Problem A. League of Legends

Sum up all 5 numbers, then it is easy to know who will win.

Problem B. Restore Atlantis

For each cell (i, j) , we can find the smallest indexed rectangle and the largest indexed rectangle $min(i, j), max(i, j)$ by scan line.

Let's consider each cell from west to east, the k-th rectangle will appear at xa_k, and will disappear at xb_k. When the k-th rectangle appears at xa_k, it will cover all the cells (i, j) such that $ya_k \le j < yb_k$. We can maintain a segment tree, and insert k into $O(\log C)$ $(C \le 2000)$ nodes corresponding to $[ya_k, yb_k)$. After handling all events for the current line we are visiting, DFS the whole segment tree to find the value of $min(i, j), max(i, j)$ for the current line. Here we need to insert/delete $O(n \log C)$ numbers, and query the smallest/largest number for $O(C^2)$ times, so we can maintain `std::priority_queue` on each node of the segment tree. Thus we can do this part in $O(n \log n \log C + C^2)$.

Now for each query $[s, t]$, the answer is $\sum_{(i,j)}[(min(i, j) < s) \vee (t < max(i, j))]$, which can be easily solved by scan line and fenwick tree in $O(q \log n + C^2 \log C)$ or sqrt decomposition in $O(q\sqrt{n} + C^2)$.

Problem C. Cube

A bruteforce solution is to enumerate the order of points and check distances and angles.

It can be proved that use pairwise distances and check ratios between them can also pass this problem. And there are many other different algorithms can pass.

Problem D. Shortest Path Query

Build a trie containing the original vertices, then the original problem becomes a complete binary tree with extra ancestor-descendent edges. For a vertex i, it is impossible to travel from its left subtree to its right subtree without passing i, and vice versa.

For each vertex i and all of its descendents j, we can use Dijkstra's algorithm to preprocess the shortest path from i to j using only edges in the subtree of i.

For each query (u, v), we can enumerate all common ancestors of u and v to update the answer, $ans = \min_{z \in \text{ common ancestors of } (u,v)}(dis(z, u) + dis(z, v))$.

Total time complexity is $O(m \log^2 n + (n + q) \log n)$.

Problem E. Specially Super Rare

Assume Chenjb modifies the string for M times, then $M \approx m \times 0.003\% \approx 300$. We denote T as the reversed version of S, the answer will be the length of longest common subsequence of S and T.

Let $dp[i][j] = k$ denotes the maximum value of k such that $S[1 \dots k]$ can match $T[1 \dots k + j]$ after deleting i characters in total. Note that we can always match pairs of same characters greedily, so $dp[i][j] = dp[i][j]+LCP(S[dp[i][j]+1 \dots n], T[dp[i][j]+j+1 \dots n])$, here LCP is the longest common prefix and can be calculated by binary search and hash. And after that, we need to delete the first character in either S or T, so we can update $dp[i + 1][j + 1]$ or $dp[i + 1][j - 1]$.

The final answer is $n - \dfrac{i}{2}$, where i is the smallest number satisfying $dp[i][0] = n$.

The total complexity is $O(n + M^2 \log n)$.

Problem F. Fair Distribution

Let's denote x as the final number of energy bars, then either the total number of robots is less or equal to $\sqrt{x}$ or the number of energy bars for each robot is less or equal to $\sqrt{x}$. Then we can enumerate the smaller one to calculate the answer.

Problem G. Wall Game

Maintain the border length of each connected component by DSU.

Each conquer operation can be done by adding a 6-length component and trying to merge it with 6 adjacent honeycombs.

The coordinates are large so that we need to use `std::map` to maintain its indices.

Total complexity is $O(n \log n)$.

Problem H. Grammy and HearthStone

The original problem is equivalent to "Arrange two 1's, two 2's, ... , two n's in a row such that for each i in $[1, n]$, two i's indices differ by i."

Firstly, consider the sum of indices mod 2, a necessary condition for the existance of solution is $1 + 2 + \cdots + n = 1 + 2 + \cdots + 2n \bmod 2$, so $n = 4k$ or $n = 4k + 1$ for some integer k.

For $n = 4k$: $4k, 4k-2, \ldots, 4, 2, 4k-1, 2, 4, \ldots, 4k-2, 4k, 2k-1, 4k-3, 4k-5, \ldots, 2k+3, 2k+1$, $2k-3, 2k-5, \ldots, 5, 3, 4k-1, 2k-1, 3, 5, \ldots, 2k-5, 2k-3, 1, 1, 2k+1, 2k+3, \ldots, 4k-5, 4k-3$ is a possible solution when $n > 4$.

For $n = 4k+1$: $4k+1, 4k-2, 4k-4, \ldots, 4, 2, 4k, 2, 4, \ldots, 4k-4, 4k-2, 2k+1, 4k+1, 4k-1, 4k-3, \ldots, 2k+5$, $2k+3, 2k-1, 2k-3, \ldots, 5, 3, 4k, 2k+1, 3, 5, \ldots, 2k-3, 2k-1, 1, 1, 2k+3, 2k+5, \ldots, 4k-3, 4k-1$ is a possible solution.

For $n = 4$: $1, 1, 3, 4, 2, 3, 2, 4$ is a possible solution.

Problem I. Grammy and Ropes

For subsets of size 2 and 3, there is only 0 or 1 ropes left, so we can easily untie them.

For subsets of size 1, we only need to check whether the other two ropes are tied together by checking the corresponding intersections. More precisely, the two ropes are tied together if and only if the two ropes' intersections are "true, false" or "false, true".

For subsets of size 0, we only need to check when the ropes are pairwise untied. We denote $A > B$ if and only if rope A is on top of B at both intersections. Then the ropes are not untied if and only if $1 > 2, 2 > 3, 3 > 1$ or $3 > 2, 2 > 1, 1 > 3$.

Problem J. Grammy and Jewelry

Use breadth first search to find the distance from vertex 1 to vertex i for each i. Then we can calculate the maximum value of one piece of jewelry obtainable in t units of time for each t.

The problem now converts to complete knapsack problem.

Total time complexity is $O(m + Tn)$.

Problem K. Grammy's Kingdom

For a pair of points $(i, j), (i < j)$. The best way for transferring from i to j is to move to the first airport after i, fly to the last airport before j, and move to j at last. Assume that there are k points on this route, then the contribution to the answer is $\dfrac{n-k}{k+1}$.

At this point we only need to count the total number, $cnt[k]$, of point pairs with distance k.

Note that the airport separates the sequence into many blocks, each with $s[i]$ points inside. The contribution of two blocks $A, B (A < B)$ is a trapezoid, and only depends on $s[A]$ and $s[B]$. So we can process blocks with the same length together. There is only $O(\sqrt{n})$ different lengths since $\sum_i s[i] = n$, so the total complexity of this part is $O(n)$.

The contribution of point pairs in the same block or in adjacent blocks should be treated as special cases.

Problem L. String Freshman

Let $|T| = m$, Chenjb is correct if and only if $\forall T[i] \neq T[1](2 \leq i \leq m)$. In such case, the greedy algorithm is indeed the KMP algorithm.

Otherwise, We can easily construct $S = T[1..i-1]+T[1..m]$ where Chenjb's greedy algorithm is incorrect.

Problem M. Game Theory

Base on the definition of expected value, we only need to consider the case between Grammy and one other student. Since Grammy will select an arbitrary value, for each possibility, we calculate its result against the student's X. We will discover that no matter what the student choose for X, the summation of all results is 0. So the answer is always 0.