A - A Simple Geometry Problem

This is an easy problem. The formula for deriving the new coordinates are given to you, and your job is to figure out how to use the language's built-in trigonometry functions (sine and cosine), and most likely you will have to covert degrees to radians ($rad = deg/180 * \pi$).

B - Barfoosia Tax Return

This is an ad hoc problem. You can use if-else branch to determine the total amount of tax, and take the difference between it and the withheld tax to find out the amount refunded/owed.

C - Common Mistake

Standard 0-1 knapsack problem. Only difference is after doing dp, you need to iterate through all weights and multiply the fire power with multiplier in that load range.

D - Dull Game

We notice that, at each step, if we choose the a point with x-coordinate x_0, then the maximum point can be obtained is y_0 where:

- If there exists segments that cover x_0, y_0 is the minimum height of all such segments.
- Otherwise, $y_0 = m$.

Thus, at each step, we only need to find the value of x_0 that results in the largest y_0.

This problem is a basic application of segment tree with lazy propagation. We need to construct a segment tree where each node stores the following value of some segment $[l, r](0 \leq l \leq r \leq n)$: The highest point can possibly attained if we choose $x_0 \in [l, r]$. The update and get methods can be implemented with lazy propagation. You can look at the solution code for more detail.

At each step, updating takes $O(\log n)$ while querying takes $O(1)$. Thus, the total complexity is $O(n \log n)$.

E - Enclose These Cows

Consider directed edges between all pairs of poles. We want to pick those directed edges that has all cows on its left side. Then a cycle formed by these

edges is a valid enclosing fence, and the problem reduces to finding the shortest cycle in a directed graph, which can be done with Floyd in $O(N^3)$. Overall complexity is $O(MN^2 + N^3)$.

F - Featured Animals

Placing animals from left to right is a Markov process, so we can use matrix multiplication. Let state vector have ith element representing the number of configurations when the current cage contains type i animal. Transition matrix is the one matrix except where corresponding adjacent types are forbidden, in which case the matrix element is zero. We must zero out some state vector elements in certain stages because these cages forbid some types of animals. Time complexity is $O(NM^2)$.

We can optimize with fast matrix exponentiation. This will give us time complexity $O(M^3 K \log \frac{N}{K}) = O(M^3 K \log N)$. We can further precompute base matrices, yielding time complexity $O(M^3 \log N + M^2 K \log \frac{N}{K}) = O((M + K)M^2 \log N)$.

G - Greedy String Match

First build a Trie tree from the keywords. Trie tree is a compact data structure to store a set of strings. Then follow the greedy matching rules and try match a keyword starting from each index $i \in [0, |S| - 1]$. If the matching starting from index i fails, move on to index $i+1$. Otherwise, move starting point accordingly. Complexity is $O(|S| * (lengthoflongestkeyword))$.

H - Hard Combinatorics

To solve this problem, we need to know the following combinatorics problem: The number of ways to move from $(0, 0)$ to (x, y) (where $x, y \in \mathbb{N}, x, y > 0$), allowing only moving right and up (along the axises with step size 1), is

$$\binom{x+y}{x} \tag{1}$$

Thus, for any $1 \leq i, j \leq N$,

$$\binom{A_i + B_j + C_i + D_j}{A_i + B_j} \tag{2}$$

Is the number of ways to move from $(-B_j, -D_j)$ to (A_i, C_i) giving the above constraints.

Let us denote $(-B_1, -D_1), (-B_2, -D_2), \cdots, (-B_N, -D_N)$ as starting points and $(A_1, C_1), (A_2, C_2), \cdots, (A_N, C_N)$ as ending points. Then

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\binom{A_i + B_j + C_i + D_j}{A_i + B_j} \tag{3}$$

is the number of ways to move from any starting point to any ending point giving the above constraints. This problem can be solved with dynamic programming. Let $f(x, y)$ be the number of ways to move from some starting points to (x, y). Then, we have the recursive formula:

$$f(x, y) = f(x - 1, y) + f(x, y - 1) \tag{4}$$

Hence, the complexity is $O((maxA - maxB) \times (maxC - maxD))$.

I - Impossible Task

Consider $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ where $p_1, p_2, \cdots, p_k$ are prime factors of n. Then, the sum S of all factors of n can be calculated as

$$S = (1 + p_1 + \cdots + p_1^{\alpha_1})\,(1 + p_2 + \cdots + p_2^{\alpha_2}) \cdots (1 + p_k + \cdots + p_k^{\alpha_k}) \tag{5}$$

Thus, we only need to compute p_i, α_i for all $1 \le i \le k$. Since $\sum_{i=1}^{k} \alpha_k \le \lceil \log_2 n \rceil$, with all p_i, α_i provided, S can be calculated in $O(\log n)$.

To calculate all prime factors of n and their power quickly, we can use Erastothene Sieve with a modification. While running the sieve, for each composite number, we can store any of its prime number. Then, we can factorize any number n in $O(\log n)$ (see solution code for more details).

For each n, we need $O(\log n)$ time to answer the query, thus the total complexity is $O(q \log n)$. Note: Here we excluded the complexity of running the sieve since its complexity $O(N \log N)$ (where N is the maximum value of n), in this case, is a constant.

J - Just A Hike!

We can solve the problem using binary search on the columns. Suppose we are searching in $[l, r]$. Let $mid = \lfloor \frac{l+r}{2} \rfloor$. Let

$$\begin{aligned}
h_{mid} &= argmax_{1 \le i \le m} h[i, mid] \\
h_l &= \begin{cases} argmax_{1 \le i \le m} h[i, mid - 1] & \text{if } mid - 1 \ge l \\ -1 & \text{otherwise} \end{cases} \\
h_r &= \begin{cases} argmax_{1 \le i \le m} h[i, mid + 1] & \text{if } mid + 1 \le r \\ -1 & \text{otherwise} \end{cases}
\end{aligned} \tag{6}$$

- If $h[1, h_{mid}]$ is a peak then return (mid, h_{mid}) as the result.
- If $h[1, h_{mid}] < h[1, h_r]$ (given that $h_r \neq -1$), there exists a peak in the segment $[mid + 1, r]$. Thus, we apply binary search on $[mid + 1, r]$.
- If $h[1, h_{mid}] < h[1, h_l]$ (given that $h_l \neq -1$), there exists a peak in the segment $[l, mid - 1]$. Thus, we apply binary segment on $[l, mid - 1]$.

We starts the search with segment $[1, n]$.

At each step, we need to calculate all the values in 3 columns $mid-1, mid, mid+1$. This can be done in $O(m)$. Thus, the total complexity is $O(m \log n)$.

K - Kindle the Bonfire

We can do brute force and enumerate all possible configurations to light the bonfires. After than, use Floyd's algorithm to compute pair-wise shortest path. The diameter would simply be the maximum among them.

L - Land of Fantasy

We can model this problem as a graph problem. Each bullet represents a node, and there is an edge between two nodes if and only if the two bullets overlap. Also model the left boundary and right boundary as special nodes. As last, check whether the left boundary is connected with the right boundary. If they are connected, then it's impossible to pass the battlefield. We can also use disjoint sets to simplify the modeling and speed up the process.

M - Macrosoft

This is an ad-hoc problem that requires you to know

1. How to read the input line by line
2. How to use basic language features and libraries
    - 1-D and 2-D arrays, strings, loops, built-in data structures etc.

Steps to implement the solution:

1. Read input line by line.
2. Split each line using comma (,) as delimeter. This can be done either using a library function if one is available, or using a loop manually. Store the splitted strings into a 2-D array.
3. Find the maximum length of all strings in each column. This will determine the width of that column.

4. Print out the table in ASCII. If a string in less than the corresponding column width, add additional white space characters on the left or right, depending on whether it is a number or not.

Time Complexity: O(size of input)