1170 – Counting Perfect BST

BST is the acronym for Binary Search Tree. A BST is a tree data structure with the following properties.

i) Each BST contains a root node and the root may have zero, one or two children. Each of the children themselves forms the root of another BST. The two children are classically referred to as left child and right child.

ii) The left subtree, whose root is the left children of a root, contains all elements with key values less than or equal to that of the root.

iii) The right subtree, whose root is the right children of a root, contains all elements with key values greater than that of the root.

An integer m is said to be a perfect power if there exists integer **$x > 1$** and **$y > 1$** such that **$m = x^y$**. First few perfect powers are {4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 81, 100, 121, 125, 128, 144, ...}. Now given two integer **a** and **b** we want to construct BST using all perfect powers between **a** and **b**, where each perfect power will form the key value of a node.

Now, we can construct several BSTs out of the perfect powers. For example, given **a = 1** and **b = 10**, perfect powers between **a** and **b** are 4, 8, 9. Using these we can form the following five BSTs.

```
4         4       8        9       9
  \        \     /  \     /       /
   8        9   4    9   4       8
     \     /              \     /
      9  8                 8  4
```

In this problem, given **a** and **b**, you will have to determine the total number of BSTs that can be formed using perfect powers between **a** and **b**.

Input

Input starts with an integer **T (≤ 20000)**, denoting the number of test cases.

Each case of input contains two integers: **a** and **b** **$(1 \le a \le b \le 10^{10}, b - a \le 10^6)$** as defined in the problem statement.

Output

For each case, print the case number and the total number of distinct BSTs that can be formed by the perfect powers between **a** and **b**. Output the result modulo **100000007**.

Sample Input	Output for Sample Input

4
1 4
5 10
1 10
1 3
	Case 1: 1
	Case 2: 2
	Case 3: 5
	Case 4: 0