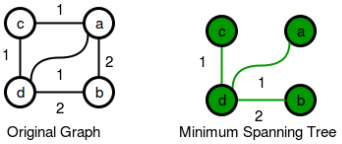
Minimum MST Graph

Allison loves graph theory and just started learning about Minimum Spanning Trees(MST). She has three integers, n, m, and s, and uses them to construct a graph with the following properties:

- The graph has n nodes and m undirected edges where *each edge has a positive integer length*.
- No edge may directly connect a node to itself, and each pair of nodes can only be directly connected by *at most* one edge.
- The graph is *connected*, meaning each node is reachable from any other node.
- The *value* of the minimum spanning tree is s. Value of the MST is the sum of all the lengths of all edges of which are part of the tree.
- The sum of the lengths of all edges is as small as possible.

For example, let's say $n = 4$, $m = 5$ and $s = 4$. We need to construct a graph with 4 nodes and 5 edges. The value of minimum spanning tree must be 4. The diagram below shows a way to construct such a graph while keeping the lengths of all edges is as small as possible:



Here the sum of lengths of all edges is 7.

Given n, m, and s for g graphs satisfying the conditions above, find and print the minimum sum of the lengths of all the edges in each graph on a new line.

Note: It is guaranteed that, for all given combinations of n, m, and s, we can construct a valid graph.

Input Format

The first line contains an integer, g, denoting the number of graphs.
Each of the g subsequent lines contains three space-separated integers describing the respective values of n (the number of nodes in the graph), m (the number of edges in the graph), and s (the value of the MST graph).

Constraints

For 20% of the maximum score:

- $1 \leq g \leq 100$
- $2 \leq n \leq 10$
- $1 \leq m \leq 50$
- $1 \leq s \leq 20$

For 50% of the maximum score:

- $1 \leq g \leq 100$
- $2 \leq n \leq 50$
- $1 \leq m \leq 2000$
- $1 \leq s \leq 200$

For 70% of the maximum score:

- $1 \leq g \leq 100$
- $2 \leq n \leq 10^5$
- $1 \leq m \leq 10^{10}$
- $1 \leq s \leq 10^6$

For 100% of the maximum score:

- $1 \leq g \leq 1000$
- $2 \leq n \leq 10^8$
- $1 \leq m \leq 10^{16}$
- $1 \leq s \leq 10^{10}$

Output Format

For each graph, print an integer on a new line denoting the minimum sum of the lengths of all edges in a graph satisfying the given conditions.

Sample Input

```
2
4 5 4
4 3 6
```

Sample Output

```
7
6
```

Explanation

- Graph $\mathbf{1}$:

The answer for this sample is already explained the problem statement.

- Graph $\mathbf{2}$:

We must construct a graph with $n = 4$ nodes, $m = 3$ edges, and an MST value of $s = 6$. Recall that a connected graph with n nodes and $n - 1$ edges is already a tree, so the MST will contain all $m = 3$ edges and the total length of all the edges of the graph will be equal to the value of the minimum spanning tree. So the answer is $\mathbf{6}$.

Submissions: 6
Max Score: 100
Difficulty: Expert

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*);



int main()
{
    char* g_endptr;
    char* g_str = readline();
    int g = strtol(g_str, &g_endptr, 10);

    if (g_endptr == g_str || *g_endptr != '\0') { exit(EXIT_FAILURE); }

    for (int g_itr = 0; g_itr < g; g_itr++) {
        char** nms = split_string(readline());

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

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

        char* m_endptr;
        char* m_str = nms[1];
        int m = strtol(m_str, &m_endptr, 10);

        if (m_endptr == m_str || *m_endptr != '\0') { exit(EXIT_FAILURE); }

        char* s_endptr;
        char* s_str = nms[2];
        int s = strtol(s_str, &s_endptr, 10);

        if (s_endptr == s_str || *s_endptr != '\0') { exit(EXIT_FAILURE); }

        // Write Your Code Here
    }

    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