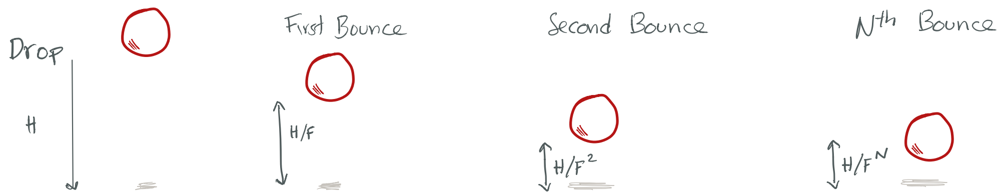
The Physics Class

Today's physics lesson involves bouncing and basic concepts of physics. For the class, a rubber ball will be used. This specific rubber ball, if dropped from a height H, then it bounces back to a maximum height H/F, when it bounces again, it reaches H/F^2, then in the third bounce, its height is H/F^3. See the pattern?



The class has N children present. The teacher wants to have two students that when the tallest of them drops a ball, its bounce height is the same as the shorter child after some number of bounces.

The problem: get how many ways are there?

The Input

- First line contains the number of test cases T.
- Then, $2T$ lines follow, 2 lines per testcase.
  - N F → Size of classroom, F
  - Height[0] Height[1] . . . Height[N] → Array of students' heights.

The Output

- Just one line of output with the number of possible combinations.

Some constraints

$1 \leq T \leq 100$		$1 \leq T \leq 100$
$1 \leq N \leq 1000$	OR	$1 \leq N \leq 10000$
$2 \leq F \leq 10^9$		$2 \leq F \leq 10^9$
$1 \leq Height[i] \leq 10^9$		$1 \leq Height[i] \leq 10^9$

A possible solution

As we're dealing with arrays with numbers, it could be wise to sort the array. It will greatly reduce complexity. This will be implemented as a quicksort with the algorithm library, as seen in class.

Once the array is sorted, we can simply iterate over it, and evaluate the possible height of the bounce → H/F^i, where i is the current iteration. Then, we use this calculated value to see how many students with that height. As height is an array of integers, then we need to use values that are **ONLY** integers.

Possible Implementation

```
H[] = {152, 162, 173, 151, 160, 170, 159};
qsort ( H, n, sizeof(int), cmpfun );          // Ascending Order          O(n·log n)
for (int i = 0; i<n; i++){                     // Iterate over H
    float x = H[i] / pow (F, i);
      if x is integer
          search within H[] for x                                        O(log n)
              count ++;
return count;  → Number of combinations
```