1001. AND Minimum Spanning Tree

This is a very simple problem. If N is 2^k-1, the cost of MST is 1, otherwise it's 0. Lexicographically smallest solution is also trivial. 2*k's parent is 1, 4*k+1's parent is 2, 8*k+3's parent is 4, and so on. If N is 2^k-1, only N's parent is not determined after this process. In this case, N's parent is 1.

1002. Colored Tree

Let v and c be the index and color of given vertex.
If the sub-tree which is rooted at vertex v has another vertex v′ of color c, there is no need to perform any operations.
Find the nearest ancestor of v whose sub-tree has any vertices of color c and perform erase operations using heavy light decomposition.
Let u be the nearest ancestor of v! The route between u and v split into $n^{1/2}$ pieces. Each piece corresponds to an array of n values and i^{th} of them is the number of sub-trees (its root is in this piece) with exactly i distinct colors.
Inserting color is the same as described above.
Time complexity is $O(n^{3/2}\log n)$.

1003. Divide the stones

If the total number of stones is a multiple of k, we can always divide the stones. Otherwise, we can't.

If $\frac{\text{ढ}}{\text{ग़}}$ is even, you can find solution quite easily.

If $\frac{\text{ढ}}{\text{ग़}}$ is an odd number, we divide first 3k stones into k groups with exactly 3 stones and equal weight.
After that, you can divide remaining stones by the way you did when $\frac{\text{ढ}}{\text{ग़}}$ is even.

Time complexity is O(n).

1004. Enveloping Convex

Let’s think the result convex is Q(Q1, Q2,...Qn in anti-clockwise order).
Then every point q1, q2, ... qm must be on the left side of every line QiQi+1.(i = 1,2,...n, Qn+1 = Q1)
For every line QiQi+1, we get a point Ti (Ti is one of q1,q2,... qm) satisfies that all qi is on the left side of QiQi+1 if and only if Ti is on the left side of QiQi+1.
Then by using binary search algorithm, we check if there is a point on the left side of all lines (Qi –ans* Ti, Qi+1 –ans*Ti) (i = 1,2,... n).
You must consider the inversed polygon of P.
So the time complexity is O(2 * n * Log(Precision) + m * log(m)).

1005. Good Numbers

It can be solved by dynamic programming. Let us calculate the numbers such that the remainder modulo P are s, and the occurrence of the 8 digits modulo 3 are t_0, t_1, ..., t_7 respectively. Let’s denote it as f[K][s][t], here t=t_7*3^7+t_6*3^6+...+t_0*3^0. Then the answer is f[K][0][0].
We can easily derive all f[2*K][s][t] from all f[K][i][j] by 3^16*P*P , so this task can be done simply by O(3^16*P*P*log(K)). But it’s too slow, it will receive time limit exceeded verdict. The key point of this problem is to reduce the complexity of the convolution* of 2 arrays with size N.
For every two ternary numbers x= x_7*3^7+x_6*3^6+...+x_0*3^0, y=y_7*3^7+y_6*3^6+...+y_0*3^0, let’s define
g(x,y)=((x_7+y_7)mod3)
*3^7+((x_6+y_6)mod3)*3^6+...+((x_0+y_0)mod3)*3^0.

Then we must compute c[i] = $\sum_{ख,ग=क, 0\le ख,ग<6561} अ[ख] * ब[ग]$, for every

i<6561. MOD=1e9+9 is 3*k+1 type prime number, it can be solved by O(Nlog(N)) time(here N = 3^8), like binary case(this is ternary case).
So we can reduce total time complexity O(T*3^8*P*P*log(K)).

1006. Horse

This problem is separated into two sub-problems.
First sub-problem:
Select M trees – p1, p2, p3, …pM.
And maximize the sum of $h_{pi} * (n - p_i + 1)$.
This can be solved in O(MN)
Secondary sub-problem:

Split N trees into at most K + 1 segments.

Suppose that the weight of a segment is the sum of $h_L + (h_L + h_{L+1}) + \ldots (h_{L+1} + h_{L+2} + \ldots + h_R)$: here L is the left of the segment and R is the right of the segment.

And minimize the sum of weight of all segments.

This can be solved in O(KN) using speeding up dynamic programming.
So final answer is ANS1 - ANS2.
Totally this problem is solved in O((M + K) * N).

1007. Just an Old Puzzle

The solution consists of three steps.
At first you have to match the numbers 1, 2, 3 and 4.
To match these numbers, we only need the positions of 1, 2, 3, 4 and empty grid.
So the total number of possible states are P_16^5=524160.
You can use BFS or any algorithms to find it.
Next, you have to match the numbers 5, 6, 7 and 8.
And at last, you have to match the numbers 9~15.
The total possible statuses are 8! = 40320.
You can check if you could find the solution by checking the parity of inversion number of input permutation.
You can easily prove that in first step, the maximum distance to target status are 46.
By using similar way, you can prove that you can match the grid in 120 moves.

1008. K-th Closest Distance

Using segment tree, we can find the number of values smaller than p

in [L, R] within O(log(n)).
So by using binary search method, we can find the answer in O(log(n)^2) time.
Total time complexity is O(N log(N) + Q log(N)^2).

1009. Linear Functions

A_i+t*B_i mod P = A_i+t*B_i-k_t*P, here k_t = [(A_i+t*B_i)/P].
If we mark all t, k_t>k_{t-1}, we can solve the task in O(sum_{i=1}^{N}{K/P*B_i}) time. With probability 1/2, B_i>=P/2, and in this case, it spends a lot of time. So in such cases, if we use A_i+t*B_i mod P =A_i-t*(P-B_i) mod P, we can improve the solution 2x faster. But it's too slow yet. Let's introduce some parameter G=O(sqrt(n)), and then define b_i = min{j=1}^{G}{min(B_i*j mod P, (P-B_i)*j mod P)}. Also let's denote the optimal j as g_i, i.e. b_i=min(B_i*g_i mod P, (P-B_i)*g_i mod P).
Then we split the array into G groups by g_i. Also in each group, we split t by modulo the same g=g_i, and we can process in this subgroup O(K/g+sum{i=1}^{N}(K/g/P*b_i)) by the same method we introduced above. Then total time complexity is O((N+K)*G+sum{i=1}^{N}(K /P*b_i)) =O((N+K)*G+K/G*N) in random cases. So the expected running time is O(sqrt(N)*K).

1010. Minimal Power of Prime

Let's first factorize N using prime numbers not larger than $N^{1/5}$. And let's denote M as the left part, all the prime factors of M are larger than $N^{1/5}$. If M=1 then we are done, otherwise M can only be P^2, P^3, P^4 or P^2*Q^2, here P and Q are prime numbers.
(1) If $M^{1/4}$ is an integer, we can know that $M=P^4$. Update answer using 4 and return.
(2) If $M^{1/3}$ is an integer, we can know that $M=P^3$. Update answer using 3 and return.
(3) If $M^{1/2}$ is an integer, we can know that $M=P^2$ or $M=P^2*Q^2$. No matter which situation, we can always update answer using 2 and return.
(4) If (1)(2)(3) are false, we can know that answer=1.
Since there are just $O(N^{1/5}/\log(N))$ prime numbers, so the expected

running time is $O(T*N^{1/5}/\log(N))$.