 cards?

a) Let X = drawing a full house from a 52 card deck when drawing 5 cards
Then P(X)= $\frac{\#\ of\ ways\ to\ make\ a\ full\ house\ from\ a\ 52\ card\ deck}{\#\ of\ ways\ to\ get\ 5\ cards\ from\ a\ 52\ card\ deck}$

The denominator is easy so let's start there.
We want the number of unique combinations since a hand of {2,3,4,5,6} = a hand of {6,5,4,3,2} for our purposes. This means we want the number of combinations, not permutations. The formula for this is n choose k, annotated as $\binom{n}{k}$ and C_k^n, and equal to $\frac{n!}{k!(n-k)!}$

So the # of ways to get 5 cards from 52 cards is $C_5^{52} = \frac{52!}{5!(52-5)!} = \frac{52*51*50*49*48}{5*4*3*2} =$ 2,598,960 combinations

The numerator is a little trickier.
We can find this by doing $C_1^{13}C_3^4C_1^{12}C_2^4 = \frac{13!}{1!12!} * \frac{4!}{3!1!} * \frac{12!}{1!11!} * \frac{4!}{2!2!}$=13*4*12*2*3 = 3744

This is first choosing one of the ranks from the 13 options, then choosing 3 out of 4 of those cards, then choosing a different rank from the 12 remaining options, and finally choosing 2 out of 4 of those cards.

So then we have P(X) = $\frac{3744}{2598960} =$ .001440

b) Using a similar approach to 1a we have
Let X = drawing a full house from a 52 card deck when drawing 7 cards

Then P(X) = $\frac{\#\ of\ ways\ to\ make\ a\ full\ house\ and\ 2\ additional\ cards\ from\ a\ 52\ card\ deck}{\#\ of\ ways\ to\ get\ 7\ cards\ from\ a\ 52\ card\ deck}$

The denominator is $C_7^{52} = \frac{52!}{7!(52-7)!} = \frac{52*51*50*49*48*47*46}{7*6*5*5*3*2} = 133{,}784{,}560$

The numerator is $C_1^{13}C_3^4C_1^{12}C_2^4C_2^{47}$ because we have at least one full house so the remaining 2 cards can be anything from the 52-5 =47 cards.
$C_1^{13}C_3^4C_1^{12}C_2^4C_2^{47} = 3744 * \frac{47!}{2!(47-2)!}$=$3744 * \frac{47*46}{2} = 3744 * 47 * 23 = 4{,}011{,}456$
So we have P(X) = $\frac{4{,}011{,}456}{133{,}784{,}560} \approx 0.02998$

(2)

2. Suppose that you play the following game against a house in Las Vegas. You pick a number between one and six, and then the house rolls three dices. The house pays you \$1,500 if your number comes up on one dice, \$2,000 if your number comes up on two dices, and \$2,500 if your number comes up on all three dices. However, *you must pay the house \$1,000* if your number does not show up at all. How much can you expect to win (or lose)?

Pick one number between 1 and 6

House rolls 3 dice

You lose 1000 if your number does not show up at all

You win 1500 if your number shows up once

You win 2000 if your number shows up twice

You win 2500 if your number shows up three times

E(X) = ?

Well, what's the odds your number shows up exactly once?

Suppose your number is 1. We want P(X=1)

There is a $\left(\frac{1}{6}\right)$ chance of rolling a 1 and a $\left(\frac{5}{6}\right)$ chance of rolling any number except one.

Number of ways to arrange exactly one 1 in a roll of 3 dice = 3

So P(X=1) = $3 * \left(\frac{1}{6}\right) * \left(\frac{5}{6}\right)^2 = \frac{25}{72} \approx 0.3472$

What's the odds your number shows up exactly twice?

Same as above, we have a $\left(\frac{1}{6}\right)$ chance of rolling a 1 and a $\left(\frac{5}{6}\right)$ chance of rolling any number except one.

Number of ways to arrange exactly two 1's in a roll of 3 dice = 3

So P(X=2) = $3 * \left(\frac{1}{6}\right)^2 * \left(\frac{5}{6}\right) = \frac{5}{72} \approx 0.0694$

What's the odds your number shows up exactly 3 times?

Number of ways to arrange three 1's in a roll of 3 dice = 1

So P(X=2) = $\left(\frac{1}{6}\right)^3 = \frac{1}{216} \approx 0.0046$

What's the odds your number doesn't show up at all = 1- P(your number shows up at least once)

= $1 - \frac{1}{216} - \frac{15}{216} - \frac{75}{216} = \frac{125}{216} \approx 0.5787$

So E(X) = $-1000\left(\frac{125}{216}\right) + 1500\left(\frac{25}{72}\right) + 2000\left(\frac{5}{72}\right) + 2500\left(\frac{1}{216}\right) = \frac{2500}{27} \approx$ 92.59 expected to win

(3)

3. You are looking for your hat in one of six drawers. There is a 10% chance that it is not in the drawers at all, but if it is in a drawer, it is equally likely to be in each. Suppose that we have opened the first two drawers and noticed that the hat is not in them.

(a) What is the probability that the hat is in the third drawer?

(b) What is the probability that the hat is not in any of the drawers?

a)

The original probability of it being in any one specific drawer can be found by solving 6x + 0.1 = 1

Which gives us that x = 0.15

So each drawer has a 15% chance of having the hat in it before we had opened any drawers.

After having opened two drawers, we have removed 30% of the original probability.

So the new probability will be $\frac{\textit{the original probability}}{\textit{the new probability space percentage of the old probability space}}$

$= \frac{.15}{.7} = \frac{3}{14}$

So **P(hat is in the third drawer, after checking that it's not in the first two) =** $\boldsymbol{\frac{3}{14}}$

b) **P(hat is not in any of the drawers, after having checked that it's not in the first two) = 1-P(hat is in one of the drawers, after having check that it's not in the first two)**

= 1 - $\boldsymbol{\left(4 * \frac{3}{14}\right) = \frac{1}{7}}$

(4)

4. Let S be the set of all sequences of three rolls of a dice.

- Let X be the sum of the number of dots on the three rolls. What is $E(X)$?
- Let Y be the product of the number of dots on the three rolls. What is $E(Y)$?

E(X = sum of the number of dots of three rolls of a dice) = ?

We are assuming the die is a 6 sided die. The answer would very drastically if we assumed it was a 3 sided die, or a 20 sided die.

$E(X_1 = sum\ of\ the\ number\ of\ dots\ of\ 1\ roll\ of\ a\ die) = \frac{1(1)+1(2)+1(3)+1(4)+1(5)+1(6)}{6} = \frac{21}{6}$=3.5

Since expected values are additive and the same die is assumed to be being used each time, we have 3.5 + 3.5 + 3.5 = 10.5

E(X) = 10.5

E(Y = the product of the number of dots on the three rolls) = ?

Since $E(X_1) = E(X_2) = E(X_3) = 3.5$, each roll of the die is independent and has the same expected value, we have E(Y) = E($X_1 * X_2 * X_3) = E(X_1) * E(X_2) * E(X_3)$ =3.5*3.5*3.5 = 42.875

(5)

5. Find x, y, and z such that

- $x \le 10$
- $x + y \le 17$
- $2x + 3z \le 25$
- $y + z \ge 11$
- $15x + 2y + z$ is maximized

Your solution:

- x : [ ]
- y : [ ]
- z : [ ]
- $15x + 2y + z$: [ ]

```python
import pulp as p

Lp_prob = p.LpProblem('Problem', p.LpMaximize)

# Create problem Variables
x = p.LpVariable("x", lowBound = 0, cat = 'Integer')   # Create a variable x >= 0
y = p.LpVariable("y", lowBound = 0, cat = 'Integer')   # Create a variable y >= 0
z = p.LpVariable("z", lowBound = 0, cat = 'Integer')   # Create a variable z >= 0

# Objective Function
Lp_prob += 15 * x + 2 * y + z

# Constraints:
Lp_prob += x <= 10
Lp_prob += x + y <= 17
Lp_prob += 2*x + 3*z <= 25
Lp_prob += y + z >= 11

status = Lp_prob.solve()   # Solver

print(p.LpStatus[status])   # The solution status

# Printing the final solution
print(p.value(x), p.value(y), p.value(z), p.value(Lp_prob.objective))
```

x = 8, y = 9, z = 3 is the optimal solution with an objective function value of 141 for when x,y and z are integers.

Alternatively, changing the variables to continuous gives Optimal x=8.6 y=8.4, z= 2.6 with an objection function value of 148.4

6. We are given a graph G with vertex set $V(G)$ and edge set $E(G)$. Every edge e has a weight w_e. Write an LP whose optimal solution is equal to the length of the shortest path from a vertex s to a vertex t in this graph. You may write some constraints over the edge set or vertex set of the graph. For instance, you can write

$$\forall (u,v) \in E(G) \qquad\qquad w_u f_u + w_v f_v \leq 10.$$

In the above example, f_i's are the variables of your LP.

Let V(G) = $\{v_1, v_2, \dots$ be the set of all vertices

Let E(G) = $\{e_{a,b}$ such that a $\in V(G)$, b $\in V(G)$ and $e_{a,b}$ *is the edge between* v_a *and* v_b

And let $e_{a,b} = \{ \begin{matrix} 1, \textit{this edge was used in the path} \\ 0, \textit{this edge wasn't used in path} \end{matrix}$

Every edge has a corresponding weight

So let W(G) = $\{w_{a,b}$ such that a $\in V(G)$, b $\in V(G)$ *and* $w_{a,b}$ *corresponds to* $e_{a,b}$

Then the length of any path from vertex s to t is $\sum_s^t e_{a,b} * w_{a,b}$

So we want to minimize $\sum_s^t e_{a,b} * w_{a,b}$

For an LP relaxation $e_{a,b}$ can take any value between 0 and 1.

(7)

7. Prove that the expected value of the binomial distribution with n draws and probability p is equal to np and its variance is $np(1-p)$.

Expected Value of the binomial distribution

E(X) = $\sum_{i=1}^{n} x_i * P(x_i)$ where x_i is the event and $P(x_i)$ is the probability of the event occuring

X = k , P(X) = $\binom{n}{k} p^k q^{n-k}$ where k = the number of successes needed, p = probability of success, q = 1-p = probability of failure and n=number of draws

Then $E(X) = \sum_{k=0}^{n} k * \binom{n}{k} p^k q^{n-k}$

When k=0 the whole terms is 0 so we have

$E(X) = \sum_{k=1}^{n} k * \binom{n}{k} p^k q^{n-k}$

Note that $k * \binom{n}{k} = n * \binom{n-1}{k-1}$

Side proof:

$$n * \binom{n-1}{k-1} = \frac{n*(n-1)!}{(k-1)!*[(n-1)-(k-1)]!} = \frac{n!}{(k-1)!(n-k)!} = \frac{k*n!}{k!(n-k)!} = k * \binom{n}{k}$$

So now we have

$$E(X) = \sum_{k=0}^{n} k * \binom{n}{k} p^k q^{n-k} = \sum_{k=1}^{n} n * \binom{n-1}{k-1} p^k q^{n-k}$$

From here we can pull out a p and bring the n to the front

$$\sum_{k=1}^{n} n * \binom{n-1}{k-1} p^k q^{n-k} = np \sum_{k=1}^{n} \binom{n-1}{k-1} p^{k-1} q^{n-k}$$

Replace n-k with (n-1)-(k-1) since it's equal

$$np \sum_{k=1}^{n} \binom{n-1}{k-1} p^{k-1} q^{(n-1)-(k-1)}$$

Let m = n-1 and l = k – 1

$$np \sum_{l=0}^{m} \binom{m}{l} p^l q^{m-l}$$

$\sum_{l=0}^{m} \binom{m}{l} p^l q^{m-l} = 1$ because it's the summation of all probabilities

So E(X) = np*1 = np for the binomial distribution.

Variance is on the next page

Variance of the binomial distribution

Var(X) = $E[(X-\mu)^2]$ where $\mu = E(X)$

So Var(X) = $E[(X-E[X]^2]$ = $E[X^2 - 2XE[X] + E[X]^2]$

The E[E[X]] = E[X] so we have $E[X^2] - 2E[X]^2 + E[X]^2 = E[X^2] - E[X]^2$

We know that E[X] = np so $E[X]^2 = (np)^2 = n^2p^2$

Now we just need to solve $E[X^2]$

$$E[X^2] = \sum_{k=0}^{n} k^2 * \binom{n}{k} p^k q^{n-k} = \sum_{k=0}^{n} kn * \binom{n-1}{k-1} p^k q^{n-k} = np \sum_{k=1}^{n} k * \binom{n-1}{k-1} p^{k-1} q^{(n-1)-(k-1)}$$

Not everything is explained above but it's using the same logic as we did when calculating the expected value of the binomial distribution.

Again using a similar approach, we let m = n-1 and l = k – 1 and we get

$E[X^2] = np \sum_{l=0}^{m} (l+1) * \binom{m}{l} p^l q^{m-l}$ which we can then breakout the $(l+1)$ part so

$E[X^2] = np(\sum_{l=0}^{m} l * \binom{m}{l} p^l q^{m-l} + \sum_{l=0}^{m} \binom{m}{l} p^l q^{m-l})$

We know that $\sum_{l=0}^{m} \binom{m}{l} p^l q^{m-l}) = 1$ since it is just the sum of the probabilities over all possible events in S

For $\sum_{l=0}^{m} l * \binom{m}{l} p^l q^{m-l}$ we can use the same technique of $k * \binom{n}{k} = n * \binom{n-1}{k-1}$ from before and get

$$\sum_{l=0}^{m} l * \binom{m}{l} p^l q^{m-l} = \sum_{l=0}^{m} m * \binom{m-1}{l-1} p^l q^{m-l} = mp \sum_{l=1}^{m} \binom{m-1}{l-1} p^{l-1} q^{(m-1)-(l-1)} = (n-1)p$$

Substituting back in we get

$E[X^2] = np[(n-1)p + 1] = np[np - p + 1] = n^2p^2 - np^2 + np$

Finally we have

Var(X) = $E[X^2] - E[X]^2 = n^2p^2 - np^2 + np - n^2p^2 = np - np^2 = np(1-p)$

8. Assume there is a set J of jobs and a set C of CPU cores such that $|C| = |J|$. The time required for a job $j \in J$ to be processed on CPU $c \in C$ is $t_{j,c}$.

(a) Write an integer program to assign each job to a CPU which minimizes the total amount of time required to finish all the jobs. Note that each CPU must receive exactly one job.

(b) Give an example with two jobs and two CPU cores for which the LP relaxation of your program has a lower objective value than the optimal integer one.

Part a

Let J be the set of all jobs such that J = $\{j_1, j_2, \ldots\}$, $j_i\ \forall\ i \in \mathbb{N}$

Let C be the set of all CPUs such that C = $\{c_1, c_2, \ldots\}$, $c_i\ \forall\ i \in \mathbb{N}$

Let X be assigning a job to a CPU such that X = $\{x_{j,c}\ \forall\ j, c \in \mathbb{N}\}$

And $0 \le x_{j,c} \le 1$, and $x_{j,c} = \begin{cases} 1, when\ j\ job\ is\ assigned\ to\ c\ cpu \\ 0, when\ j\ is\ not\ assigned\ to\ c\ cpu \end{cases}$

So for example $x_{1,1} = 1$ means we assigned job 1 to cpu 1

$x_{1,1} = 0$ means we did not assign job 1 to cpu 1

Let T be the set of times each job takes to run on it's cpu such that T = $\{t_{j,c}\ \forall\ j, c \in \mathbb{N}\}$

Then the total time taken to run is

$Total\ Time = \sum_{j,c \in \mathbb{N}} t_{j,c} * x_{j,c}$

So we want to minimize the $Total\ Time = \sum_{j,c \in \mathbb{N}} t_{j,c} * x_{j,c}$

We have constraints

$\sum_c x_{j,c} = 1$ for all j ∈ J since all jobs must be assigned to exactly one CPU

$\sum_j x_{j,c} = 1$ for all c ∈ C since all CPU's may only receive exactly one job

$x_{j,c} \in \{0,1\}$ for all j ∈ J and c ∈ C

Part b

There doesn't exist a LP relaxation with a lower objective value than the optimal integer one.

(10)

9. The bisection of a graph of a graph is defined as the smallest number of crossing edges when dividing the vertices of the graph into two sets of equal size (there is no connectivity requirements for the sets). Write an integer program that computes the bisection of a graph with even number of vertices.

Let V(G) = $\{v_1, v_2, \ldots v_n\}$ be all existing n vertices of graph G. Note that n is an even number.

Let E(G) = $\{e_{a,b}$ such that $a \in V(G), b \in V(G)$ and $e_{a,b}$ *is the edge between* v_a *and* v_b

Let $e_{a,b} = \begin{cases} 1, \textit{edge exists} \\ 0, \textit{edge does not exist} \end{cases}$

Let Q(G) = $\{q_{a,b}$ such that $a \in V(G), b \in V(G)$

Let $q_{a,b} = \begin{cases} 1, \textit{bisection crossed } e_{a,b} \\ 0, \textit{bisection did not cross } e_{a.b} \end{cases}$

We want to make two sets V_1 *and* V_2 such that $|V_1| = |V_2|$ (the number of elements in each is equal) and $V_1 \subseteq V(G), V_2 \subseteq V(G)$

We want to make V_1 and V_2 such that we minimize $\sum_{\forall a,b \in V(G)} e_{a,b} * q_{a,b}$