MIT Primes 2020 General Math Problem Set

Matthew Ding

1 Problem G1

1.1 Part A

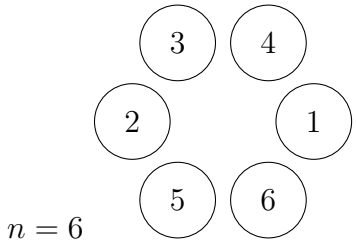
There is no odd $n \geq 5$ that satisfies the condition:

> Proof by Contradiction: Let us assume that n is odd and $n \geq 5$. The sum of any two distinct numbers within {1,2,...,n} is at least 3. Therefore, if all consecutive number sums are prime numbers, they must all be odd.
>
> If each consecutive number sum is odd, then each pair of consecutive numbers must contain exactly one odd number and one even number. Therefore, if we consider every single number in every single pair of consecutive numbers, we can see that there are an equal number of odd and even numbers.
>
> We also note that considering every number in every consecutive pair counts each number in the circle exactly twice. Thus we conclude that in the set of all numbers around the circle, there are still an equal number of odd and even numbers.
>
> If n is odd however, within the set {1,2,...,n}, there are $\frac{n+1}{2}$ odd numbers and $\frac{n-1}{2}$ even numbers. These two values are not equal, so we have arrived at a contradiction. Thus, our original assumption that n is odd and $n \geq 5$ is false.

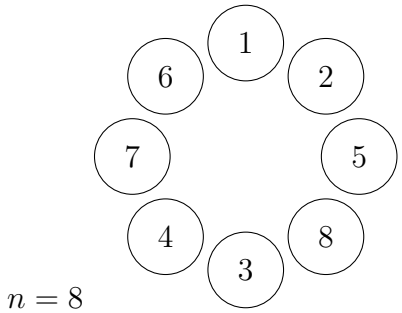
Therefore there is no odd $n \geq 5$ that produces an arrangement of numbers to satisfy the condition.

1.2 Part B

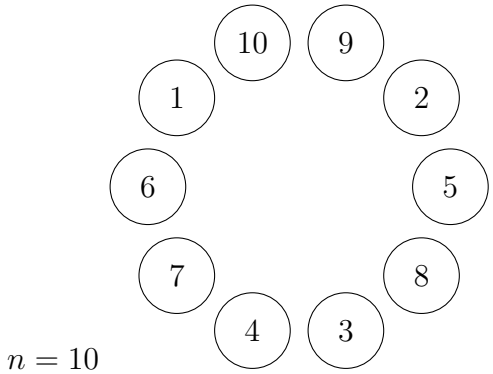
For each of $n = 6$, $n = 8$, and $n = 10$, there exists a circle of numbers 1 through n for which any two consecutive numbers sum to a prime number.



$n = 6$



$n = 8$



$n = 10$

2 Problem G2

2.1 Part A

$$\begin{aligned} q &= \sqrt{a+\sqrt{b}}+\sqrt{a-\sqrt{b}} \\ &= \sqrt{(\sqrt{a+\sqrt{b}}+\sqrt{a-\sqrt{b}})^2} \\ &= \sqrt{a+\sqrt{b}+a-\sqrt{b}+2\sqrt{a^2-b}} \\ &= \sqrt{2a+2\sqrt{a^2-b}} \end{aligned}$$

For $(a, b) = (17943, 321185624)$, we have:

$$\begin{aligned} q &= \sqrt{2a + 2\sqrt{a^2 - b}} \\ &= \sqrt{2(17943) + 2\sqrt{17943^2 - 321185624}} \\ &= \sqrt{35886 + 2\sqrt{765625}} \\ &= \sqrt{35886 + 1750} \\ &= \sqrt{37636} \\ &= 194 \end{aligned}$$

Therefore $q = 194$, which is a rational number.

2.2 Part B

By squaring q, we get:

$$q^2 = 2a + 2\sqrt{a^2 - b} \tag{1}$$

Since $2\sqrt{a^2 - b}$ is non-negative, we see that $q^2 \geq 2a$, or $a \leq \frac{q^2}{2}$
Then solving for b, we get:

$$\begin{aligned} b &= a^2 - (\frac{q^2 - 2a}{2})^2 \\ &= (a + \frac{q^2 - 2a}{2})(a - \frac{q^2 - 2a}{2}) \\ &= (\frac{q^2}{2})(2a - \frac{q^2}{2}) \\ &= q^2(a - \frac{q^2}{4}) \end{aligned}$$

We see that in order for b to be an positive integer, $a - \frac{q^2}{4}$ must be an positive integer as well. Therefore q must be an even number, and $a > \frac{q^2}{4}$. We also want b to not be a square number. Since q^2 is a square number, $q^2(a - \frac{q^2}{4})$ will not be a square number if $a - \frac{q^2}{4}$ is not a square number.

To find some value (a, b), we set $q = 4$. Now we want to find some $\frac{q^2}{4} < a \leq \frac{q^2}{2}$, or $4 < a \leq 8$.

1. Testing $a = 5$ gives us $a - \frac{q^2}{4} = 1$, which does not work as 1 is a square number.

2. Next, we try $a = 6$, which gives us $a - \frac{q^2}{4} = 2$, which is not a square number. Therefore $a = 6$ is valid.

Solving for b, we get:

$$b = q^2(a - \frac{q^2}{4}) = 4^2(2) = 32$$

Therefore we find the ordered pair $(a, b) = (6, 32)$. To double check, we can plug our ordered pair back into (1):

$$q^2 = 2a + 2\sqrt{a^2 - b} = 2(6) + 2\sqrt{6^2 - 32} = 12 + 2(2) = 16$$

We find that $q = 4$, a rational number, so the ordered pair $(6, 32)$ satisfies our condition.

2.3 Part C

Yes, there are infinitely many pairs (a, b) such that q is rational. We have $q^2 = 2a + 2\sqrt{a^2 - b}$. From Part B, it is known that $\frac{q^2}{4} < a \leq \frac{q^2}{2}$. We choose the value $a = \frac{q^2}{2} - 1$, as it provides a simplified relation between a and b, and it satisfies the inequality for $q > 2$. We now have the equation:

$$q^2 = 2(\frac{q^2}{2} - 1) + 2\sqrt{(\frac{q^2}{2} - 1)^2 - b}$$

Solving for b gives us:

$$b = \frac{q^4}{4} - q^2 = (\frac{q^2}{2} - 1)^2 - 1 = a^2 - 1 \tag{2}$$

We see that if $q = 2$, then $b = 0$, which is invalid, so $q \neq 2$.
Now if we set q to equal any even integer greater than or equal to 4:

1. a is a positive integer: Since q is even, q^2 is a multiple of 4, and $\frac{q^2}{2} - 1$ is an integer. Therefore a is an integer greater than or equal to $\frac{4^2}{2} - 1 = 7$.

2. b is a positive integer: From (2), $b = a^2 - 1$. Since a is a positive integer ≥ 7, b is a positive integer. We also see here that $b < a^2$.

3. Since $b = a^2 - 1$ and a^2 is a perfect square, b is *not* a perfect square.

We see that for any even integer q such that $q \geq 4$, we can construct at least one unique pair (a, b), where q, a, b are all positive integers, $b < a^2$, and b is not a perfect square. Since there are an infinite number of even integers greater than or equal to 4, there are infinitely many pairs (a, b) such that q is rational.

2.4 Part D

It is impossible to find (a, b) such that q is rational but not an integer.

To begin, we note that the square root of a positive integer is always either an integer or irrational (Proof can be found at [2]). We also note that the square root of a positive irrational number is irrational:

> Proof by Contradiction: Let us assume that an irrational number q has a rational square root. We denote $\sqrt{q} = \frac{a}{b}$, where a, b are integers. We can see that $q = \frac{a^2}{b^2}$. Since a, b are integers, then a^2, b^2 must be integers as well. q is then a ratio between two integers, so it is rational. We arrive at a contradiction, so our original assumption that q has a rational square root is false.

We now prove that q is an integer given that q is rational. We have $q = \sqrt{2a + 2\sqrt{a^2 - b}}$. Let us consider the expression $\sqrt{a^2 - b}$, which can either be an integer or irrational:

1. If $\sqrt{a^2 - b}$ is irrational, since $2a$ is an integer, then $2a + 2\sqrt{a^2 - b}$ is irrational as well. Since the square root of an irrational number is irrational, we have $\sqrt{2a + 2\sqrt{a^2 - b}}$ is irrational, and q is therefore irrational. Since it is given that q is a rational number, we arrive at a contradiction. Therefore $\sqrt{a^2 - b}$ must be an integer.

2. Because $\sqrt{a^2 - b}$ is an integer, since $2a$ is an integer, then $2a+2\sqrt{a^2 - b}$ is an integer as well. Once again, $\sqrt{2a + 2\sqrt{a^2 - b}}$, or q, can either be irrational or an integer. Since we are given that q is rational, we know that q must be an integer.

Therefore if q is rational, q must be an integer.

3 Problem G3

3.1 Part A

We first attempt to define the three points of the triangle in terms of the three slopes. We define the triangle to have three arbitrary points, (x_1, x_1^2), (x_2, x_2^2), and (x_3, x_3^2). We denote the slope of the line opposite of (x_1, x_1^2) to be m_1, and define m_2 and m_3 similarly.

We then see that

$$m_1 = \frac{x_2^2 - x_3^2}{x_2 - x_3} = x_2 + x_3 \tag{3}$$

$$m_2 = \frac{x_1^2 - x_3^2}{x_1 - x_3} = x_1 + x_3 \tag{4}$$

$$m_3 = \frac{x_1^2 - x_2^2}{x_1 - x_2} = x_1 + x_2 \tag{5}$$

By summing the three equations and dividing by 2, we get:

$$x_1 + x_2 + x_3 = \frac{m_1 + m_2 + m_3}{2}$$

By separately subtracting equations (3), (4), (5) from the above equation, we get:

$$x_1 = (x_1+x_2+x_3)-(x_2+x_3) = \frac{m_1 + m_2 + m_3}{2} - m_1 = \frac{-m_1 + m_2 + m_3}{2} \tag{6}$$

$$x_2 = (x_1+x_2+x_3)-(x_1+x_3) = \frac{m_1 + m_2 + m_3}{2} - m_2 = \frac{m_1 - m_2 + m_3}{2} \tag{7}$$

$$x_3 = (x_1+x_2+x_3)-(x_1+x_2) = \frac{m_1 + m_2 + m_3}{2} - m_3 = \frac{m_1 + m_2 - m_3}{2} \tag{8}$$

Using the Shoelace Formula to find the area of a triangle, we get:

$$\frac{1}{2}|(x_1 - x_3)(y_2 - y_1) - (x_1 - x_2)(y_3 - y_1)|$$

Given that $y_1 = (x_1)^2$, $y_2 = (x_2)^2$, $y_3 = (x_3)^2$, we can expand the equation to:

$$\frac{1}{2}|(x_1 - x_3)(x_2 - x_1)(x_2 + x_1) - (x_1 - x_2)(x_3 - x_1)(x_3 + x_1)|$$

Using the values for the slopes and x-coordinates obtained from equations (3) to (8), we can get:

$$\frac{1}{2}|(\frac{-m_1 + m_2 + m_3}{2} - \frac{m_1 + m_2 - m_3}{2})(\frac{m_1 - m_2 + m_3}{2} - \frac{-m_1 + m_2 + m_3}{2})(m_3))-$$
$$(\frac{-m_1 + m_2 + m_3}{2} - \frac{m_1 - m_2 + m_3}{2})(\frac{m_1 + m_2 - m_3}{2} - \frac{-m_1 + m_2 + m_3}{2})(m_2)|$$
$$= \frac{1}{2}|(m_3 - m_1)(m_1 - m_2)(m_3) - (m_2 - m_1)(m_1 - m_3)(m_2)|$$
$$= \frac{1}{2}|(m_3 - m_1)(m_1 - m_2)(m_3 - m_2)|$$

We can multiply the term inside the absolute value and rearrange to get our final equation for the area of the triangle:

$$\frac{1}{2}|(m_1 - m_2)(m_2 - m_3)(m_3 - m_1)|$$

3.2 Part B

We first attempt to find an possible upper bound on the area. Let $a = m_1 - m_2$, $b = m_2 - m_3$, and $c = m_3 - m_1$. To find the upper bound, we find the maximum of $\frac{1}{2}|abc|$, where $-2m \leq a, b, c \leq 2m$ and $a + b + c = 0$. Knowing that $a, b, c \neq 0$, which must be true for the triangle to exist, there must be at least one those variables less than 0 and one of those variables greater than 0 for the three of them to sum to 0. From this, we can deduce that if the value of $\frac{1}{2}|abc|$ is maximized, one of a, b, or c must equal $\pm 2m$.

> Proof by Contradiction: Assume that the above claim is false, and that none of a, b, or c is equal to $\pm 2m$ in the set of a, b, c that produces the maximum triangle area. Assume WLOG that a is the number greater than 0 and b is the number less than 0. We can then create a new set of values for $(a, b, c) \rightarrow (a', b', c)$, where $a' = a + k$, $b' = b - k$, and k is an arbitrary positive number such that $a + k \leq 2m$ and $b - k \geq -2m$. Since $a < 2m$ and $b > -2m$, there will always exist a positive number k that satisfies those conditions. Since a' is positive and b' is negative, $|a'b'| > |ab|$. Thus $\frac{1}{2}|a'b'c| > \frac{1}{2}|abc|$.
>
> We therefore conclude that $\frac{1}{2}|abc|$ is not the upper bound on triangle area, a contradiction. Our original assumption that none of a, b, or c is equal to $\pm 2m$ is false.

Assume WLOG that $a = \pm 2m$, let us consider the case where $a = 2m$. We get $b + c = -2m$, and wish the maximize the value of $\frac{1}{2}|abc|$. Since both b and c are greater than $-2m$, we see that both of them must be non-positive. Therefore abc is non-negative. The value of bc is maximized when $b = c = -m$, which maximizes $\frac{1}{2}|abc|$ as well.

If $a = -2m$, we can similarly find that the values $a = -2m$ and $b = c = m$ creates an equivalent maximum triangle area as well. Therefore, we find an upper bound on triangle area is $\frac{1}{2}|2m * -m * -m| = m^3$.

Now that we have found an upper-bound for area of m^3, we now seek a triangle with area equal to this upper-bound. For $a = m_1 - m_2 = 2m$, $b = m_2 - m_3 = -m$, and $c = m_3 - m_1 = -m$, the values $m_1 = m$, $m_2 = -m$, and $m_3 = 0$ fulfil the conditions. We can also see that a triangle with points $(0, 0)$, $(-m, m^2)$, and (m, m^2) exists with slopes m, 0, and $-m$. Since a triangle always exists that meets our defined upper-bound, we conclude that the maximum area for the triangle is always m^3.

4 Problem G4

We define the entries of matrix M as follows:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & k & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix}$$

Since the four continuous 2x2 submatrices have determinant of 1, we get the following set of equations:

$$\begin{vmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & k \end{vmatrix} = x_{1,1} * k - x_{1,2} * x_{2,1} = 1$$

$$\begin{vmatrix} x_{1,2} & x_{1,3} \\ k & x_{2,3} \end{vmatrix} = x_{1,2} * x_{2,3} - x_{1,3} * k = 1$$

$$\begin{vmatrix} x_{2,1} & k \\ x_{3,1} & x_{3,2} \end{vmatrix} = x_{2,1} * x_{3,2} - k * x_{3,1} = 1$$

$$\begin{vmatrix} k & x_{2,3} \\ x_{3,2} & x_{3,3} \end{vmatrix} = k * x_{3,3} - x_{2,3} * x_{3,2} = 1$$

We solve for the corner values of our matrix. We can divide by k, as it is known that k is positive, so $k \neq 0$.

$$\begin{aligned} x_{1,1} &= \frac{x_{1,2} * x_{2,1} + 1}{k} \\ x_{1,3} &= \frac{x_{1,2} * x_{2,3} - 1}{k} \\ x_{3,3} &= \frac{x_{2,3} * x_{3,2} + 1}{k} \\ x_{3,1} &= \frac{x_{2,1} * x_{3,2} - 1}{k} \end{aligned}$$

Solving for the determinant of M, we get:

$$\begin{aligned}
\det M &= x_{1,1}\begin{vmatrix} k & x_{2,3} \\ x_{3,2} & x_{3,3} \end{vmatrix} - x_{1,2}\begin{vmatrix} x_{2,1} & x_{2,3} \\ x_{3,1} & x_{3,3} \end{vmatrix} + x_{1,3}\begin{vmatrix} x_{2,1} & k \\ x_{3,1} & x_{3,2} \end{vmatrix} \\
&= x_{1,1} - x_{1,2}(x_{2,1} * x_{3,3} - x_{2,3} * x_{3,1}) + x_{1,3} \\
&= \frac{x_{1,2} * x_{2,1} + 1}{k} - x_{1,2}(x_{2,1} * x_{3,3} - x_{2,3} * x_{3,1}) + \frac{x_{1,2} * x_{2,3} - 1}{k} \\
&= \frac{x_{1,2} * x_{2,1} + 1}{k} - x_{1,2}(x_{2,1} * \frac{x_{2,3} * x_{3,2} + 1}{k} - x_{2,3} * \frac{x_{2,1} * x_{3,2} - 1}{k}) + \frac{x_{1,2} * x_{2,3} - 1}{k} \\
&= \frac{x_{1,2} * x_{2,1} + x_{1,2} * x_{2,3}}{k} - x_{1,2}(\frac{x_{2,1} * x_{2,3} * x_{3,2} + x_{2,1} - x_{2,1} * x_{2,3} * x_{3,2} + x_{2,3}}{k}) \\
&= \frac{x_{1,2} * x_{2,1} + x_{1,2} * x_{2,3}}{k} - x_{1,2}(\frac{x_{2,1} + x_{2,3}}{k}) \\
&= 0
\end{aligned}$$

Therefore the determinant of M is 0.

5 Problem G5

5.1 Part A

We can construct any subset of S with equal probability in n independent steps in sequence. On step i, where $1 <= i <= n$, we choose whether or not integer element i will be in the subset. Both choices (i is in or not in the subset) have equal probabilities of 1/2 of occurring.

Consider a sequence of 3 such simultaneous constructions of subsets A, B, C. Once again, there are a total of n steps in this sequence, where at each step i we determine whether the integer element i will be in any combination of subsets A, B, or C. Out of the 8 total combinations of $\{A, B, C\}$ that i could be in, only one of the will increase the value of $|A \cap B \cap C|$ by 1, the case where element i exists in all three sets. The rest of the 7 remaining combinations will not change the value of $|A \cap B \cap C|$, as element i does not exist in all three sets. Each of the 8 combinations are equally likely to occur, with probability 1/8.

We can think of the value $|A \cap B \cap C|$ as the sum of n independent discrete random variables X_i for $1 <= i <= n$, representing whether that element i is in all three subsets A, B, C. X_i has a value of 1 if i is in all three subsets, and a value of 0 if it is not in all three subsets. We then see that:

$$|A \cap B \cap C| = X_1 + X_2 + X_3 + ... + X_n$$

The expected value of any X_i is $1 * \frac{1}{8} + 0 * \frac{7}{8} = \frac{1}{8}$. The expected value of $|A \cap B \cap C|$ is the sum of the expected values of each X_i for $1 <= i <= n$. $n * \frac{1}{8} = \frac{n}{8}$, so the expected value of $|A \cap B \cap C|$ is $\frac{n}{8}$.

5.2 Part B

$|A \cap B|$ and $|B \cap C|$ are no longer independent random variables, so we cannot merely multiply their expected values to find the answer.

Let us consider an arbitrary set of B, and call it B'. The elements not in set B' can not be in either $A \cap B'$ or $B' \cap C$, so we do not consider them in our calculation. **Considering only the elements that are in set B'**, we want to find $|A \cap B'| * |B' \cap C|$. Given a set B', we denote the value of $|A \cap B'|$ to be $|A'|$, and the value of $|B' \cap C|$ to be $|C'|$. Given a specific set B', the value of $|A'|$ and $|C'|$ are independent.

Our problem now becomes finding the expected value of $|A'| * |C'|$, given the set B'. Since $|A'|$ and $|C'|$ are now independent random variables, we can multiply their expected values to find the expected value of $|A'| * |C'|$. An element in B' has a 50% chance of being in set A' and a 50% chance of being in set C'. Since we are considering a total of $|B'|$ elements, the expected value of $|A'|$ and $|C'|$, given the set B', are both $\frac{|B'|}{2}$. We find:

$$E(|A'| * |C'|) \; given \; B' = \frac{|B'|}{2} * \frac{|B'|}{2} = \frac{|B'|^2}{4}$$

The original problem becomes finding $\mathrm{E}(\frac{|B'|^2}{4})$ over the distribution of B', which is equivalent to $\mathrm{E}(\frac{|B|^2}{4})$. $|B|$ is a binomial random variable, where n is equal to $|S|$, and p is equal to the chance any specific element is in B, which is 0.5. The expectation of the square of a binomial random variable is $n^2p^2 + np(1-p)$ [6]. We thus see that:

$$|B|^2 = n^2p^2 + np(1-p) = n^2(0.5)^2 + n(0.5)(0.5) = \frac{n^2+n}{4}$$

Solving for our expected value, we get:

$$E(\frac{|B|^2}{4}) = \frac{n^2+n}{16}$$

We find the expected value of $|A \cap B| * |B \cap C|$ to be $\frac{n^2+n}{16}$.

6 Problem 6

Let us define p as the probability that a robot starting at 0 ever reaches 1. We note that the movement of the robot is independent of the start location.

The robot will move the same distances with the same probability regardless of what number it is currently on. We can therefore deduce that the start location of 0 is arbitrary, and p actually defines the probability that a robot starting at *any* integer n ever reaches $n+1$.

We also note that the probability that a robot starting at n ever reaches $n+2$ is p^2. This event occurs if and only if the robot ever reaches $n+1$ from n, and then reaches $n+2$ from $n+1$. These are independent random events, so we find the probability of it occurring by multiplying their respective probabilities: $p * p = p^2$. **In general, the probability that the robot reaches $n+k$ from n is p^k.**

We can calculate the probability of p by considering each of the three cases where a robot, starting at 0, can travel in its first step, assuming that the robot does indeed reach 1. In this situation $n = 0$. The three possible choices are as follows:

1. The robot goes to $n+1$ with probability 1/2. The robot is already at 1, so the probability the robot reaches 1 after moving to $n+1$ on its first step is 1/2.

2. The robot goes to $n-1$ with probability 1/3. The robot is now at -1, so it must move up two points on the number line to reach 1. The probability of this occurring is p^2, so the probability that the robot reaches 1 after moving to $n-1$ on its first step is $\frac{1}{3}p^2$.

3. The robot goes to $n-2$ with probability 1/6. The robot is now at -2, so it must move up three points on the number line to reach 1. The probability of this occurring is p^3, so the probability that the robot reaches 1 after moving to $n-2$ on its first step is $\frac{1}{6}p^3$.

Since the robot only has those three options to move on its first step, the probability the robot ever reaches 1 is equal to the probability that the robot ever reaches 1 given that the robot moves to $n+1$, $n-1$, or $n-2$ on its first move. We then get the equation:

$$p = \frac{1}{2} + \frac{1}{3}p^2 + \frac{1}{6}p^3.$$

Setting the variables to one side and factoring, we obtain the equation:

$$\frac{1}{6}(p-1)(p^2+3p-3) = 0$$

We are given that $p \neq 1$, so we take the positive zero of p^2+3p-3, getting $p = \frac{\sqrt{21}}{2} - \frac{3}{2}$.

The probability of the robot ever reaching 1 after starting from 0 on the number line is therefore $\frac{\sqrt{21}}{2} - \frac{3}{2}$, or approximately 0.791.

7 Problem 7

NOTE: I have written my description of the algorithm under "standard" chess rules for rook capturing, i.e. a rook cannot capture another rook in the same row or column if there is a third rook between the two.

However, my algorithms apply to both standard chess rules and the abridged chess rules described in the problem. This is because the calculations for maximum and minimum bounds for the number of possible captures apply under both rule sets.

7.1 Part A

I will start off by defining a couple of terms:

First, if rook A captures rook B, the board state remains identical with the single change of the position of rook A now becomes empty. I will refer to this move as "removing" rook A, even though rook B is technically the rook being "removed". Therefore we can remove any rook that is attacking another as a legal move.

Next, we define an "isolated group" of rooks to be any set of rooks such that each member of the set is attacking at least one other member of the set and no rooks that are not in the set. Also, any isolated group will remain isolated throughout the entire algorithm. This is because if a group is isolated, no rooks outside the set are on the same row or column as any member in the set. If we think of moves as "removals", we can see that with any amount of moves, no rook can be added to the same row or column as a rook within the set.

We begin the algorithm by constructing a separate graph for each isolated group of rooks on the board. Adjacent nodes in this graph are defined as rooks that are attacking each other. We do this by constructing adjacency lists for each rook, as well as a "visited" set that will store every location that has already been visited in our search. We then use a Flood-Fill-like algorithm starting from a rook to find the entire isolated group it is a part of:

1. Find $rookCounter$, the total number of rooks, by searching the entire board in $O(n^2)$ time
2. We begin by running ChessFloodFill on an arbitrary square with a rook
3. Whenever the stack of ChessFloodFill calls is emptied, we append $groupCounter$ by 1.

4. We then call ChessFloodFill at another arbitrary square that contains a rook and is not in *visitedSet*.

5. We repeat this process until all squares with rooks are visited. At this point, the value of *groupCounter* will be equal to the number of isolated groups.

Algorithm 1 Finds a single isolated group on the board

```
1: visitedSet → set of all squares with rooks that we have visited
2: procedure chessFloodFill(ROW r, COLUMN c)
3:     Add (R, C) to visitedSet
4:     currentRow = R + 1
5:     currentColumn = C
6:     while currentRow <= n do
7:         if (currentRow, currentColumn) contains Rook not in visitedSet then
8:             add (currentRow, currentColumn) and (R, C) to each other's adjacency lists
9:             ChessFloodFill(currentRow, currentColumn)
10:            break While loop
11:        add 1 to currentRow
12:    currentRow = R − 1
13:    currentColumn = C
14:    while currentRow >= 0 do
15:        if (currentRow, currentColumn) contains Rook not in visitedSet then
16:            add (currentRow, currentColumn) and (R, C) to each other's adjacency lists
17:            ChessFloodFill(currentRow, currentColumn)
18:            break While loop
19:        subtract 1 to currentRow
20:    currentRow = R
21:    currentColumn = C + 1
22:    while currentColumn <= n do
23:        if (currentRow, currentColumn) contains Rook not in visitedSet then
24:            add (currentRow, currentColumn) and (R, C) to each other's adjacency list
25:            ChessFloodFill(currentRow, currentColumn)
```

```
            break While loop
        add 1 to currentColumn
    currentRow = R
    currentColumn = C − 1
    while currentColumn >= n do
        if (currentRow, currentColumn) contains Rook not in visitedSet then
            add (currentRow, currentColumn) and (R, C) to each other's adjacency lists
            ChessFloodFill(currentRow, currentColumn)
            break While loop
        subtract currentColumn by 1
```

The visited set ensures that we only check each square a constant number of times. Since there are $O(n^2)$ squares on the board, the run-time for this search is $O(n^2)$. The maximum number of captures is:

$$rookCounter - groupCounter$$

Proof of Correctness: Within every single isolated group of rooks, we can construct a sequence of captures such that only 1 rook remains. For each of these isolated groups, if we conduct this sequence of captures, the board will not be peaceful until every single original isolated group has been reduced to size 1. Since each move reduces the number of rooks by exactly 1, the number of moves will be the initial number of rooks minus the number of rooks left on the board, which is equal to the initial number of rooks minus the number of isolated groups.

For each isolated group, we can construct our sequence of captures as follows:

1. The isolated group is its own isolated graph. Starting from any node, we run a Breadth-First Search Algorithm [5]. This algorithm iterates through all nodes and determines the length of the shortest path from our start node to every other node. It also stores the parent pointer for each node, which is the node visited directly before in the shortest path.

2. From the lengths of the shortest path and each node's parent pointer, we can construct a tree, where any path from the root to a node in the tree represents a shortest path from the root to that node in our graph.

3. The sequence of removals will be to always remove the node of the greatest depth, until only the parent node remains. To remove a node, we make that rook capture its parent node. Since we always removing nodes of the greatest depth, we will only be removing leaves of the tree. Since leaves do not have children, we never remove a parent node in this algorithm. This ensures that every node we remove will always have a parent node, and that a sequence of captures always exists that can remove every rook in the tree besides the root.

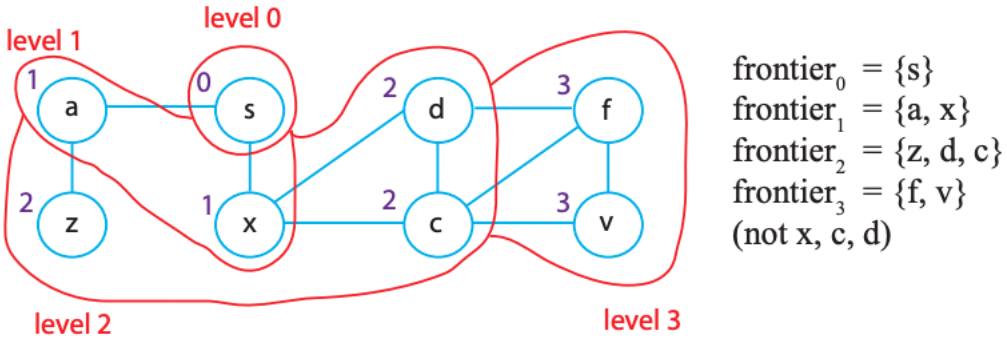


Figure 4: Breadth-First Search Frontier

Picture from [5] denotes how a BFS search assigns a level to each node. In our case the graph cannot have cycles of 3 as pictured, but it gives a relative idea of how the search will be done.

7.2 Part B

To find the minimum number of moves to achieve a peaceful configuration, we first find the greatest sized peaceful subset of rooks of the initial configuration, because the greatest amount of rooks on the board implies the least amount of captures. Once we have found this peaceful configuration, we prove that it is always achievable in a sequence of captures.

First, to find the peaceful configuration with the greatest number of rooks from our initial configuration, we present any configuration of rooks on the board as a bipartite graph, with two sets of n vertices each. One set of n vertices represents the n rows of the board, while the other set represents the n columns of the board. Each edge in the bipartite graph represents a rook at some position on the board. There is a bijection between edge configuration and a square on the board: each square on the board has a unique pair of

row number and column number, and each edge on the graph is connected to exactly 1 unique vertex in the row set and 1 unique vertex in the column set.

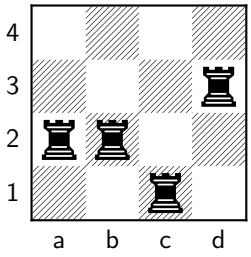


Figure 1: This is a possible chessboard configuration for $n = 4$

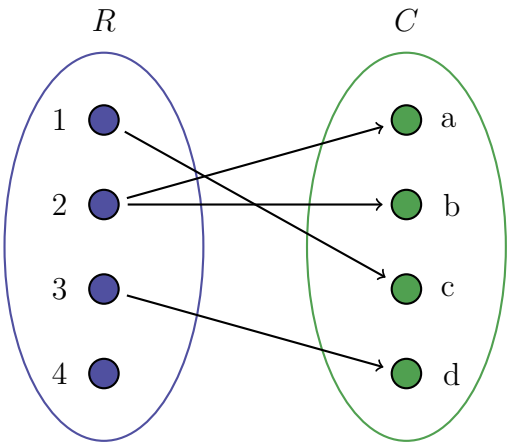


Figure 2: This is the unique bipartite graph that represents the chessboard in Figure 1

To find the greatest number of rooks in a peaceful configuration given our start state, we solve the Maximum Bipartite Matching Problem[3] for the bipartite graph that represents our initial board state. A matching is defined as a configuration of edges in a bipartite graph such that no two edges share an endpoint. Translating this to a board state, a matching is where no two rooks exist in the same row or column, the definition of a peaceful configuration. The Maximum Bipartite Matching is a matching with the greatest number of edges. Translating this to a board state, it is the peaceful configuration with the greatest number of rooks from our initial state, the problem we want to solve.

We can solve the Maximum Bipartite Matching Problem using the Ford-Fulkerson Algorithm. We do this by creating a flow network, as described in [3]:

1. Add a source node and add edges from the source node to every row node/vertex.

2. Add a sink node and add edges from the sink node to every column node/vertex. Edges that represent rooks are directed from a row vertex to a column vertex.

3. Set all edge capacities to 1.

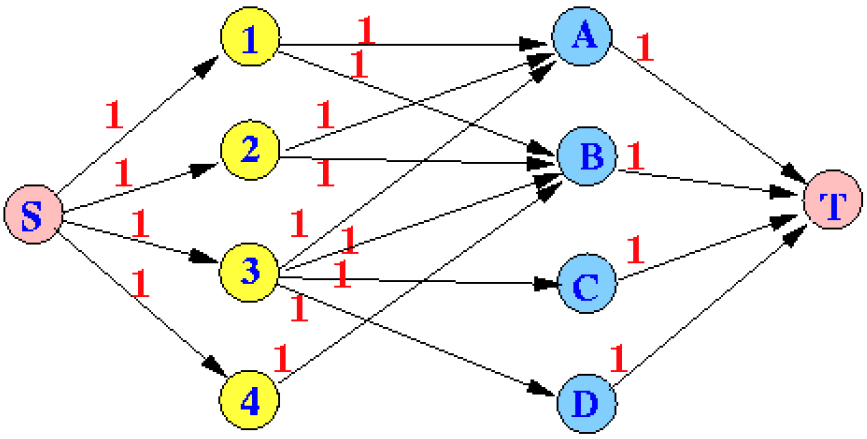


Figure 3: The diagram from [1] shows how we set up our bipartite graph for the Ford-Fulkerson algorithm

The Ford-Fulkerson algorithm will output the maximum flow, or in our case, the maximum number of edges in a bipartite matching. The lower bound on the minimum number of moves to create a peaceful configuration is thus (# of rooks in initial state) - (maximum number of edges in bipartite matching). The run-time for Ford-Fulkerson is $O(mn)$, where m is the number of edges between the row and column vertices, and n is the size of our row/column sets[4]. We have a maximum of n^2 rooks, or edges, and the size of both the row and column sets are n. Therefore, the run-time complexity of our entire algorithm is $O(mn) = O(n^2 * n) = O(n^3)$.

Finally, we prove the correctness of our algorithm by showing that any maximum-sized peaceful configuration is obtainable from our initial configuration through a sequence of captures:

First, we consider the set of all rooks in the maximum sized peaceful configuration, calling is S. We claim that every rook not in set S but in the

initial configuration must share a column or row with some rook in S. We prove this with a proof by contradiction:

> Proof by Contradiction: First, assume a rook exists that isn't in S and also doesn't share a row or column with any rook in S. Adding this rook to the set S will still create a peaceful configuration, since this rook shares no row or column with any rook originally in S. This new set is of rooks in a peaceful configuration, and it contains one more rook than S. We see that S is not the peaceful set of maximum size, a contradiction. Therefore our original assumption, that some rook exists that isn't in S and also doesn't share a row or column with any rook in S, is false.

Since every rook not in our peaceful configuration shares a row or column with a rook that is in our peaceful configuration, the sequence of captures to get from our initial configuration to our maximum peaceful configuration is as follows: if any rooks exist that are not in the maximum peaceful configuration, have them capture the rook that shares a row or column with them that is in said peaceful configuration. Note that if multiple rooks are in the same row or column, the one closest to the peaceful configuration rook must capture first.

We now see that any maximum sized peaceful configuration is obtainable. The minimum number of moves is thus: (# of rooks in initial state) - (maximum number of edges in bipartite matching).

References

[1] Emory College. *Algorithm for finding a maximal matching in a bi-partite graph.* URL: `http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/Matching/algorithm.html`. (accessed: 11.29.2019).

[2] John D. Cook. *Roots of integers.* URL: `https://www.johndcook.com/blog/2009/12/20/roots-of-integers`. (accessed: 11.24.2019).

[3] GeeksforGeeks. *Maximum Bipartite Matching.* URL: `https://www.geeksforgeeks.org/maximum-bipartite-matching/`. (accessed: 11.17.2019).

[4] Carl Kingsford. *CMSC 451: Maximum Bipartite Matching.* URL: `https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/matching.pdf`. pg. 5 (accessed: 11.17.2019).

[5] MIT OpenCourseWare. *6.006 Lecture 13: Breadth-first search (BFS).* URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/MIT6_006F11_lec13.pdf. (accessed: 11.17.2019).

[6] ProofWiki. *Variance of Binomial Distribution.* URL: https://proofwiki.org/wiki/Variance_of_Binomial_Distribution. (accessed: 11.24.2019).