Competitive Programming and Mathematics Society

2025 T3 Launch Week Quokka Quontest

Debrief and Prizes

CPMSoc 2025 Competitions Portfolio

Table of contents

Attendance form :D

You **MUST** fill this out to be eligible for raffle prizes, and for your vote to be counted for the upcoming AGM.

Quokka Power

Evaluate $1^6 \times 3^4 \times 5^2 \times 7^0$.

$1^6 \times 3^4 \times 5^2 \times 7^0 = 2025.$

Quoccabank

Six regular dice are rolled at the same time.
What is the probability that all the values are unique?

This question is the same as rolling one die six times, which might be helpful as it gives an "ordering" to the values. The number of ways to get 6 unique values is $6!$. The total number of possibilities for the values will be 6^6.

Therefore, the probability is $6!/6^6$, which can be simplified to $5/324$.

Basic Instances

How many times does the digit 6 appear between 1 and 678?
How many numbers between 1 and 678 contain the digit 7?

6 appears as the units digit in $6, 16, 26, \ldots, 676$, for a total of 68 times. 6 appears in the tens digit in $60-69, 160-169, \ldots, 660-669$ for a total of 70 times. 6 appears in the hundreds digit in $600, 601, \ldots, 678$, for a total of 79 times. Overall, 6 appears $68+70+79=217$ times.

Between 1 and 100, 7 is in 19 numbers (appears 10 times as the units, 10 times as the tens digit, but 77 is double counted). Similarly, for $101-200, \ldots, 501-600$, it appears $19 \times 5 = 95$ more times. For $601-678$, it appears in 9 numbers as the tens digit, and in 7 other numbers as the units digit. Overall, $19+95+9+7=130$ numbers contain the digit 7.

Basic Instances

How many times does the digit 0 appear between 6769 and 41420?
How many numbers between 15092025 and 21092025 contain the digit 9?

We can do a similar method as the first two questions, but this time we split each question into two parts, and subtract the amount in the lower bound from the amount in the upper bound.

0 appears between 1 and 6768 a total of 1947 times, and appears between 1 and 41420 a total of 16383 times. This gives an answer of $16383 - 1947 = 14436$ times.

Similarly, there are $10,935,598 - 7,592,802 = 3,342,796$ numbers which contain the digit 9.

F(Quokka)

Let $f(x)$ represent the sum of digits x.
What is the smallest positive integer x such that $(f(x)+1)^2$ is a factor of x?

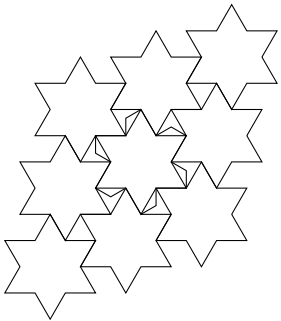
It is useful to start our search by considering numbers which are multiples of a square number $\geq 2^2$ (since $f(x) \geq 1$).

The smallest number with $f(x) = 1$ and 4 as a factor is 100. We can continue this for other $f(x)$, although it's not guaranteed that a number exists. For example, a number can't have a digit sum of $f(x) = 2$ while also being a multiple of 9.

Since we are only checking numbers which are multiples of squares that are less than 100, it doesn't take long to conclude that the smallest x is indeed 100.

Quokkstar

- A unit hexagram is a regular hexagon of side length 1 with an equilateral triangle of side length 1 affixed to each of its sides. Let $f(x)$ be the maximum proportion of area of a square of side length x you can occupy with non-overlapping unit hexagrams. Evaluate $\lim_{x\to\infty} f(x)$.



- Answer: $6/7$. Exercise: generate the diagram.

Bizarre Pea Stall

Simon is playing some carnival games, and one particularly odd stall requires him to toss a pea chip into a bottlecap. He is successful 40% of the time, and misses the other 60% of the time. His score starts at zero.
Every time he manages to put the chip into the cap, his score goes up by one. Every time he misses, his score decreases by up to two, but it will never go negative.
What is the expected number of tosses that Simon will need to reach a score of 4?

My approach was to work backwards. Let $E(X)$ represent the expected number of tosses left to reach a score of 4. We know that $E(4) = 0$, and for $X \in \{0, 1, 2, 3\}$ we have:

$$E(X) = 1 + 0.4 \times E(X + 1) + 0.6 \times E(max(0, X - 2))$$

.

Bizarre Pea Stall

$$
\begin{aligned}
E(4) &= 0 \\
E(3) &= 1 + 0.4 \times E(4) + 0.6 \times E(1) \\
&= 1 + 0.6 \times E(1) \\
E(2) &= 1 + 0.4 \times E(3) + 0.6 \times E(0) \\
&= 1.4 + 0.24 \times E(1) + 0.6 \times E(0) \\
E(1) &= 1 + 0.4 \times E(2) + 0.6 \times E(0) \\
&= 1.56 + 0.096 \times E(1) + 0.84 \times E(0) \\
&= 1/(0.904) \times (1.56 + 0.84 \times E(0))
\end{aligned}
$$

Bizarre Pea Stall

$$
\begin{aligned}
E(0) &= 1 + 0.4 \times E(1) + 0.6 \times E(0) \\
&= 1/(0.4) \times (1 + 0.4 \times E(1)) \\
&= 2.5 + 1/(0.904) \times (1.56 + 0.84 \times E(0)) \\
&= (2.5 + 1.56/0.904) + (0.84/0.904) \times E(0) \\
0.904 \times E(0) &= 2.5 \times 0.904 + 1.56 + 0.84 \times E(0) \\
0.064 \times E(0) &= 2.5 \times 0.904 + 1.56 \\
E(0) &= 3.82/0.064 \\
&= 955/16
\end{aligned}
$$

Triples

Find maximum number of triples (x_i, y_i, z_i), such that:

- $x_i + y_i + z_i = 20$ for all i,
- $x_i \neq x_j$ for any i, j where $i \neq j$,
- $y_i \neq y_j$ for any i, j where $i \neq j$,
- $z_i \neq z_j$ for any i, j where $i \neq j$,
- And lastly, the numbers x_i, y_i, z_i are all non-negative integers for all i, $(x_i, y_i, z_i \in \mathbb{Z}^+ \cup \{0\})$

.

Triples

First, let's determine the upper bound of the number of triples. Let s be the number of triples. Then,

$$S = \sum_{i=1}^{s} (x_i + y_i + z_i) = 20s.$$

From the other side,

- $S_1 = \sum_{i=1}^{s} x_i, S_2 = \sum_{i=1}^{s} y_i, S_3 = \sum_{i=1}^{s} z_i.$

These sum are at least

$$0 + 1 + 2 + \ldots + (s-1) \leq S_1, S_2, S_3,$$

which implies that:

$$S = S_1 + S_2 + S_3 \geq 3 \cdot \frac{(s-1)s}{2} = 20s.$$

Triples

Following this, we get that $s \leq \frac{40}{3} + 1$ and s is a whole number, $s \leq [\frac{40}{3}] + 1 = 14$. Since we can have up to 14 triples, we can just find such a case with $k = \frac{14}{2} = 7, 20 = 3k - 1$:

$$(0, 2k, k-1), (2, 2k-1, k-2), \ldots (2k-2, k+1, 0)$$

$$(1, k-1, 2k-1), (3, k-2, 2k-2) \ldots (2k-1, 0, k)$$

. If we substitute k, we will get triples:

$$(0, 6, 14), (1, 13, 6), (2, 5, 13), (3, 12, 5), (4, 4, 12), (5, 11, 4), (6, 3, 11),$$

$$(7, 10, 3), (8, 2, 10), (9, 9, 2), (10, 1, 9), (11, 8, 1), (12, 0, 8), (13, 7, 0)$$

Fair and Square

Andy and Brian are playing a game on a square grid. They take turns placing their initial ('A' or 'B') into an empty cell on the grid, and win if they are the first to claim all four corners of any square larger than the trivial 1 x 1. Note that this square must be axis aligned (for instance, $(0, 1), (1, 0), (1, 2), (2, 1)$ does not form a winning square).

This game can be played either on a 36 by 36 board, or an infinitely large board. For both versions, either find a winning strategy for one player, or prove that the game will end in a draw after optimal play.

1. Mistakes + Tips
2. Example solution

Fair and Square - Mistakes + Tips

1. Some submissions "proved" that the game ends in a draw by explaining that Andy can play to get a draw. The explanations were sensible, however the conclusion was wrong because they didn't show why Andy can't win. In fact, Andy can win in either board size, which is what the proof should be.

Fair and Square - Mistakes + Tips

1. Some submissions "proved" that the game ends in a draw by explaining that Andy can play to get a draw. The explanations were sensible, however the conclusion was wrong because they didn't show why Andy can't win. In fact, Andy can win in either board size, which is what the proof should be.
2. Some people skipped over the "trivial" case of Brian completely ignoring Andy's strategy and just making a square elsewhere in their first four moves. The best (cleanest) solutions were ones that won in four moves if ignored, which also meant that there were less edge cases to deal with.

Fair and Square - Mistakes + Tips

1. Some submissions "proved" that the game ends in a draw by explaining that Andy can play to get a draw. The explanations were sensible, however the conclusion was wrong because they didn't show why Andy can't win. In fact, Andy can win in either board size, which is what the proof should be.
2. Some people skipped over the "trivial" case of Brian completely ignoring Andy's strategy and just making a square elsewhere in their first four moves. The best (cleanest) solutions were ones that won in four moves if ignored, which also meant that there were less edge cases to deal with.
3. Proofs were massively improved if they could neatly explain why Brian's first (and to an extent, second) move can be ignored. Some submissions had strategies for Andy where Brian could "accidentally" win just by playing blocking moves.

Fair and Square - Mistakes + Tips

1. Some submissions "proved" that the game ends in a draw by explaining that Andy can play to get a draw. The explanations were sensible, however the conclusion was wrong because they didn't show why Andy can't win. In fact, Andy can win in either board size, which is what the proof should be.
2. Some people skipped over the "trivial" case of Brian completely ignoring Andy's strategy and just making a square elsewhere in their first four moves. The best (cleanest) solutions were ones that won in four moves if ignored, which also meant that there were less edge cases to deal with.
3. Proofs were massively improved if they could neatly explain why Brian's first (and to an extent, second) move can be ignored. Some submissions had strategies for Andy where Brian could "accidentally" win just by playing blocking moves.
4. There was a lot of submissions which had coordinates that didn't make sense. I tended to be pretty generous if I could understand what they meant to say. However, my recommendation is just to explain and justify a simplified labelling system at the start, then use that throughout. Using "Without loss of generality" can help.

Fair and Square - Example Solution

1. Credit goes to Caterpillow, whose submission largely inspired this solution.
2. We will prove that Andy can win on an "infinite board", then explain how the strategy can be modified to fit onto a 36×36 board, thus accounting for both board sizes.
3. Let $A_i = (x_{Ai}, y_{Ai})$ represent the coordinates of the cell that Andy takes on their ith turn. $B_i = (x_{Bi}, y_{Bi})$ represents similar information for Brian.
4. After playing $A_1 = (0, 0)$, Andy will only play on coordinates of the form $(\pm n_x D, \pm n_y D)$, where $n_x, n_y \in \{0, 1, 2\}$ and D is decided after Brian's first move to make it redundant.
5. We will explain how to find D later. However, know that it must be selected so that any moves Andy forces Brian to make will never give two or more corners that can form the same square with B_1.
6. Additionally, Andy should play to force Brian's moves, which prevents Brian from having time to form a square with B_1 using "free" moves.
7. $A_2 = (D, 0)$. Here, Brian's next move goes into one of three cases.

Fair and Square - Example Solution

- Case 1: $B_2 = (0, \pm D)$
  - WLOG, $B_2 = (0, D)$. Andy can achieve this by reflecting across $y = 0$.
  - From here, Andy can make Brian play a forced series of moves, where Andy is always one move off winning, and Brian is not.
  - $A_3 = (D, -D)$, $B_3 = (0, -D)$, $A_4 = (2D, 0)$, $B_4 = (2D, -D)$, $A_5 = (2D, D)$, $B_5 = (D, D)$, $A_6 = (0, -2D)$, $B_6 = (2D, -2D)$, $A_7 = (-2D, 0)$, $B_7 = (-2D, -2D)$, $A_8 = (0, 2D)$ which simultaneously threatens wins for Andy at $A_9 = (-2D, 2D)$ and $A_9 = (2D, 2D)$.
- Case 2: $B_2 = (D, \pm D)$
  - WLOG, $B_2 = (D, D)$. Andy can achieve this by reflecting across $y = 0$.
  - From here, Andy can make Brian play a forced series of moves, where Andy is always one move off winning, and Brian is not.
  - $A_3 = (D, -D)$, $B_3 = (0, -D)$, $A_4 = (2D, 0)$, $B_4 = (2D, -D)$, and A_5 onwards follows the same pattern as A_6 onwards in case 1.

Fair and Square - Example Solution

- Case 3: $B_2 \notin \{(0, \pm D), (D, \pm D)\}$
  - WLOG, Andy can modify B_2 such that $x_{B2} \geq 0, y_{B2} \geq 0$. Andy can achieve this by
  - From here, Andy can make Brian play a forced series of moves, where Andy is always one move off winning, and Brian is not.
  - $A_3 = (0, -D)$, $B_4 = (D, -D)$, $A_5 = (-D, 0)$, $B_5 = (-D, -D)$, $A_6 = (0, D)$, which simultaneously threatens wins for Andy at $A_7 = (-D, D)$ and $A_7 = (D, D)$.
- Note that in all of the cases, the furthest cell that Andy needs to claim is at most $2D$ away from the starting point, and Brian only has two unforced moves (B_1, B_2).
- To select D based on B_1:
  - If $|x_{B1}| \leq 7$ and $|y_{B1}| \leq 7$, then $D = 8$. Any square using B_1 would need at least two more points not of $(\pm nD, \pm nD)$ form. The proof is left to the reader.
  - Otherwise, use $D = 1$, such that any cell Brian is forced to take has a distance of ≥ 6 cells from B_1, meaning the other two corners lie outside the forced 5×5 zone.
- Since this strategy can fit onto the 36×36 board (start with $A_1 = (18, 18)$), then we have also proved that this strategy will work on an infinitely large board.

Quokka Pairs

A list of integers is given and a target value is given. Find if there exists a pair of numbers in the list of integers that adds up to the target value.

$\mathcal{O}(N^2)$ solution: Use a double loop that brute forces all possible pairs, see if any of them add up to the target value.

See if you can find a solution with a lower time complexity!

Alphabet

You have to sort a string in alphabet order. However, this is the Quokka's alphabet with a slightly different order.

- You can sort the array with a custom comparator which checks which letter is earlier in the alphabet in $O(N \log N)$.
- Alternatively, you can just count the amount of each letter and output them in the order of the alphabet in $O(N)$.

UNSWap

Given an array of non-negative numbers, how many pairs of digits can be swapped without affecting the sum? 0 can't be the starting digit of any positive number.

Two digits can be swapped either if:

- The digits are the same in value, or
- The digits are of the same magnitude, and 0 does not become the starting digit of a positive number

Quokkas and Koalas

You are given pairs of numbers (Q_i, K_i), two pairs can be matched iff $Q_i \cdot Q_j = K_i \cdot K_j$. Count the number of matches.

Assuming the numbers are nonzero, we can rearrange the $Q_i \cdot Q_j = K_i \cdot K_j$ into $\frac{Q_i}{K_i} = \frac{K_j}{Q_i}$.

Therefore, if we let $C_i = \frac{Q_i}{K_i}$ we want to pair up reciprocal values of C. We can do this by adding these values to a map and counting. Floats are messy, so we should just store the numerator and denominator of the reduced form.

We can also easily extend this to support matching ones with $P_i = 0$ with the ones with $Q_i = 0$. The last case is $(0, 0)$, which can be matched with anything.
With a map, we can implement this in $O(N \log N)$.

Quokkas and Koalas

If we consider these as vectors on a grid, two vectors can be matched iff their angle bisector is $(1, 1)$ (or if one of them is $(0, 0)$).

Easiest way to see this is by observing that (Q_i, K_i) and $(Q_j, -K_j)$ are perpendicular as their dot product is 0. This is what inspired the problem.

CPM Banner

You are given a prefix and you must make a string that contains “CPM” as a subsequence exactly N times. If multiple valid sequeunces exist, output the lexicographically smaller one

What matters in a prefix is the number of “CPM”s, the number of “CP”s and the number of “C”s.
We can have a DP state as DP[#“C”][#“CP”][#“CPM”] stores the minimum number of characters we must add to have N “CPM”s.
Our base case is when the number of “CPM”s is N. Transitions aren’t too hard to figure out, we can also prioritise lexicographically smaller strings at this stage.

We can output the answer by backtracking through the DP array. The complexity of this is $O(N^3)$ as we don’t need to make more than N “C”s or “CP”s.

Magic

- Quokkas have colors, and they have some chance to change their color into another.
- For each color that doesn't change its color anymore, find probabilities of this color being the final color.
- The statement was incorrect/misleading, but there is a transition, that doesn't change the color of the quokka and this color transition can appear on the final colors too.

Subtask 1: There is no cyclic transitions, so you can just find the chance by multiplying the probabilities of colors on the path from the final color to the initial color.

Magic

Full solution: This is a stochastic process, where each state is dependent on the previous state. So, one way to solve this problem would be to use An Absorbing Markov Chain (there is a Wikipedia page if you are interested.)

An absorbing Markov Chain has canonical form:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$$

where Q is transient colors, R is the final colors. If you find it's ∞-th power, it becomes:

$$P = \begin{pmatrix} Q^k & (I - Q^k)NR \\ 0 & I \end{pmatrix} = I - Q$$

, by using the quality $\sum_{k=0}^{\infty} q^k = \frac{1}{1-q}$.

Magic

Hence, we can find the probabilities by: $B = NR = (I - Q)^{-1}R$.

```
Q, R = [], []
for i in transitionalColors:
    total = sum(mat[i])
    Qrow, Rrow = [], []
    for j in transitionalColors:
        Qrow.append(Fraction(mat[i][j], total))
    for j in finalColors:
        Rrow.append(Fraction(mat[i][j], total))
    Q.append(Qrow)
    R.append(Rrow)

I = identity(len(Q))
IQ = subtract(I, Q)
N = inverse(IQ)
B = matrixMult(N, R) # B = (I-Q)^(-1) * R
print(" ".join([get_str_state(num) for num in B[0]]))
```

Word Count

There is a grid of $N \cdot M$ characters, and a dictionary of D words.
You must answer Q queries of the form

- Shift the i^{th} row by j.
- Shift every other row except for the first by a random amount.
- Output the expected number of words in the columns.

- $1 \leq N \leq 50$
- $1 \leq M, D \leq 10'000$
- $1 \leq Q \leq 500'000$

Subtasks are $M \leq 500$, $M \leq 1000$ and $M \leq 5000$.

Word Count

For the naive approach, we can shift the row by i and compare every column with every word. The probability of that word appearing in the column is easy to calculate. The expected wordcount is the sum of these probabilities.

To speed this up, we can loop through the dictionary and precompute the probability that should be added if we see character a_0 in row 0 and a_1 in row i. We can store these in $val[i][a_0][a_1]$.

A word only affects $O(N)$ probabilities, and we can calculate it's effect on the relevant entries in total of $O(N)$ time.

Each query can now be done in $O(M)$ time, so our total complexity is $O(MQ)$. If your implementation is fast, this gets you the first two subtasks.

Word Count

Our next observation is that there are only $O(NM)$ possible unique queries. As $Q \approx NM$, in the worst case we'll have to calculate every possible query anyway.

So we might as well calculate every answer beforehand, fill out a lookup table and refer to this for the relevant queries.

The advantage to doing this is we can calculate everything in our preferred order, which means we can do techniques such as sweepline, divide and conquer and (spoilers) fft.

Besides with $Q = 500'000$, it's not like we can do much else anyway.

Word Count

It makes sense to consider all possible queries row by row.

For some $val[i][a_0][a_1]$, the amount of times we need to add it for a query (i, j) is the number of a_0s in row 0 and a_1s in row i that lines up when we shift row i by j.

This is a classic FFT problem, and we can calculate these values for the entire row in $O(M \log M)$.

Right now, we do this once for every possible pairs of letters so the complexity is $O(26 \cdot 26 \cdot \log M \cdot NM)$. This doesn't improve our running time by much, but it's an important step as we avoided the clunky $O(M)$ factor.

Word Count

Instead of finding these amounts and multiplying them by $val[i][a_0][a_1]$ after, we can multiply one polynomial (let's choose the one corresponding to the i^{th} row) with $val[i][a_0][a_1]$.

With this, we can notice the relevant polynomial from the first row we are multiplying against is constant for many multiplications. We can add up the polynomials that are multiplied against it and calculate values in one big FFT.

In fact, this big summed polynomial can be calculated in $O(M)$ time as the nonzero values in the individual polynomials are disjoint.

This is now $O(26 \cdot \log M \cdot NM)$, which gets you the third subtask. However, more constant optimisations are intended to get full.

Word Count

The magic of FFT comes from our roots of unity in the complex plane, which are values w_i such that $w_i^{2^k} = 1$. However, roots of unity also exist under the mod space, and have the same properties that we need.

Notably $15311432^{2^{23}} = 1 \pmod{998244353}$. There are some maths that lets you find this value from our primitive root 3, but I won't get into that here.

We can do the same maths as in FFT, but we use this value as w instead and work within mod space. This is known as NTT (Number Theoretic Transform).

This is a lot faster as we are working with integers and we avoid the long doubles that are now needed for precision in FFT. If I choose another mod (like $1e9 + 7$), you would need to find other suitable mods and do Chinese remainder theorem tricks. However, there are nice roots of unity for 998244353 which is why it is commonly used for Competitive Programming.

Word Count

Right now, we have to do $3 \cdot 26$ NTTs or inverse NTTs for each row.

We observe that the polynomials corresponding to the first row are equal each time. Therefore, we can do the NTTs for them outside the loop and cache the results. This gets us down to $2 \cdot 26$ NTTs.

For each row, we are doing an inverse NTT for every multiplication then adding the results together. Instead, we can add the values at the roots together then do one inverse NTT at the end.

This reduces it to 27 NTTs per row, which should pass the time limit.

Contest Statistics

- Overall number of submissions
  - Maths: 967
  - Programming: 1057
  - Total: 2024
- Most submissions on specific problems:

Contest Statistics

- Overall number of submissions
  - Maths: 967
  - Programming: 1057
  - Total: 2024
- Most submissions on specific problems:
  1. 274 - Magic (Programming)
  2. 218 - Basic Instances (Maths)
  3. 216 - UNSWap (Programming)
  4. 197 - Quokkstar (Maths)
- Most submissions by individual contestants:

Contest Statistics

- Overall number of submissions
  - Maths: 967
  - Programming: 1057
  - Total: 2024

- Most submissions on specific problems:
  1. 274 - Magic (Programming)
  2. 218 - Basic Instances (Maths)
  3. 216 - UNSWap (Programming)
  4. 197 - Quokkstar (Maths)

- Most submissions by individual contestants:
  - Overall: 117, caterpillow
  - Problem: 76, caterpillow, Magic (The record from the T2 Launch Week Chicken Contest was "only" 70 submissions)

Contest Statistics

- Overall number of submissions
  - Maths: 967
  - Programming: 1057
  - Total: 2024
- Most submissions on specific problems:
  1. 274 - Magic (Programming)
  2. 218 - Basic Instances (Maths)
  3. 216 - UNSWap (Programming)
  4. 197 - Quokkstar (Maths)
- Most submissions by individual contestants:
  - Overall: 117, caterpillow
  - Problem: 76, caterpillow, Magic (The record from the T2 Launch Week Chicken Contest was "only" 70 submissions)
  - That was overall

Top 10

1. 150 - caterpillow
2. 145 - lmkae
3. 142 - Antiimony
4. 142 - OggyP
5. 140 - SirLemonMeringue
6. 132 - asb123
7. 132 - Blaiz
8. 132 - quokka
9. 131 - ciple
10. 130 - programmer123

Other Prizes

- Women and Gender Minorities
  - poko
  - Tini
  - taiga137
- First years
  - vivant
  - gavvin
  - goon
- Raffle Prizes
  - icecarbon
  - cosintheta
  - rtrgrd

Further events

Please join us for:

- Annual General Meeting (starting now!)
- WiseTech Global Challenge - Sunday September 28, 12 - 3 pm. Applications close today!
- SPAR #9 - Saturday October 4, 12 - 5 pm. Mirror of Monash PC.

Attendance form :D

You **MUST** fill this out to be eligible for raffle prizes, and for your vote to be counted for the upcoming AGM.