***Trip Game***

C371\_Coding\_March2023

**Topic**: Searching Algorithms

**Difficulty Level:** Hard

**Question / Problem Statement**:

Eleanor and Bella are on their way to Las Vegas. On their trip they visit a lot of different places. But one place left Eleanor awe struck. She saw **N** numbersof round big stones laid in a straight line with a number encrypted on each of the **N** stones given in array **A**. She called this number the value of the stones.

Bella converted the present environment to a problem. She labelled all the stones from 1 onwards. She then asked Eleanor to tell her the total number of triplets such that the label of 1st stone < label of 2nd stone < label of 3rd stone and the product of value of 1st stone, value of 2nd stone and value of 3rd stone is less than or equal to a given value **K** i.e. (value of 1st stone) \* (value of 2nd stone) \* (value of 3rd stone) *<=* ***K***.

Write a program to calculate the total number of triplets which satisfies the above condition.

**Note**

1. Two triplets are considered different if the label value's of both the triplets are not the same.

2. The number on the stone needs not to be unique.

**Function Description**

In the provided code snippet, implement the provided **tripGame(...)** method using the variables to calculate the total number of triplets which satisfies the above condition. You can write your code in the space below the phrase **“WRITE YOUR LOGIC HERE”**.

There will be multiple test cases running so the Input and Output should match exactly as provided.  
The base Output variable **result** is set to a default value of **-404** which can be modified. Additionally, you can add or remove these output variables.

**Input Format**

The first line contains 2 integers **N** indicating the number of stones and the given value **K.**

The second line contains **N** space separated integers denoting array **A,** the value on the encrypted stone. The first value denotes the value of stone labelled 1 the second value for stone labelled 2 and so on.

**Sample Input**

4 42 –denotes **N** and **K**.

3 2 5 7 –denotes array **A**

**Constraints**

1 <= **N** <= 2000.

1 <= **A[i]** <= 10^6.

1 <= **k** <= 1000001.

**Output Format**

Output the total number of triplets which satisfies the above condition.

**Sample Output**

2

**Explanation**

1. Consider the triplet (3, 2, 5):

Here 2 < 3 < 5 and also 2\*3\*5 <= K which is 42.

2. Similarly (3, 2, 7) is also a triplet that satisfies the above condition.

So, a total of 2 triplets are possible.

**Solution Steps**

1. Observation: The order in which the stones are kept on the ground is not relevant, only the values on the top of the stones in counting the number of required triplets is. That is true because given any triplet, one can always rearrange them in increasing order of labels such that the required property of product of values of those 3 is unchanged.

2. Now using this observation, the problem reduces to finding the number of triplets out of N numbers such that the product of their values is ≤ K.

3. Subproblem: Given a sorted array of integers every pair of integers *(a, b), (a ≤ b)*, how many integers *c* are there such that *abc ≤ K.*

4. Solution: Given that it's a sorted array, we need to find the number of integers between index of *b* and the end of the array such that the value of each of those is less than or equal to *K/(ab). This can be done by doing a binary search from index b* to the end of the array and finding the index of the largest value in the array such that the value is *≤ K/(ab).*

*All the integers between index b* and this integer will also satisfy that their value is *≤ K/(a\*b)* (given that this is a sorted array) and they will all form valid triplets with the pair (a, b).

6. This way we can count the number of integers for each pair and sum them up to get the final answer.

**Running Solution in C++** :

#include <bits/stdc++.h>

using namespace std;

#define MAXN 2000

#define MAXK 1000001

vector<long long> ar;

vector<long long>::iterator up;

int main(){

int N, K;

long long t;

scanf("%d %d", &N, &K);

for(int i=0;i<N;i++){

scanf("%lld", &t);

ar.push\_back(t);

}

sort(ar.begin(), ar.begin()+N);

long long ans = 0;

for(int i=0;i<N;i++)

for(int j=i+1;j<N;j++){

//binary search

long long f = (long long)K / ar[i];

f = f/ar[j];

up = upper\_bound(ar.begin(), ar.end(), f);

if(up-ar.begin()>j)

ans = ans + (up - (ar.begin()+j) - 1);

}

printf("%lld\n", ans);

}

Input:

3 30

8 2 1

Output:

1

**Test Cases [ Qty: 12 ]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case No** | **Input** | **Output** | **Score** |
| 1 | 4 42  3 2 5 7 | 2 | 0 |
| 2 | 3 30  8 2 1 | 1 | 0 |
| 3 | 100 19132  42 57 38 92 71 42 42 72 55 47 67 46 69 83 30 21 36 42 71 33 72 36 29 70 48 5 49 51 99 50 72 18 33 29 16 77 8 28 45 70 54 95 60 95 8 95 68 51 43 13 7 69 22 88 19 51 72 34 41 29 20 14 37 59 58 38 52 99 4 93 70 84 25 19 24 72 36 79 5 62 92 62 100 7 14 99 74 14 8 10 95 19 93 53 2 10 29 12 28 35 | 38526 | 1 |
| 4 | 500 19511  73 17 34 67 20 38 26 72 75 90 77 83 31 2 100 48 42 97 88 12 21 52 23 47 15 97 85 87 71 12 34 17 61 34 57 68 64 38 5 72 66 26 7 85 48 73 7 16 28 47 63 37 77 94 10 18 77 70 44 66 93 55 43 9 60 85 97 43 56 61 79 76 86 94 85 3 28 38 87 82 30 14 83 16 88 100 48 48 45 11 63 57 5 4 64 11 54 19 7 65 49 11 87 19 19 66 55 98 22 4 3 17 27 56 45 58 96 72 49 38 82 80 78 73 31 49 78 6 16 30 59 41 70 92 42 61 10 63 49 1 91 48 20 50 59 7 29 75 53 84 89 71 49 58 48 34 3 52 68 66 21 96 77 56 28 44 56 51 2 87 82 85 6 76 42 32 95 89 3 54 85 2 51 64 100 35 83 99 34 68 56 24 65 43 41 34 74 67 81 65 96 55 27 66 19 61 5 40 66 32 23 24 2 100 88 84 53 88 45 1 80 52 32 68 77 49 46 6 4 98 37 34 54 4 73 95 90 11 85 41 28 90 28 21 23 41 21 92 20 78 87 83 39 94 21 9 80 37 77 13 26 3 80 47 38 80 30 36 28 94 93 24 1 62 47 84 13 29 40 40 3 43 77 94 29 49 87 16 90 47 6 66 61 15 1 16 100 49 94 90 84 51 79 73 70 28 60 33 11 60 73 99 72 73 39 22 69 1 30 84 20 2 89 86 72 67 65 44 86 93 83 1 54 80 57 91 7 84 35 57 73 66 86 6 23 7 48 30 32 2 85 62 76 8 70 100 35 21 4 27 42 74 31 84 47 19 84 6 50 46 24 56 63 79 13 80 78 64 2 22 29 72 26 57 93 84 13 9 39 43 54 14 3 71 96 32 14 66 84 76 33 74 34 65 90 59 24 2 23 50 47 43 52 91 27 70 100 34 33 57 15 91 55 49 38 44 84 9 87 23 48 6 47 44 26 65 9 70 25 76 5 16 88 45 37 43 33 89 55 43 35 59 39 31 78 63 38 5 59 86 75 53 10 35 16 84 12 43 21 4 92 97 97 62 12 19 33 77 60 25 77 77 39 72 100 86 72 81 59 9 3 43 97 20 68 17 57 2 24 24 | 5456617 | 1 |
| 5 | 7 56456  323 232 34 3434 44 23 45 | 3 | 1 |
| 6 | 10 7  1 2 9 323 232 34 3434 44 23 45 | 0 | 1 |
| 7 | 17 730  3 5 1 6 10 8 10 4 4 4 7 6 2 7 7 6 9 | 678 | 1 |
| 8 | 26 730  3 36 2 7 1 9 1 9 3 5 1 6 10 1 8 10 4 4 4 7 6 2 7 7 6 9 | 2440 | 1 |
| 9 | 1079 730  5 7 8 4 6 1 7 6 10 1 3 9 6 6 2 2 9 7 8 5 10 8 10 6 5 3 10 4 9 4 5 3 4 6 10 8 7 2 7 4 4 6 8 6 1 3 3 5 8 4 2 8 4 4 1 8 2 2 5 2 5 4 4 7 5 10 10 10 3 9 2 5 5 2 10 3 7 5 9 1 6 7 4 4 10 4 2 2 8 9 2 10 6 3 8 9 1 2 1 2 1 3 2 9 10 3 6 10 5 4 4 2 4 3 8 2 5 6 3 6 2 5 1 10 6 1 2 5 7 2 5 4 2 6 6 2 3 2 2 10 6 6 3 9 6 6 7 4 10 5 7 1 9 3 2 3 10 1 6 10 10 10 1 10 4 8 6 1 1 6 10 3 7 1 4 8 9 1 9 9 2 9 4 9 4 7 9 6 10 6 3 6 1 8 5 7 6 5 8 5 1 10 6 5 1 7 1 5 8 3 9 5 2 4 2 8 4 6 2 9 4 5 1 10 2 2 8 9 3 1 6 5 10 3 8 7 3 4 8 9 10 5 10 6 9 10 7 1 4 8 1 6 7 10 4 2 4 2 10 10 10 10 8 4 7 9 7 7 2 2 1 10 2 3 2 1 6 10 5 7 4 8 10 1 10 1 4 8 10 2 3 1 5 2 10 7 1 8 7 10 2 9 10 2 4 1 3 6 10 6 6 5 7 3 2 8 5 7 1 3 5 10 3 3 10 7 6 1 3 9 5 2 8 9 1 6 1 3 1 2 3 7 10 7 8 2 1 10 8 7 2 9 10 10 2 4 7 7 4 5 10 3 4 5 3 6 10 10 2 7 1 1 1 9 8 6 8 10 8 2 1 7 5 3 2 8 9 7 6 9 7 7 9 4 4 5 3 6 5 6 3 9 9 6 5 7 6 7 5 2 7 3 5 10 1 6 6 9 3 9 4 7 9 9 4 10 3 8 9 6 10 7 8 5 6 6 6 9 3 10 1 5 5 7 8 7 4 5 1 4 10 7 3 1 8 7 1 7 4 5 1 3 10 5 1 2 3 1 7 9 4 1 4 6 9 4 6 9 9 3 3 7 5 9 10 10 6 1 2 5 2 8 2 9 2 10 6 4 10 3 8 4 8 10 4 3 10 7 5 6 10 3 2 2 5 6 7 6 6 8 5 10 7 6 8 6 5 10 7 9 2 5 9 6 4 4 1 2 3 10 4 2 10 3 6 2 5 3 6 9 2 1 6 4 4 6 2 5 8 5 8 2 10 4 2 1 9 8 9 8 9 2 9 8 3 3 7 3 7 3 5 6 3 2 1 6 7 9 6 8 8 5 9 5 5 1 9 6 6 1 3 1 9 2 4 3 8 8 9 7 6 4 9 1 9 4 5 5 10 7 4 7 1 4 9 10 6 8 2 7 2 5 7 4 1 2 1 3 10 1 9 2 3 2 6 6 1 2 6 3 3 4 2 5 2 9 8 10 5 1 6 10 9 8 8 3 6 3 2 8 2 10 9 3 1 6 3 1 6 2 8 8 8 3 10 3 5 1 8 7 7 6 7 10 10 2 7 8 9 8 5 3 9 10 2 5 7 7 5 5 9 7 4 4 4 3 2 7 2 5 5 4 8 8 4 6 5 3 4 2 1 3 8 8 5 4 5 5 2 3 10 5 9 9 6 8 1 5 2 7 6 3 1 9 8 5 4 4 7 3 9 3 3 1 5 3 1 8 8 8 5 7 7 1 1 8 4 6 3 10 10 1 2 4 4 10 2 9 9 2 6 6 8 1 4 1 2 6 6 3 7 1 2 6 9 5 10 10 4 8 8 8 5 9 4 2 4 6 1 9 5 8 1 4 5 1 7 9 8 2 1 10 3 2 8 6 5 3 8 8 1 5 7 3 3 6 6 7 1 1 10 3 5 2 9 9 7 9 4 4 9 7 4 7 3 9 1 10 2 4 6 8 5 8 7 2 9 9 3 1 5 9 2 2 8 7 8 2 3 1 8 6 7 5 5 1 7 10 5 3 2 5 9 6 2 7 5 8 8 2 7 1 1 3 9 9 4 8 3 6 1 10 1 6 10 3 2 3 2 8 3 9 6 8 6 5 5 6 2 7 1 9 1 9 3 5 1 6 10 1 8 10 4 4 4 7 6 2 7 7 6 9 43 4 5 67 34 3 4 4 5 6 7 3 3 4 5 5 55 5 | 205953421 | 1 |
| 10 | 50 19132  42 57 38 92 71 42 42 72 55 47 67 46 69 83 30 21 36 42 71 33 72 36 29 70 48 5 49 51 99 50 72 18 33 29 16 77 8 28 45 70 54 95 60 95 8 95 68 51 43 13 | 2304 | 1 |
| 11 | 1000 10021  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 166167000 | 1 |
| 12 | 888 5465  6 5 3 10 4 9 4 5 3 4 6 2 2 5 2 5 4 4 7 5 9 1 6 7 4 4 10 4 2 2 8 9 2 10 6 3 8 9 1 2 1 2 1 3 2 9 10 3 6 10 5 4 4 2 4 3 8 2 5 6 3 6 2 5 1 10 6 1 2 5 7 2 5 4 2 6 6 2 3 2 2 10 6 6 3 9 6 6 7 4 10 5 7 1 9 3 2 3 10 1 6 10 10 10 1 10 4 8 6 1 1 6 10 3 7 1 4 8 9 1 9 9 2 9 4 9 4 7 9 6 10 6 3 6 1 8 5 7 6 5 8 5 1 10 6 5 1 7 1 5 8 3 9 5 2 4 2 8 4 6 2 9 4 5 1 10 2 2 8 9 3 1 6 5 10 3 8 7 3 4 8 9 10 5 10 6 9 10 7 1 4 8 1 6 7 10 4 2 4 2 10 10 10 10 8 4 7 9 7 7 2 2 1 10 2 3 2 1 6 10 5 7 4 8 10 1 10 1 4 8 10 2 3 1 5 2 10 7 1 8 7 10 2 9 10 2 4 1 3 6 10 6 6 5 7 3 2 8 5 7 1 3 5 10 3 3 10 7 6 1 3 9 5 2 8 9 1 6 1 3 1 2 3 7 10 7 8 2 1 10 8 7 2 9 10 10 2 4 7 7 4 5 10 3 4 5 3 6 10 10 2 7 1 1 1 9 8 6 8 10 8 2 1 7 5 3 2 8 9 7 6 9 7 7 9 4 4 5 3 6 5 6 3 9 9 6 5 7 6 7 5 2 7 3 5 10 1 6 6 9 3 9 4 7 9 9 4 10 3 8 9 6 10 7 8 5 6 6 6 9 3 10 1 5 5 7 8 7 4 5 1 4 10 7 3 1 8 7 1 7 4 5 1 3 10 5 1 2 3 1 7 9 4 1 4 6 9 4 6 9 9 3 3 7 5 9 10 10 4 3 10 7 5 6 10 3 2 2 5 6 7 6 6 8 5 10 7 6 8 6 5 10 7 9 2 5 9 6 4 4 1 2 3 10 4 2 10 3 6 2 5 3 6 9 2 1 6 4 4 6 2 5 8 5 8 2 10 4 2 1 9 8 9 8 9 2 9 8 3 3 7 3 7 3 5 6 3 2 1 6 7 9 6 8 8 5 9 5 5 1 9 6 6 1 3 1 9 2 4 3 8 8 9 7 6 4 9 1 9 4 5 5 10 7 4 7 1 4 9 10 6 8 2 7 2 5 7 4 1 2 1 3 10 1 9 2 3 2 6 6 1 2 6 3 3 4 2 5 2 9 8 10 5 1 6 10 9 8 8 3 6 3 2 8 2 10 9 3 1 6 3 1 6 2 8 8 8 3 10 3 5 1 8 7 7 6 7 10 10 2 7 8 9 8 5 3 9 10 2 5 7 7 5 5 9 7 4 4 4 3 2 7 2 5 5 4 8 8 4 6 5 3 4 2 1 3 8 8 5 4 5 5 2 3 10 5 9 9 6 8 1 5 2 7 6 3 1 9 8 5 4 4 7 3 9 3 3 1 5 3 1 8 8 8 5 7 7 1 1 8 4 6 3 10 10 1 2 4 4 10 2 9 9 2 6 6 8 1 4 1 2 6 6 3 7 1 2 6 9 5 10 10 4 8 8 8 5 9 4 2 4 6 1 9 5 8 1 9 7 9 4 4 9 7 4 7 3 9 1 10 2 4 6 8 5 8 7 2 9 9 3 1 5 9 2 2 8 7 8 2 3 1 8 6 7 5 5 1 7 10 5 3 2 5 9 6 2 7 5 8 8 2 7 1 1 3 9 9 4 8 | 116310536 | 1 |

Plagiarism found – No

Clarity of the problem statement - Yes

Clarity of the example in the problem statement - Yes

Clarity of sample test cases - Yes

Clarity of test cases (Dual output) – Yes

Clarity of explanations - Yes

Provided Solution running – Yes

EEOC complaint (using abusive words/Indian Names/) - No

Similar Question in System - No

Difficulty Level – Hard

Question w.r.t searching algorithms concepts- Yes

Final Comment: **Accepted**