We hope you all enjoyed the problem statements. Since this was our(npsabari and karthik-abinav) first time setting problem statements we did have some issues with typos in testcases which led to rejudging. We apologize for that. We will try to give a brief idea about solutions we had in mind while setting the problems.

1. GuruP and his Sweetheart(APOWB)

Solution: This question boils to finding, if a number n can be expressed as a^b, where a and b are integers greater than equal to 2.
Two observations were required.

- Given a value b, we can check if there exits a integer a such that $a^b = n$ in $\log(n)$ time using a binary search
- The number of value of b is bounded by $O(\log(n))$. Hence an exhaustive search over all b's can be performed in $O(\log n)$ time.

One must be careful while performing the binary search within the correct limits for a given b to avoid overflow.

Time Complexity: $O(t * \log^3(n))$.

2. Awesome Threesome (AWTHR)

Solution: Cutting out all the story the problem was to find the following. Given a graph, partition into three sets(possibly empty) A, B, C such that A and C were complete sub graphs, while B had no internal edges. Also, for a given vertex in set B, it should have edges to atmost one of the sets A or C.

The following was a modified version of the Chocolates Division problem from Kanpur regionals - 2013.
Remove all isolated vertices. Now check if the graph has atmost 2 components. If not, output NO. If yes, on each component use a similar approach to that above mentioned problem.

A brief idea for the above mentioned problem.
Though, there were solutions which used strongly connected components, a simpler degree based analysis can do the job. Lets call Q as the set with complete subgraph,

and T as the set with no internal edges. Any vertex in set T will have degree no greater than any vertex in Q(easy to see, if not try proving). Hence, just sort the degrees and there are n ways to split the vertices into Q and T. Check if any of them satisfies the required property.

Time Complexity: $O(t * (n^2 \log n))$

3. Night at the Library (EIGVAL)

Solution: This question had a graph as input and a matrix corresponding to it defined. The problem was to check if the eigen values of the matrix had the property that the largest - the second smallest was atleast 2.

The expected solution doesnt have anything to do with eigen value decomposition algorithm(!). This matrix is well known in spectral graph theory as the normalized laplacian matrix.

Let $\lambda_0 \leq \lambda_1 \leq .. \leq \lambda_{n-1}$ be the n eigen values of the matrix. The following property holds true, $\lambda_1 = 0 \leftrightarrow$ the graph is disconnected. Also, $\lambda_{n-1} \leq 2$ for all graphs.
$\lambda_{n-1} = 2 \leftrightarrow$ there exists atleast one non-trivial bipartite component. Non-trivial implies there exists atleast one edge in the component. Therefore, for graphs which are disconnected and has atleast one non-trivial bipartite component the required difference is atleast 2.

We will let the reader explore the proofs for these above statements. As a reference to topics in spectral graph theory we suggest the monograph Lx=b.

Time Complexity: $O(t * (m + n))$

4. Dreamer's E-trip (GRETHIEF)

Solution: This can be modeled as min cost maximum flow problem. The problem has two parts. Given the network of M directed roads and N cities, first find the minimum time at which the network can be decomposed into disjoint cycles. Second part involves finding the maximum of min cost decompositions over a range of time.

Build a bipartite graph $G^{'}$ where every vertex of the original graph $G = (V, E)$ has two copies (one on left part and the other on right). If there is an edge from node u to node v in G, then draw an edge from node u on the left copy to the node v on the right copy of $G^{'}$. G can be decomposed into disjoint cycles iff $G^{'}$ has a perfect matching. Now, the problem of finding the min cost decomposition of G is reduced to finding min cost weighted matching of $G^{'}$.

Part 1: To check whether there exists a perfect matching, we can use either Hopcroft-Karp or Hungarian algorithm. It is clear that once a perfect matching is formed, $G^{'}$ always has a perfect matching (as $b_i \geq 0$). As this function is monotone, we can binary search for minimum time t($\leq 10^9$) by setting the cost[i][j] to be 0 if the road is construted otherwise INF.

Part 2: Here, the roads that are constructed has non-increasing tax function. Hence the min cost weighted matching of $G^{'}$ is non-increasing function of time t. Now, we again binary search on t, to find the largest min cost weight matching that is less than the given constant. Hungarian algorithm is required as the edges that are part of the perfect matching may change over time.

Time Complexity: $O(t * (n^3 + n^3) \log T_{max}))$

5. New York Magnets (SHAMASS)

Solution: This was one of the simpler problems. The question was given N bodies starting at various times, the time it takes to reach distance d. The only catch being if it comes next to a body which went before it, it will have the speed of that body. Also, every body has a maximum speed it can reach.

This can be solved using some basic equations of motion.
Note that, the time for a particular object to reach destination is maximum of time for it to reach based on equations of motion and time taken by the object that left before it.

To calculate the time based on equation of motion, there can be two cases:

- The object didint reach its maximum velocity. In this case, time is $\sqrt{(\frac{2d}{a})}$
- The object reached its maximum velocity mid way. In this case find the distance

when it reached its maximum velocity and the time from there till destination using this constant maximum velocity. (use motion equations)

Time Complexity: $O(t * n)$

6. Art of Throwing (SHASFUN)

Solution: This problem involves just simulating the problem statement. Given a normal form game, we need to report the first Nash equilibria. The payoff matrix $(n * m)$ of the game can be obtained using modular exponentiation of the given values.

A given action set (A_1, A_2) is in nash equilibrium iff the actions of both the players are their best responses for the other player's action. This is same as saying that "At nash equilibrium, any unilateral deviation by a player would not result in greater payoff for that player".

We compute the best response for *player i* given the action of *player j*. Note that there can be multiple best responses for a given action. Now just iterate over the normal form game row by row and check whether the given action set are their mutual best responses.

Time Complexity: $O(t * (n * m))$

7. Curious Case of Vatsan Mama(SHASGAME)

Solution:

The problem involves determining whether a given state of the modified nim game is a winning position for player 1. We can use grundy numbers to solve this problem. In this game, a player can add atmost $\left\lfloor (\frac{p_i - c_i}{j}) \right\rfloor$ coins to the pile i. Now, this is same as the classical n-piles problem. In other words, player's moves can be viewed as, removing coins from the piles with a upper bound of $\left\lfloor (\frac{k_i}{j}) \right\rfloor$ where $k_i = p_i - c_i$.

Now, the grundy number of each pile can be calculated using the recursive function.

$$grundy(k_i, j) = \begin{cases} \frac{k_i}{j} & \text{if } k_i \text{ is divisible by j} \\ 0 & \text{if } k_i < j \\ grundy(k_i - k_i/j - 1) & \text{otherwise} \end{cases}$$

Using the theorem on grundy numbers, which states, "a state is in losing position iff the xor's of grundy numbers of all the piles results in 0", we can check whether the given set of piles favor player 1.

To find all winning subarrays of piles, we can find the cumulative xor-array B of grundy numbers. We can see that if $B[i] = B[j]$, then xor of all the grundy numbers from i+1 to j is 0. So, this boils down to counting number of times a given number occurs in B. For example if a number x occurred y times, then there are ${}^{y}C_2$ subarrays with xor sum 0. Special case is when x is 0.

Time Complexity: $O(t * (n * \log k_{max}))$

8. Good Boy Eshwar (SHASGRA)

Solution: In this problem we just have to find number of perfect matchings possible in the given bipartite graph. This can be solved using two methods. First involves use of dynamic programming with bitmask. dp[mask] represents the number of matchings possible using the bits set in the mask. This dp can be recursively or iteratively calculated for all subsets of boys.

The other method is to just find the permanent of the adjacency matrix. This again involves finding the number of ways to match i boys with N girls by iterating over all subsets. Though this method is slower than the dp method, the time limit set high enough for this approach to pass.

Time Complexity: $O(n^2 * 2^n)$

9. No Strings Attached(SHASTR)

Solution: The question is to find, given two strings, the length of the maximum common subsequence such that a K length substring of both s1 and s2 is removed.

The question involved using dynamic programming approach to precompute longest common subsequences and iterate through all possible substrings that can be removed from string1 and string 2.(There were multiple approaches all of which used

some form of dynamic programming).

Define a dp[string1_size][string2_size][2][2] matrix to precompute for all possible lengths of string1 and string2 and directions of both forward and backward(the third and fourth dimension of this matrix). Using this we can query for a given substring of sring1 and string2 removed in constant time.

Time Complexity: $O(t * (n^2 + m^2))$.