Non-Divisible Subset

locked

by zxqfd555

Problem | Submissions | Leaderboard | Discussions | Editorial

Given a set of distinct integers, print the size of a maximal subset of S where the sum of any 2 numbers in S' is *not* evenly divisible by k

For example, the array $S = [19, 10, 12, 10, 24, 25, 22]$ and $k = 4$. One of the arrays that can be created is $S'[0] = [10, 12, 25]$. Anot $S'[1] = [19, 22, 24]$. After testing all permutations, the maximum length solution array has 3 elements.

Function Description

Complete the *nonDivisibleSubset* function in the editor below. It should return an integer representing the length of the longest subset meeting the criteria.

nonDivisibleSubset has the following parameter(s):

- *S*: an array of integers
- *k*: an integer

Input Format

The first line contains 2 space-separated integers, n and k, the number of values in S and the *non* factor.
The second line contains n space-separated integers describing $S[i]$, the unique values of the set.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq k \leq 100$
- $1 \leq S[i] \leq 10^9$
- All of the given numbers are distinct.

Output Format

Print the size of the largest possible subset (S').

Sample Input

```
4 3
1 7 2 4
```

Sample Output

```
3
```

Explanation

The sums of all permutations of two elements from $S = \{1, 7, 2, 4\}$ are:

```
1 + 7 = 8
1 + 2 = 3
1 + 4 = 5
7 + 2 = 9
7 + 4 = 11
2 + 4 = 6
```

Rate This Challenge

More

Current Buffer (saved locally, editable)

C

```c
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char** split_string(char*);

// Complete the nonDivisibleSubset function below.
int nonDivisibleSubset(int k, int S_count, int* S) {


}

int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    char** nk = split_string(readline());

    char* n_endptr;
    char* n_str = nk[0];
    int n = strtol(n_str, &n_endptr, 10);

    if (n_endptr == n_str || *n_endptr != '\0') { exit(EXIT_FAILURE); }

    char* k_endptr;
    char* k_str = nk[1];
    int k = strtol(k_str, &k_endptr, 10);

    if (k_endptr == k_str || *k_endptr != '\0') { exit(EXIT_FAILURE); }

    char** S_temp = split_string(readline());

    int* S = malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        char* S_item_endptr;
        char* S_item_str = *(S_temp + i);
        int S_item = strtol(S_item_str, &S_item_endptr, 10);

        if (S_item_endptr == S_item_str || *S_item_endptr != '\0') { exit(EXIT_FAILURE); }

        *(S + i) = S_item;
    }

    int S_count = n;

    int result = nonDivisibleSubset(k, S_count, S);

    fprintf(fptr, "%d\n", result);

    fclose(fptr);

    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;
    char* data = malloc(alloc_length);

    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_length, stdin);

        if (!line) { break; }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length - 1] == '\n') { break; }

        size_t new_length = alloc_length << 1;
```