Jesse and Cookies

locked

by vatsalchanana

Problem Submissions Leaderboard Discussions Editorial

Jesse loves cookies. He wants the sweetness of all his cookies to be greater than value K. To do this, Jesse repeatedly mixes two cookies with the least sweetness. He creates a special combined cookie with:

sweetness $= (1\times$ *Least sweet cookie* $+ 2\times$ *2nd least sweet cookie*).

He repeats this procedure until all the cookies in his collection have a sweetness $\geq K$.
You are given Jesse's cookies. Print the number of operations required to give the cookies a sweetness $\geq K$. Print -1 if this isn't possible.

Input Format

The first line consists of integers N, the number of cookies and K, the minimum required sweetness, separated by a space.
The next line contains N integers describing the array A where A_i is the sweetness of the ith cookie in Jesse's collection.

Constraints

$1 \leq N \leq 10^6$
$0 \leq K \leq 10^9$
$0 \leq A_i \leq 10^6$

Output Format

Output the number of operations that are needed to increase the cookie's sweetness $\geq K$.
Output -1 if this isn't possible.

Sample Input

```
6 7
1 2 3 9 10 12
```

Sample Output

```
2
```

Explanation

Combine the first two cookies to create a cookie with *sweetness* $= 1 \times 1 + 2 \times 2 = 5$
After this operation, the cookies are $3, 5, 9, 10, 12$.
Then, combine the cookies with sweetness 3 and sweetness 5, to create a cookie with resulting *sweetness* $= 1 \times 3 + 2 \times 5 = 13$
Now, the cookies are $9, 10, 12, 13$.
All the cookies have a sweetness ≥ 7.

Thus, 2 operations are required to increase the sweetness.

Submissions: 85
Max Score: 12

Rate This Challenge:

More

Current Buffer (saved locally, editable)

C

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

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

/*
 * Complete the cookies function below.
 */
int cookies(int k, int A_count, int* A) {
    /*
     * Write your code here.
     */

}

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** A_temp = split_string(readline());

    int A[n];

    for (int A_itr = 0; A_itr < n; A_itr++) {
        char* A_item_endptr;
        char* A_item_str = A_temp[A_itr];
        int A_item = strtol(A_item_str, &A_item_endptr, 10);

        if (A_item_endptr == A_item_str || *A_item_endptr != '\0') { exit(EXIT_FAILURE); }

        A[A_itr] = A_item;
    }

    int result = cookies(k, A_count, A);

    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;
        data = realloc(data, new_length);

        if (!data) { break; }

        alloc_length = new_length;
    }

    if (data[data_length - 1] == '\n') {
        data[data_length - 1] = '\0';
    }

    data = realloc(data, data_length);

    return data;
}

char** split_string(char* str) {
    char** splits = NULL;
    char* token = strtok(str, " ");

    int spaces = 0;

    while (token) {
        splits = realloc(splits, sizeof(char*) * ++spaces);
        if (!splits) {
            return splits;
        }

        splits[spaces - 1] = token;

        token = strtok(NULL, " ");
    }

    return splits;
}

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code