Competitive
Programming and
Mathematics
Society

2025 T2 Launch Week Chicken Contest

Debrief and Prizes

CPMSoc 2025 Competitions Portfolio

Table of contents

Sum 90

When

$$X + Y = 90,$$

$X \times Y$ is maximised if

$$X = Y = 45.$$

Therefore,

$$X \times Y = 45 \times 45 = 2025.$$

Happy 2025!

Joining Points

All distances can be created by fixing one point to $(0, 0)$, and selecting a second point.

Since the distances are symmetric about the line of $y = x$, we are left with 5 other points along $y = x$, and $30 \div 2 = 15$ points where $y < x$. This results in a total of 20 points.

However, $(3, 4)$ and $(5, 0)$ are both a distance of 5 away from $(0, 0)$, so this distance is currently being double counted.

Therefore, the total number of distances is $20 - 1 = 19$.

Wide Ruler

Disclaimer: This problem has a lot of methods, here is a dumber one.

- Let's explain this: https://www.desmos.com/calculator/8fxjmd6xwc
- Construct the page as the rectangle contained by $0 \leq x \leq 210$ and $0 \leq y \leq 297$.
- We'll define a 300 mm edge of the ruler as $\overline{AB}$, where $A = (a, 0)$ and $B = (0, i)$.
- The shorter edges of the ruler $\overline{AD}$ and $\overline{BC}$ are both perpendicular to $\overline{AB}$, and intersect the edges of the page at D and C respectively.
- We can express everything in terms of a, which has a domain of $\left(\sqrt{300^2 - 297^2}, 210\right)$.
- For instance, $i = \sqrt{300^2 - a^2}$ (which is already capped at 297 due to the domain of a).
- $\overline{AB}$ has a gradient $m_1 = -i/a$, and since the ruler is a rectangle, where sides are perpendicular, lines $\overline{AD}$ and $\overline{BC}$ have a gradient $m_2 = -1/m_1 = a/i$.
- From here, we can find that $D = \left(210, \frac{a}{i}(210 - a)\right)$ and $C = \left((297 - i) \div \frac{a}{i}, 297\right)$.

Wide Ruler

- Next, let's explain this: https://www.desmos.com/calculator/h1mplb7im7
- We can use Pythagorean Theorem to find distances $|AD|$ and $|BC|$, and express them as a function of a.
- $|AD| = \sqrt{(210-a)^2 + \left(\frac{a}{i}(210-a)\right)^2}$
- $|BC| = \sqrt{\left((297-i)\frac{i}{a}\right)^2 + (297-i)^2}$
- The width of the ruler is bounded by the lesser of $|AD|$ and $|BC|$.
- While we *could* solve for that, we can also just graph both functions.
- When we do that, we find that the solution exists at the intersection point.
- Therefore, the greatest width is 72.36 units, which is rounded to 72.

- Bonus link of another "method": https://www.desmos.com/calculator/dpne40weq3

Consecutive Counting

A positive integer is called "egg-like" if it is the sum of 2 or more consecutive positive integers. For example, 22 is egg-like since $22 = 4 + 5 + 6 + 7$. Find the number of positive integers under:

(a) 10^6

(b) 10^{11}

(c) 10^{16}

that are not egg-like.

Consecutive Counting

We note that the sum of consecutive integers from m to $m+n-1$ is

$$S = mn + \frac{n(n-1)}{2} = \frac{n}{2}(2m+n-1).$$

We can show that S is never a power of two. Firstly, we see that $n \geq 2$ and $2m + (n-1) \geq 2$. Thus, if either are odd, then S is not a power of two.

- If n is odd, then since S is a multiple of n, S isn't a power of two.
- If n is even, then $2m+n-1$ is odd. As S is a multiple of $2m+n-1$, S isn't a power of two.

and thus S can never be a power of two.

Consecutive Counting

For any integer x that is not a power of two, we get that $x = 2^a b$ for some non-negative integer a and odd integer $b \geq 3$. We now have two different cases:

- If $b < 2^{a+1}$, we set $n = b$ and $m = \frac{1}{2}(2^{a+1} - b + 1)$, and we will get $S = x$.
- If $b > 2^{a+1}$, we set $n = 2^{a+1}$ and $m = \frac{1}{2}(b - 2^{a+1} + 1)$ and we will get $S = x$.

and we thus see that a positive integer is not egg-like iff it is a perfect power of two.

For extraction, take $\lceil \lambda \log_2 10 \rceil$ for $\lambda = 6, 11, 16$ to get $20, 37, 54$ respectively.

Massive Mall

After a long day's work, Jerry decided to reward himself with some food. As he entered a mall, he was struck with awe from it's size, and of how many restaurants were serving fried chicken. Wondering how many floors were in the building he thought...

A building has 13 elevators which each stopping for at most 4 floors. If one can go from any floor to any other floor using only one elevator, what is the maximum number of floors in the building?

Massive Mall

We first find an upper bound.

Let there be n floors in the building. We find that for every pair of floors, we must have at least one elevator stopping at both of them, and so we require $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs to have an elevator between them. Note that every elevator can stop for at most $\binom{4}{2} = 6$ pairs floors, and thus there must be at most $6 \times 13 = 78$ pairs of floors. Therefore, we find that:

$$78 \geq \frac{n(n-1)}{2}$$

and as $\frac{13 \times 12}{2} = 78$, we see that $n \leq 13$.

Massive Mall

Here is the construction for $n = 13$ (Ps are the floors, ms are the elevators).

Lines / Points	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}
P_1	1	1	1	1									
P_2	1				1	1	1						
P_3	1							1	1	1			
P_4	1										1	1	1
P_5		1			1			1			1		
P_6		1				1			1			1	
P_7		1					1			1			1
P_8			1		1				1				1
P_9			1			1				1	1		
P_{10}			1				1	1				1	
P_{11}				1	1					1		1	
P_{12}				1		1		1					1
P_{13}				1			1		1		1		

Figure: Construction (Source: Wikipedia)

Algebra 1

Find all functions $f : \mathbb{R} \to \mathbb{R}$ such that:

$$f(af(-b) + f(a-b)) + b = (b-1)f(a)$$

for all real numbers $a, b \in \mathbb{R}$

Algebra 1 Solution

We first see that the function can't be a constant equal to 1, as otherwise we could take $(a, b) = (1, 1)$ and obtain:

$$f(f(-1) + f(0)) + 1 = (1 - 1)f(1)$$
$$1 + 1 = 0$$

which is obviously false. Thus there exists a $c \in \mathbb{R}$ be a value such that $f(c) \neq 1$, and so

$$\begin{aligned} f(cf(-b) + f(c - b)) &= (b - 1)f(c) - b \\ &= b(f(c) - 1) - f(c) \end{aligned}$$

and as $f(c) - 1 \neq 0$, by varying b we can see that our function f can attain any real value and thus is surjective (for and $k \in \mathbb{R}$, we consider the substitution $(a, b) = \left(c, \frac{f(c)+k}{f(c)-1}\right)$ to get $f(x) = k$ for some $x \in \mathbb{R}$).

Algebra 1 Solution

Claim: $f(0) \neq 1$.

For the sake fo contradiction, assume that $f(0) = 1$. From the substitution $(a, b) = (0, b)$, we find

$$\begin{aligned} f(f(-b)) + b &= (b-1)f(0) \\ f(f(-b)) &= b - 1 - b \\ &= -1 \end{aligned}$$

and so $f(f(-b)) = -1$ for all $b \in \mathbb{R}$. As f is surjective, we additionally find that $f(x) = -1$ for all $x \in \mathbb{R}$. However, this means that f is constant, contradicting surjectivity.

Algebra 1 Solution

Claim: f is injective.

Indeed if $f(x) = f(y)$ for some $x, y \in \mathbb{R}$, then from $(a, b) = (0, -x)$ and $(a, b) = (0, -y)$, we find that

$$\begin{aligned} x(1 - f(0)) - f(0) &= f(f(x)) \\ &= f(f(y)) \\ &= y(1 - f(0)) - f(0) \end{aligned}$$

and as $1 - f(0) \neq 0$, we find that $x = y$, as desired.

Algebra 1 Solution

From the substitutions $(0, 1) = (a, b)$, and $(x + 1, 1) = (a, b)$, we find that

$$f(f(-1)) + 1 = 0$$
$$f(f(-1)) = -1$$

and

$$f((x + 1)f(-1) + f(x)) + 1 = 0$$
$$f((x + 1)f(-1) + f(x)) = -1.$$

As f is injective, we thus find that:

$$f(f(-1)) = f((x + 1)f(-1) + f(x))$$
$$f(-1) = (x + 1)f(-1) + f(x)$$
$$-xf(-1) = f(x)$$

and so $f(x) = \lambda x$ for all $x \in \mathbb{R}$ for some fixed $\lambda \in \mathbb{R}$.

Algebra 1 Solution

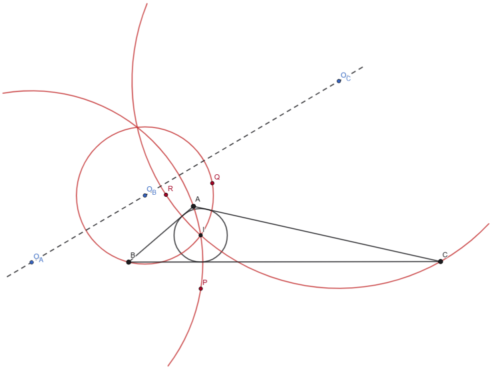
We substitute this into our original equation to get:

$$\begin{aligned} f(af(-b) + f(a-b)) + b &= (b-1)f(a) \\ f(-\lambda ab + \lambda a - \lambda b) + b &= (b-1)\lambda a \\ -\lambda^2 ab + \lambda^2 a - \lambda^2 b + b &= \lambda ab - \lambda a \\ (-\lambda^2 - \lambda)ab + (\lambda^2 + \lambda)a + (1 - \lambda^2)b &= 0 \end{aligned}$$

and thus $\lambda^2 + \lambda = 0$ and $1 - \lambda^2 = 0$. This gives us the unique value $\lambda = -1$ for the only solution $f(x) = -x$.

Geometry 1

Let ABC be a scalene triangle with incentre I. Denote P, Q, and R as the reflections of I over lines BC, AC, and AB respectively. Prove the centre of the circles (PIA), (QIB), and (RIC) are colinear. (Note: (XYZ) is the circle passing through the 3 points X, Y, and Z).



Geometry 1 (Sol 1)

We can see as the three circles intersect at a point, it suffices to show that they will all pass through a second point K (as then the three centres of the triangles will lie on the perpendicular bisector of IK).

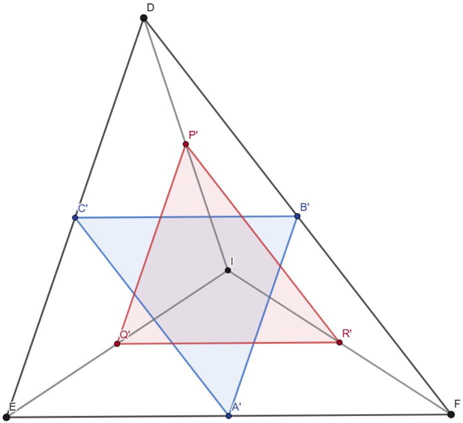
Let the incircle touch sides BC, AC, AB at D, E, F respectively. Consider the inversion about the incircle of the triangle. We can see that:

- D, E, and F map to themselves.
- P, Q, R map to the the midpoints of ID, IE, and IF respectively.
- A, B, C map to the midpoints of EF, DF, and DE respectively.

Now it suffices to prove that lines $P'A'$, $Q'B'$, and $R'C'$ concur.

Geometry 1 (Sol 1)

Now it suffices to prove that lines $\overline{P'A'}$, $\overline{Q'B'}$, and $\overline{R'C'}$ concur.



Geometry 1 (Sol 1)

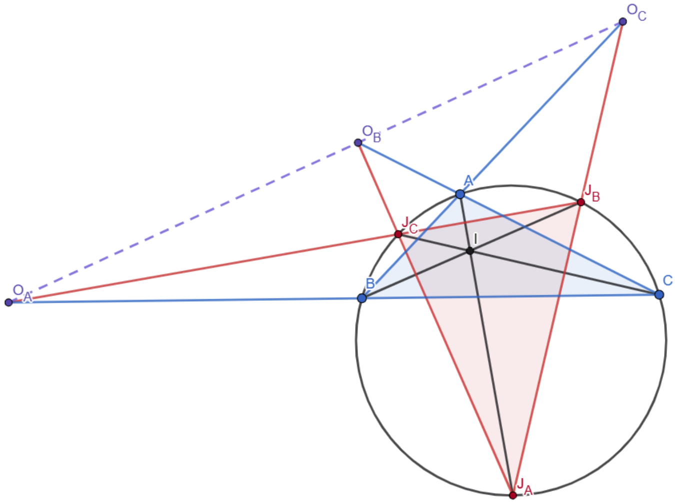
We see that $P'R' = \frac{1}{2}DF = A'C'$ and $\overline{P'R'} \parallel \overline{DF} \parallel \overline{A'C'}$ and so $P'R'A'C'$ is a parallelogram and $\overline{R'C'}$ passes through the midpoint of line segment $\overline{P'A'}$. Similarly, we find that $P'Q'A'B'$ is a parallelogram and $\overline{Q'B'}$ passes through the midpoint of $\overline{P'A'}$, and thus the three lines intersect (at the midpoint of $\overline{P'A'}$

Geometry 1 (Sol 2)

Denote the circumcentres of (PIA), (QIB), and (RIC) as O_A, O_B, O_C respectively, and rays AI, BI, CI to intersect (ABC) again at J_A, J_B, and J_C.

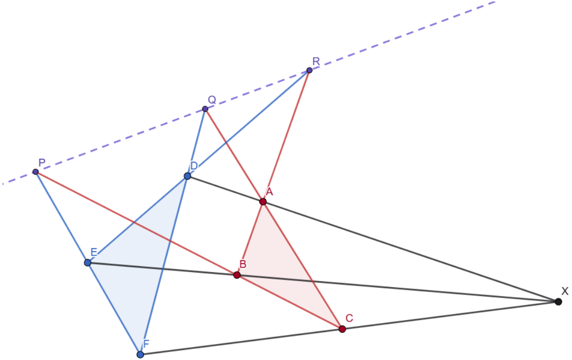
Now, we see that O_A is the intersections of the perpendicular bisectors of AI and PI, which are lines BC and J_BJ_C respectively. Similarly, $O_B = AC \cap J_AJ_C$ and $OC = BC \cap J_BJ_C$.

Geometry 1 (Sol 2)



Geometry 1 (Sol 2)

Now, we can see triangles ABC and $J_AJ_BJ_C$ are in perspective of a point, and so by Desargues' Theorem they are in perspective of a line; i.e, $AB \cap J_AJ_B = O_C$, $AC \cap J_AJ_C = O_B$, and $BC \cap J_BJ_C = O_A$ are colinear as desired.



Geometry 1 (Sol 2)

Alternatively, we can see that as $I = J_C C \cap J_B B$, from Brocard's Theorem, $J_C J_B \cap BC = O_A$ lies on the polar of I wrt (ABC). Similarly, we find that O_B and O_C also lie on the polar of I wrt (ABC), and thus all 3 are colinear (specifically lying on this polar).

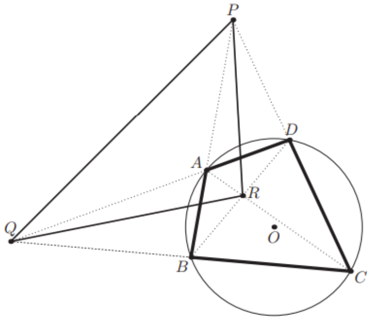
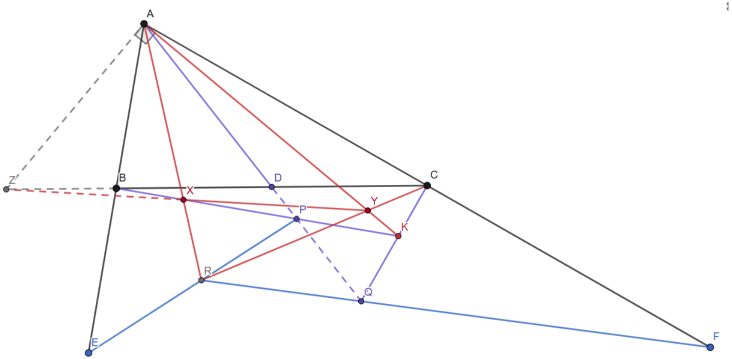


Figure: Image from Euclidean Geometry in Mathematical Olympiads by Evan Chen

Wild Chicken Chase

Let ABC be a scalene triangle, D be the midpoint of side BC and E, F, the reflections of A over points B, C. Let K be the point such that $\angle ABK = \angle ACK = 90°$. Lines BK and CK intersect line AD at P and Q respectively, and lines EP and FQ intersect at R. Let AR and KB meet at X, AK and CR meet at Y, and lines BC XY intersect at Z. Show that $\angle ZAK = 90°$.



Wild Chicken Chase

Claim: R lies on (EKF).

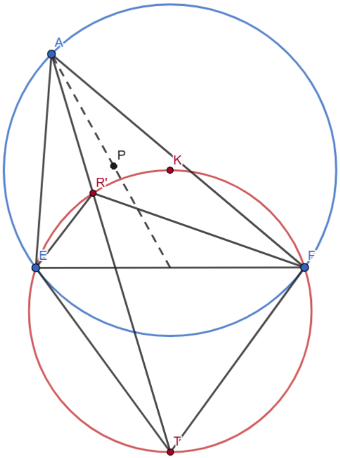
We first note that K is the circumcentre of triangle AEF. Thus, $2\angle EAF = \angle EKF$. We then find that:

$$\begin{aligned}\angle ERF &= \angle EAF + \angle AER + \angle AFR \\ &= \angle EAF + \angle EAP + \angle FAP \\ &= 2\angle EAF\end{aligned}$$

and thus $\angle ERF = \angle EKF$ and so R lies on (EKF) as desired.

Wild Chicken Chase

Claim: Let the tangents of E and F to (AEF) meet at T. Then $A - R - T$, and $R \neq T$.
We construct R' to be the point such that $AT \cap (EKF) = \{T, R'\}$.



Wild Chicken Chase

We see that as $ET = FT$ (from ice cream cone), we get that $\angle ER'T = \angle FR'T$, and so

$$\angle ER'T = \frac{1}{2}\angle ER'K = \angle EAF.$$

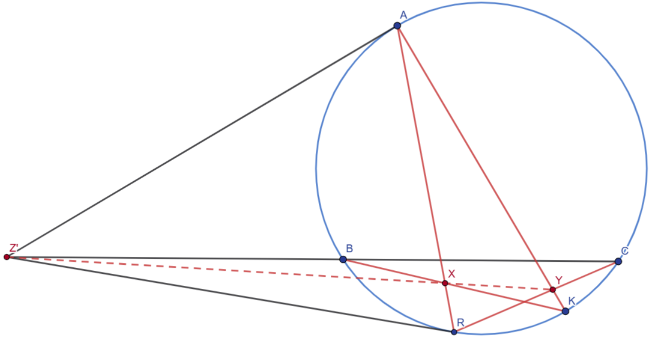
Thus, we find that $\angle R'EA + \angle R'AE = \angle R'AE + \angle R'AF$, or $\angle R'EA = \angle R'AF = \angle EAP$ as AT and AD are isogonal wrt $\angle EAF$. We thus find that:

$$\angle R'EA = \angle EAP = \angle AER$$

and as R' and R both lie on (EKF), and $\angle R'EA = \angle REA$ we find that $R = R'$, and thus $A - R - T$

Wild Chicken Chase

Now, we see that as $\angle TRK = 90^\circ$, we require $\angle ARK = 90^\circ$, and thus $R \in (ABC)$. We construct point $Z' = AA \cap BC$, and it suffices to show $X - Y - Z$, (as then $Z' = Z$, and AZ is a tangent to (ABC), and since K is the antipode of A wrt (ABC) we get $\angle ZAK = 90^\circ$).



Solution 1: Pascal's

Pascal's on the degenerate cyclic hexagon $AARCBK$ will get us the desired condition.

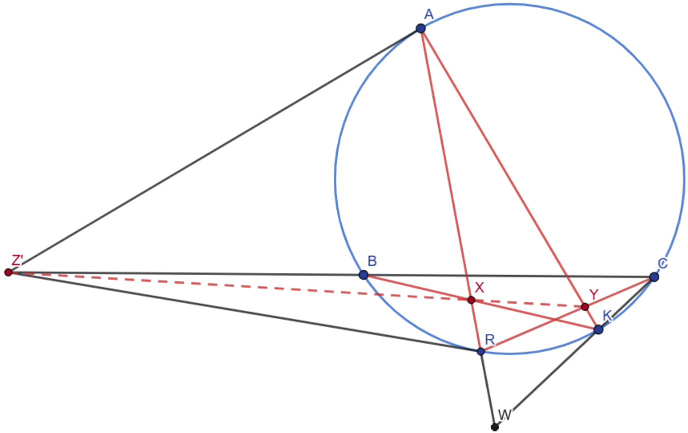
Solution 2: Duality

It suffices to show that the polars of the 3 points (wrt (ABC)) are concurrent. Indeed, we can prove that these lines pass through the point $AR \cap CK = W$.

- The polar of Z' is line AR as AR is the A-symmedian of triangle ABC.
- The polar of Y passes through W from Brocard's Theorem
- We can see that $(A, R; X, W) = (A, R; B, C) = -1$ by taking perspectivity from K. Thus, as AR is a chord of (ABC), the polar of X passes through W.

Solution 2: Duality

As a statement is equivalent to its dual, we find that $Z' - Y - X$, as desired.



Solution 3: Harmonic Chase

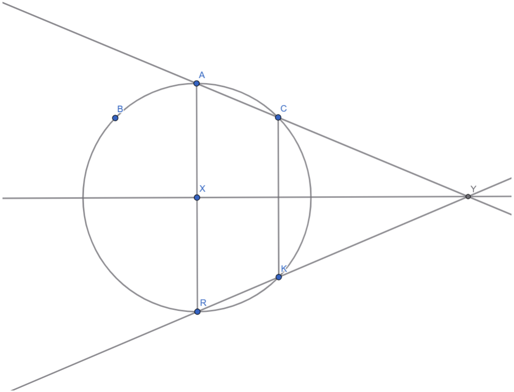
Define W similarly to before. Then we find:

$$\begin{aligned}(BC \cap AX, XY \cap BC; C, B) &\overset{X}{=} (A, Y; CX \cap AK, K) \\ &\overset{C}{=} (A, R; X, W) \\ &\overset{K}{=} (A, R; B, C) \\ &= -1\end{aligned}$$

and as we know that $(BC \cap AX, Z'; C, B) = -1$, we find that $XY \cap BC = Z'$, or $Z' - X - Y$ as desired.

Solution 4a: Projective Transform

This covers essentially 2 solutions. We note that the problem is purely projective, and so we can consider a projective transformation that sends (ABC) to itself and X to the centre of the circle.



Solution 4a: Projective Transform

We then see that AR is a diameter, and as cross ratio is preserved, we find that C is the reflection of B over AR. As BK is a diameter, we find that K is the reflection of B over $X \in AR$.

Thus, we find that $CK \parallel AR$, and $ACKR$ is an isosceles trapezium as it is cyclic. Hence $Y = AC \cap KR$ lies on the perpendicular bisector of AR, and $XY \perp AR$.

Finally, we note that Z' is the point at infinity that lies on the tangent at A, which is also perpendicular to AR, and thus $Z' - X - Y$ as desired.

Solution 4b: MMP (Tethered)

UNSW CPMSoc

Let K be a moving point on (ABC), and fix A, B, C, R. Then we find that Z' is fixed, and the mappings $f : K \overset{B}{\mapsto} AR \overset{Z'}{\mapsto} RC$ and $g : K \overset{A}{\mapsto} RC$ are both projective.
Here, we can see $f(K) = Z'X \cap RC$ and $g(K) = Y$.
Finally, we see that

$$f(P) = g(P)$$
$$f(C) = g(C)$$
$$f(A) = g(A)$$

and thus the two maps f and g are the same; i.e, $Z'X \cap RC = Y$, and thus $Z' - Y - X$ as desired.

Fair Digsum

- Brute-force all numbers between 1 and M, check if the product between the modulus and the digsum squared is equal to the number itself.
- The time complexity is around O(M*logM), as for each number number between 1 to M, a digit sum operation is done, which is approximately log(M).
- Below is a c++ implementation to get the sum of digits for any number.

```cpp
int s(int a) {
    int sum = 0;
    while (a > 0) {
        sum += a%10;
        a /= 10;
    }
    return sum;
}
```

Stupid Dot Product

- Somewhat trivially, if the dot product is already 0 then no pecks are required.
- The first proper observation is that for any pair of arrays A and B, the dot product can always be made into 0 with exactly 2 pecks. One way to do this is to set $A_1 = 1$, then set B_1 to cancel out the dot product of all following terms.
- Thus, the challenge becomes determining when only one peck is required.
- The main observation is that when any A_i changes, the overall dot product changes by a multiple of B_i, and vice versa. As such, with only one peck, the overall dot product can only change by a multiple of an existing value in A or B.
- Therefore, if at least one term in A or B is a factor of the initial dot product, then the problem can be solved with just one peck. Otherwise, two pecks will be required.
- To confirm if this approach if fully sufficient, have a think about if/how this approach accounts for the case where $N = |A| = |B| = 1$.

Frog Pond

- **Abridged statement:** You are given an undirected graph with weighted edges. For each query, find the maximum value that is the smallest weighted edge on some path from the q_i-th node to the 0-th node.
- This problem didn't have any subtasks, but a naive way to solve this problem would be to, for each query, start with an empty graph, then starting from the heaviest edge, add it onto the graph till the nodes q_i and 0 become connected.
- Q. What can we improve from this solution?
- A. Every time we build the graph from the empty state, it's going to do the same steps over and over again. So we need to build the graph only once.
- How do we know if a node is connected to the 0-th node? We can use Disjoint Set Union (DSU). Not only keeping track of the parent node of a node, but additionally, we need to keep track of the nodes that are connected to the current node. Since this can be quite taxing on both memory and time, we can only merge the smaller side to the bigger side. This makes the total time complexity run in: $O(M\log(N))$ where N is the number of nodes and M is the number of edges.

Chicken merger

- **Abridged statement:** Given a list of numbers, you can perform the following operations on these numbers:
  - Add a number A.
  - Remove an existing number B.
  - Find the highest number you can get by repeatedly merging all duplicate numbers.
  - Merge two of the same numbers C to get $C + 1$.
- For the first subtask, you can keep track of all your numbers and for each "Find" operation, make a duplicate of the existing numbers. After sorting all numbers in decreasing order, you can merge the smallest numbers first, and remove the smallest number if no duplicates are left. Once you have only one number, you can get your answer.
- As for the second subtask, it's hard to merge all numbers when you need to remove or add a new number to this list. So we will merge all of the numbers that can be merged and create a list with no duplicate numbers. Now, if we were to group the consecutive numbers into a single group, we can do the insertion or deletion of an

Chicken merger

- existing number in approximately $O(\log N)$ time assuming you are using some data structure that is similar to C++ ‘set<pair<int, int> >‘
  - For insertion, if the new value a is non-existent we can insert the value (a, a). Also if there is a group that forms a new consecutive list, we can update that group.
  - For insertion, if the new value a is present and assuming that value was in group (l_i, r_i), we can form new groups $(l_i, a-1)$, (r_i+1, r_i+1).
  - For deletion, if the value a is non-existent, we can get the smallest group (l_i, r_i) and replace it with the new groups (a), (l_i+1, r_i)
  - For deletion, if the value exists, we can get the smallest group (l_i, r_i) and replace it with the new groups (a, l_i-1), (l_i+1, r_i).
- We can do all of these four operations in approximately $O(\log N)$ time. As long as we make sure that the groups are as long as possible and all of the elements are unique, the biggest value you can get after all the merges is always the biggest value in this set. This means, we can do all of the operations in total $O(Q\log(Q+N))$ time.

Chicken Run

- Given weighted directed graph. You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score is capped below at 0. Max score after exactly K moves?

Chicken Run

- Given weighted directed graph. You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score is capped below at 0. Max score after exactly K moves?
- At most 100 vertices, $1 \leq K \leq 10^9$ and $-10^9 \leq w \leq 10^9$.

Chicken Run

- Given weighted directed graph. You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score is capped below at 0. Max score after exactly K moves?
- At most 100 vertices, $1 \leq K \leq 10^9$ and $-10^9 \leq w \leq 10^9$.
- Subtask 1: $K \leq 1000$. DP, find max score at vertex i after j moves.

Chicken Run

- Given weighted directed graph. You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score is capped below at 0. Max score after exactly K moves?
- At most 100 vertices, $1 \leq K \leq 10^9$ and $-10^9 \leq w \leq 10^9$.
- Subtask 1: $K \leq 1000$. DP, find max score at vertex i after j moves.
- Subtask 2: weights are positive. Matrix exponentiation: $M^k[u][v]$ is the max sum from u to v using exactly k edges. The addition is max and the multiplication is addition.

Chicken Run

- Given weighted directed graph. You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score is capped below at 0. Max score after exactly K moves?
- At most 100 vertices, $1 \leq K \leq 10^9$ and $-10^9 \leq w \leq 10^9$.
- Subtask 1: $K \leq 1000$. DP, find max score at vertex i after j moves.
- Subtask 2: weights are positive. Matrix exponentiation: $M^k[u][v]$ is the max sum from u to v using exactly k edges. The addition is max and the multiplication is addition.
- Full: matrix exponentation: $M^k[u][v]$ is a function, the maximum final score from u to v using exactly k edges, for each starting score. The addition is pointwise max and the multiplication is composition. Every function will be of the form $x \mapsto \max(a, x+b)$.

Chicken Run

- Given weighted directed graph. You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score is capped below at 0. Max score after exactly K moves?
- At most 100 vertices, $1 \leq K \leq 10^9$ and $-10^9 \leq w \leq 10^9$.
- Subtask 1: $K \leq 1000$. DP, find max score at vertex i after j moves.
- Subtask 2: weights are positive. Matrix exponentiation: $M^k[u][v]$ is the max sum from u to v using exactly k edges. The addition is max and the multiplication is addition.
- Full: matrix exponentation: $M^k[u][v]$ is a function, the maximum final score from u to v using exactly k edges, for each starting score. The addition is pointwise max and the multiplication is composition. Every function will be of the form $x \mapsto \max(a, x + b)$.
- Alternative: if answer is positive, $\max_{u,v}\{d(u) + M^{K-d(u)}[u][v]\}$, where $d(u)$ is BFS distance from 1 to u and M is from subtask 2, will find an optimal walk, and also won't overestimate the answer. Wait is this correct?

Chicken Run

- Given weighted directed graph. You start at vertex 1 with score 0. When you move along an edge, its weight is added to your score, however your score is capped below at 0. Max score after exactly K moves?
- At most 100 vertices, $1 \leq K \leq 10^9$ and $-10^9 \leq w \leq 10^9$.
- Subtask 1: $K \leq 1000$. DP, find max score at vertex i after j moves.
- Subtask 2: weights are positive. Matrix exponentiation: $M^k[u][v]$ is the max sum from u to v using exactly k edges. The addition is max and the multiplication is addition.
- Full: matrix exponentation: $M^k[u][v]$ is a function, the maximum final score from u to v using exactly k edges, for each starting score. The addition is pointwise max and the multiplication is composition. Every function will be of the form $x \mapsto \max(a, x + b)$.
- Alternative: if answer is positive, $\max_{u,v}\{d(u) + M^{K-d(u)}[u][v]\}$, where $d(u)$ is BFS distance from 1 to u and M is from subtask 2, will find an optimal walk, and also won't overestimate the answer. Wait is this correct?
- Some fake solutions passed, oh well.

Tree Building

You are given a rooted tree with N nodes numbered from 1 to N. The tree has $N-1$ edges, and node 1 is the root. Each node i has a value $X[i]$.

In one operation, you may select any two nodes i and j. This operation costs $|X[i] - X[j]|$ units of energy and activates every edge along the path connecting nodes i and j. You can perform this operation as many times as you wish.

Your task is to find, for each non-root node, the minimum amount of energy needed to activate all edges on the path from that node to the root.

Extension: What if two activated paths can't share an endpoint?

Tree Building

The main observation is that you only want to select nodes with adjacent X values.

If you want to select (A, C) and there is some $X[A] \leq X[B] \leq X[C]$, it is better to select to select either (A, B), (B, C) or both.

Therefore, there are only $N - 1$ paths we have to consider activating to cover all the cases.

Tree Building

The lowest common ancestor of two points is the deepest node that has both points as the descendants.

If you find the LCA of the endpoints of a path, it splits the path into two paths that goes down the tree.
To cover the path leading down to an arbitrary node, only one of these paths are necessary.

We now just have to solve the problem:
You are given $2 \cdot (N - 1)$ paths which are going down the tree.
For every node, find the cheapest set of paths that cover the path from the root.

Tree Building

This ends up working quite similarly to the single source shortest path problem. In fact, we can solve this with Dijkstra.

For all of the paths we made (A, B) with A an ancestor of B, add a path with that cost from A to B.

For all non-root nodes, add a path with cost 0 to it's parent.

If you run dijkstra from the root, it solves this question.

Tree Building

Orz jllyfish for finding this solution. The intended solution is a DP.

It makes sense to calculate values going down the tree, the deeper values depends on higher values.

If you can get to node A with cost $DP[A]$ and there is a path (A, B) with cost C, you can get to all nodes on the path with $DP[A] + C$.

Notably, these are all the points with B in it's subtree (as higher points have already been calculated). From any point, it's value would be $DP[X] = \min_{B \in \text{subtree}(X)}(DP[A] + C)$.

Therefore, we can put the value $DP[A] + C$ into the node B and do subtree min query to find future DP values.

Contest Statistics

- Overall number of submissions
  - Maths: 485
  - Programming: 560
  - Total: 1045

Contest Statistics

- Overall number of submissions
  - Maths: 485
  - Programming: 560
  - Total: 1045
- Most submissions on specific problems:
  - Maths: 149, Wide Ruler
  - Programming: 168, Chicken Merger
- Most submissions by individual contestants:

Contest Statistics

- Overall number of submissions
  - Maths: 485
  - Programming: 560
  - Total: 1045
- Most submissions on specific problems:
  - Maths: 149, Wide Ruler
  - Programming: 168, Chicken Merger
- Most submissions by individual contestants:
  - Overall: 70, Albert
  - Problem: 28, Albert, Chicken Merger
  - Problem: 28, horiseun, Chicken Run

Top 10

1. 123 - ds
2. 120 - caterpillow
3. 117 - Jlyfish
4. 110 - programmer123
5. 110 - ciple
6. 100 - horiseun
7. 91 - praneelm
8. 89 - wourt
9. 88 - LouisNguyen
10. 87 - csonda

Other Prizes

- Women and Gender Minorities
  - chimken_wingz
  - tspmo
  - xklii_
- First years
  - SirLemonMeringue
  - casma.
  - vivant
- Raffle Prizes
  - cosintheta
  - asd213
  - tseegii_33700
  - rain

Further events

Please join us for

- Week 5
  - SPAR #5 - Saturday 5/07, 12 - 5 pm
  - Integration workshop
  - Advanced graphs workshop
- Week 6 - Flex week (take a break)!
- Week 7 and beyond
  - Maths contest
  - Clubs Takeover
  - More workshops
  - CPMSoc hoodies

Attendance form