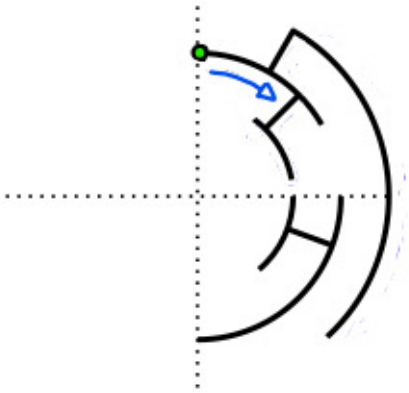
Problem Tutorial: "Advanced Compass"

This is problem Mad Diamonds from CTU Open 2021.

Your task was to solve a brain teaser and figure out how to transport a diamond from the start point to the end point. At the same time you need to minimize the total sum of degrees that the maze needs to be rotated by in order to achieve that.

- The task can be modelled as a graph search.
- The state configuration consists of the layer at which ball is located, the rotation of the maze in degrees and the angular position of the ball in the maze, in degrees as well.
- There were at most 20 layers.
- Therefore as a result there might be at most $20 \cdot 360 \cdot 360 = 2592000$ reachable states.

There are a few tricky cases to handle. In the case below if the ball starts rolling down to the right we need to decide whether the ball should continue straight through the circular segment or rather fall through the radial segment:



That needs to be decided based on the angle at which ball (considering the direction of gravity) meets the crossing between radial and circular segments.

Another potential downfall is that at the start point the marble can start at such a position that the gravity will move it somewhere else right away.

And similar thing applies to the finishing of the puzzle as well that the marble has to stop at the end point precisely, not just run through it which might make the puzzle impossible to solve.

Problem Tutorial: "Breaking Bars"

This is problem B from NCPC-2021. Problem authors: Mats Petter Wallander and Andreas Bjorklund.

First consider the non-optimal solution to partitioning all the chocolate to get the upper bound:

- Compute the number of bars of each bar size (there are 21 sizes).
- Always greedily pair up bars so that you never have 2 or more of any size.
- Find minimum number of breaks needed for remaining bars.
- Can be computed efficiently by dynamic programming over 2^{21} states.
- Does not always give the optimum number of breaks. Example: `3x2 3x3 1x5 2x5 3x5 3x5`. Split one `3x5` as `3x2 + 3x3` and the other as `1x5 + 2x5` to get away into two splits.

This does give upper bound on number of breaks that may be needed (it is 9).

Then the actual solution will be:

- Recursively search for the best way to break the bars, going from larger bars to smaller ones.
- Keep track of number of breaks made and how many squares can be partitioned among the larger bars already considered.
- Avoid recursive calls that will produce more breaks than best solution found.
- Since there is no need for more than 9 breaks and any bar be broken in at most 5 different ways, this turns out to be fast enough.

Problem Tutorial: "Customs Control"

This is problem C from NCPC-2021. Problem author: Nils Gustafsson

1. Dijkstra's algorithm finds all edges that are part of some shortest path from 1 to n.
2. These edges form a directed acyclic graph. Find a topological ordering.
3. Color the first k vertices in the ordering red, and the remaining ones blue:
   - A shortest path from 1 to n now only switches between red and blue once, so every shortest path on 3 or more vertices must have a monochromatic edge.
4. Special case: this does not work if there is a direct edge from 1 to n.
   - Since graph is vertex-weighted, edge from 1 to n is the only shortest path.
   - We only need to make sure 1 and n get the same color.
   - Always possible, except if n = 2 and k = 1.

Problem Tutorial: "Dog"

This is problem C from BAPC-2021. Problem author: Abe Wits

- **Solution:** Do DP and calculate what the minimal number of cans is needed if you fill up the last r rows for a given can placement of the top row.

  For calculating the next row, iterate over all rows that support a can placement and take the best.

$$\texttt{DP}[\text{row}][\mathcal{C}] = \begin{cases} |\mathcal{C}| + \min\limits_{\mathcal{D} \text{ supports } \mathcal{C}} \texttt{DP}[\text{row}-1][\mathcal{D}] & \text{if } \mathcal{C} \text{ covers locations,} \\ \infty & \text{else.} \end{cases}$$

- Number of can placements is F_{m+1}, the $(m+1)$th Fibonacci number.

  Time complexity: $\mathcal{O}\left(n \cdot F_{m+1}^2\right) = \mathcal{O}\left(n \cdot 3.3^m\right)$ when using bitmasks.

Problem Tutorial: "Eavesdropper Evasion"

This is problem E from NCPC-2021. Problem author: Jimmy Mardell

Let's find a solution if at most 1 message is allowed to be intercepted:

- If a message (regardless of length) starts at time a, the *next* message can start no earlier than time $a + x + 1 - t$, where t is the length of the next message.
- Repeating this, the total time to send all n messages becomes

$$x + 1 + \sum_{i=2}^{n-1} (x + 1 - t_i)$$

  where t_i is the length of the i-th message sent.
- Note: length of the first and last message does not affect the total send time!
- By placing the two shortest messages first and last, we get the optimal solution.

Now solving for 2-message case:

1. Key observation: we can view this as having 2 separate channels, and we want at most 1 intercepted message in each channel (not immediately obvious!).
2. Only thing to decide is which messages to send on which channel:
   - Channel with r msgs of lengths $t_1, \ldots, t_r$ finishes in time $x + 1 + \sum_{i=2}^{r-1} (x + 1 - t_i)$.
   - Take the 4 shortest messages and put first and last in each channel.
   - Each remaining message of length t gets a weight $w = x + 1 - t$.
3. We have $m = n - 4$ weights $w_1, \ldots, w_m$ of total weight $W = w_1 + \ldots + w_n$.
4. Want to find subset S such that $\sum_{i \in S} w_i$ is as close to $W/2$ as possible.
5. This is a Subset Sum/Knapsack problem. Classic DP approach use $O(n^2 x)$ — too slow.
6. Can be solved in $O(nx)$, using the algorithm from the paper at `https://www.sciencedirect.com/science/article/abs/pii/S0196677499910349?via%3Dihub`

Problem Tutorial: "Fortune From Folly"

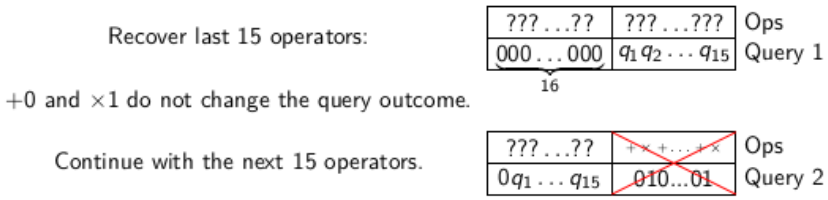
This is problem F from NCPC-2021. Problem author: Bergur Snorrason

1. At any point, only the n most recent x_i's matter.
2. Let $E_{z_1 z_2 z_3 \ldots z_n}$ be expected number of steps until k ones, if most recent x_i's are z_1, …, z_n.
   - If $\sum_{i=1}^{n} z_i \geq k$ then $E_{z_1 z_2 z_3 \ldots z_n} = 0$
   - Otherwise $E_{z_1 z_2 z_3 \ldots z_n} = 1 + p \cdot E_{z_2 z_3 \ldots z_n 1} + (1 - p) \cdot E_{z_2 z_3 \ldots z_n 0}$
3. This is a system of linear equation in the 2^n unknowns. Solve using Gaussian elimination to find out answer $E_{00\ldots0}$. Time complexity is $O(2^{3n})$. The bit strings may be represented as an n-bit integers.

Problem Tutorial: "Gyrating Glyphs"

This is problem G from BAPC-2021. Problem author: Reinier Schmiermann

- First solve the problem for 15 operators with a single query $0, q_1, \ldots, q_{15}$.
- Use this to find all operators in $20000/15 < 1400$ queries.
- Example with 30 operators:



- We consider the case with 15 operators.
- Let $0, q_1, \ldots, q_{15}$ where q_i is random in $\{1, \ldots, 10^9 + 6\}$.
- For all 2^{15} possibilities for the 15 operators compute the query outcome.
- If all outcomes are distinct $(\mathrm{mod} 10^9 + 7)$ we have a lookup table.
- If not, repeat with a new random query.

Problem Tutorial: "Hiring Help"

This is problem H from NCPC-2021. Problem author: Bergur Snorrason

- Let's switch to a geometric view:
- The coders are a set of points P in 2D.
- The consultant query is another point q in 2D.
- The weighted average exists if and only if q is below the upper side of the convex hull of P.
- Coders leaving corresponds to points being removed, leading to the convex hull changing.

So we can reformulate the problem in the following way: Given set of points (x, y), maintain upper side of its convex hull, under removals of points and queries about whether other points (x', y') are below the hull.

1. The hull is a piecewise linear function, represent it as a sorted set of points (x_1, y_1), (x_2, y_2), ..., (x_t, y_t), where $x_1 < x_2 < \ldots < x_t$ and $y_1 > y_2 > \ldots > y_t$.
2. For a query (x', y'), find index i such that $x_{i-1} < x' \leq x_i$ check if (x', y') is below line from (x_{i-1}, y_{i-1}) to (x_i, y_i).
3. Handling removals can be done, but if we instead run the events in reverse order, the removals become additions, which are easier to handle:
   - If point to add is outside current hull, add it to our current set of points.
   - Remove any concavities formed to left and right of new point.
   - Issue(?): addition may take a long time because many old points could be discarded from hull.
   - Not an issue: once a point is discarded, it can never be added back, so total number of removals for all events is not greater than n.
   - $O((n + e)logn)$ total time complexity.

Problem Tutorial: "Intact Intervals"

This is problem I from NCPC-2021. Problem author: Nils Gustaffson

First solve non-circular case:

1. Let $A_i = a_1, \ldots, a_i$ be the prefix of first i values of A.
2. Let s be the number of indices $1 \leq i < n$ such that A_i is a permutation of B_i.
3. Then the number of ways is $2^s - 1$:
   - breaking up a at any non-empty subset of these indices results in a valid separation
   - breaking at any other index results in an invalid separation
4. To find s quickly, can use a permutation-invariant hash function.
   - Assign a random hash value $h(x)$ to each array value x.
   - Define hash $h(A_i)$ of a prefix to be $\sum_{j=1}^{i} h(a_j)$.
   - If no hash collisions then A_i is a permutation of B_i if and only if $h(A_i) = h(B_i)$.

Then solution for the circular case will look like:

1. For each hash value z, let $s(z)$ be number of indices $0 \leq i < n$ such that $h(A_i) - h(B_i) = z$.
2. Then (assuming no hash collisions), the number of ways is $\sum_z 2^{s(z)} - s(z) - 1$.
   - For each z , taking any subset of at least 2 of the s(z) indices is a valid separation.
   - Taking two indices with different values of $h(A_i) - h(B_i)$ gives an invalid separation.
3. Results in an $O(n)$ time solution, assuming $O(1)$-time dictionaries which can be used to store the frequency of each hash value.

Problem Tutorial: "Jail or Joyride"

This is problem J from BAPC-2021. Problem author: Reinier Schmiermann

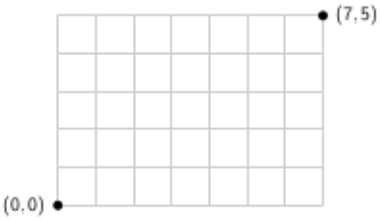
- Observation 1: If the police can approach the criminals via multiple edges, then the criminals can always reach every vertex in the graph.
  - In particular: the approach direction of the police does not matter.
  - The police should always take the shortest path.
- Observation 2: If the police can approach the criminals via only one edge, then either the criminals are in a leaf, or they are not as far away as possible from the police.
  - Second case only happens at the start.
  - After this, the criminals can always either reach the whole graph, or nothing at all.
- The police always takes the shortest path to the criminals.
- After the first approach of the police, the criminals can always either reach the whole graph, or nothing.

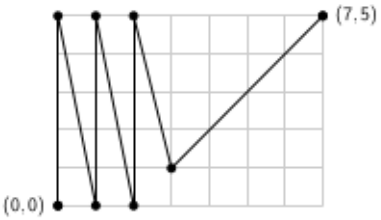
- Simulate the first approach of the police separately.
- For every vertex which is not a leaf: find all vertices which are as far away as possible (use APSP).
- Use DFS on this new directed graph to compute for every vertex v the maximal distance the police needs to travel after approaching the criminals in v.
  - If there is a reachable cycle in this new graph, the police cannot catch the criminals.

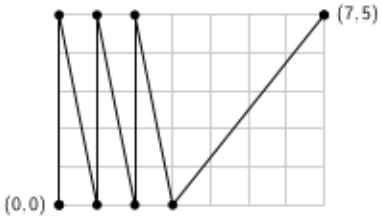
Problem Tutorial: "Kinky Cables"

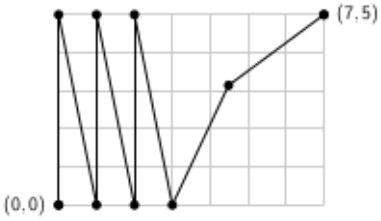
This is problem K from BAPC-2021. Problem author: Boas Kluiving

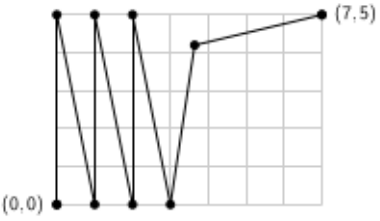
Solution 1: Zig-zag and use the binary search for last point:

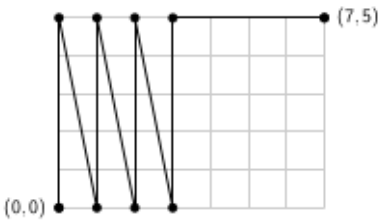


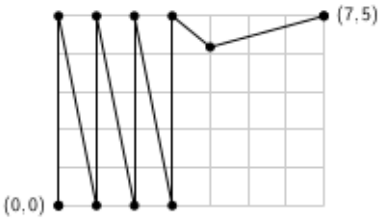




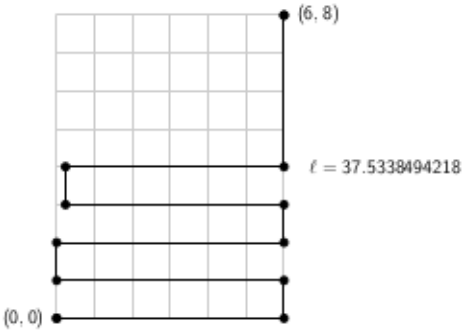








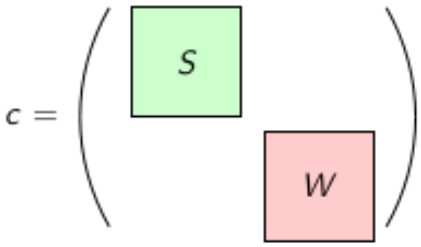
Solution 2 from BAPC-2021 team "print(math.tan(float(input())))" that created a solution without any diagonal lines and just simple arithmetic:



Problem Tutorial: "Lopsided Lineup"

This is BAPC-2021 problem L. Problem author: Jorke de Vlas

- Although the question is about synergy, the solution is actually to take strong players for the winning team.
- Take the first $n/2$ players as the strong team. Then what is the difference in scores?



$$\text{score} = \frac{1}{2}(S - W)$$

$$c = \begin{pmatrix} S & X \\ X & W \end{pmatrix} \qquad \text{score} = \frac{1}{2}(S - W)$$

$$c = \begin{pmatrix} S & X \\ X & W \end{pmatrix} \qquad \text{score} = \frac{1}{2}((S+X)-(W+X))$$

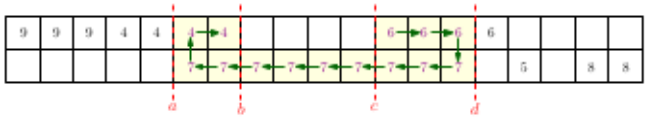
$$c = \begin{pmatrix} S+X \\ W+X \end{pmatrix} \qquad \text{score} = \frac{1}{2}((S+X)-(W+X))$$

- The score of each team is the sum of its players' row sums.
- If you take any *other* strong team, you can reorder the matrix c so that your chosen team is the first $n/2$. **That does not change the row sums!**
- **Solution:** for each player compute its strength (i.e. the sum of its row). Take the $n/2$ strongest players for the strong team, and the others for the weak team.
- Complexity: $\mathcal{O}(n^2)$.

Problem Tutorial: "Marvelous Marathon"

This is problem M from NCPC-2021. Problem author is Jimmy Mardell.

Let's formalize the problem: find integers $0 \le a \le b \le c \le d < m$ such that $2(b-a)+(c-b)+2(d-c) = x$. Maximize sum of cells in one row in range $[a, d)$ plus sum of cells in other row in ranges $[a, b)$ and $[c, d)$.



Solution outline:

- m is very large, cannot loop over all cells.
- Large parts of grid look the same because only n segments.
- Separately handle three main cases: 0, 1 or 2 U-turns
- We focus here only on the hardest case with 2 U-turns.

Insight 1:

1. We can assume solution has the gap in the lower half: run solution again on flipped input to cover opposite case
2. There is an optimal solution where a or d is at a segment endpoint (or 0 or m). Otherwise we could decrease (or increase) both a and d with 1 until either a or d reaches a segment endpoint, or $d = c$ or $a = b$, in which case we end up with the 1 U-turn case (handled separately, left as an exercise!)
3. We can assume a is the endpoint. Run solution again on reversed input to cover opposite case.

Insight 2:

1. There is an optimal solution where b or c is at a segment endpoint, for the same reasoning as before (except that this time we would show it by shifting b or c).
2. We end up with two cases to consider:
   - a and b are segment endpoints;
   - a and c are segment endpoints.
3. Will focus on the first of these; the other must also be solved, but is done in a very similar fashion.

1. Fix some a and b ($O(n^2)$ possible choices).
2. Set $c = b$ (or $c = b + 1$ if x is odd) and $d = a + \lceil x/2 \rceil$.
3. Idea: slide c and d right (c twice as fast as d) and maintain current score.
   - Since grid values rarely change value, we can slide many steps at a time.
   - Calculate when either c or d hits next segment endpoint and jump directly there.
   - Repeat until c reached the end of the road.
4. "Next segment endpoint" can be found in $O(1)$, for a total complexity of $O(n^3)$.

Problem Tutorial: "New Strategy"

This is BAPC 2021 problem B. Problem author: Harry Smit

- First set all attributes to the lowest value they need to be to pass all the challenges (if this is impossible, the maximum score is 0).
- Improve your score by increasing an attribute by one. There are two cases:
  - If the attribute score equals a of the challenge requirements, you get points equal to a times the new attribute score, plus the number of events that require a lower score for that attribute.

  - Otherwise, spending a point here gives additional score equal to the number of events that use this attribute.

- You can only spend one point on the first case (per attribute).
- Be greedy: sort these options and spend points until none are left.
- If you ever run into the second case, spend all of your points there.
- Runtime: $\mathcal{O}(n \log n + l)$.