18^{th} Marathon of Parallel Programming
SBAC-PAD & WSCAD – 2023

Calebe Bianchini[1] **and Roland Teodorowitsch**[2]

[1]Mackenzie Presbyterian University

[2]Pontifical Catholic University of Rio Grande do Sul

Rules for Remote Contest

For all problems, read carefully the input and output session. For all problems, a sequential implementation is given, and it is against the output of those implementations that the output of your programs will be compared to decide if your implementation is correct. You can modify the program in any way you see fit, except when the problem description states otherwise. You must upload a compressed file (*zip*) with your source code, the *Makefile* and an execution script. The script must have the name of the problem. You can submit as many solutions to a problem as you want. Only the last submission will be considered. The *Makefile* must have the rule all, which will be used to compile your source code. The execution script runs your solution the way you design it – it will be inspected not to corrupt the target machine.

The execution time of your program will be measured running it with time program and taking the real CPU time given. Each program will be executed at least three times with the same input and the mean time will be taken into account. The sequential program given will be measured the same way. You will earn points in each problem, corresponding to the division of the sequential time by the time of your program (speedup). The team with the most points at the end of the marathon will be declared the winner.

This problem set contains 4 problems; pages are numbered from 1 to 7.

General Information

Compilation

You should use **CC** or **CXX** inside your *Makefile*. Be careful when redefining them! There is a simple *Makefile* inside you problem package that you can modify. Example:

```
FLAGS=-O3
EXEC=sum
CXX=icpc

all: $(EXEC)

$(EXEC):
	$(CXX) $(FLAGS) $(EXEC).cpp -c -o $(EXEC).o
	$(CXX) $(FLAGS) $(EXEC).o -o $(EXEC)
```

Each judge machine has its group of compilers. See them below and choose well when writing your *Makefile*. The compiler that is tagged as *default* is predefined in **CC** and **CXX** variables.

machine	compiler	command
host	GCC 13.2.0 (*default*)	C = gcc C++ = g++
MPI	Open MPI 4.1.2a1 (*default*)	C = mpicc C++ = mpic++
	GCC 13.2.0	C = gcc C++ = g++
gpu	NVidia CUDA 12.0.1 (*default*)	C = nvcc C++ = nvcc
	GCC 8.3.1	C = gcc C++ = g++

Submitting

General information

You must have an execution script that has the same name of the problem. This script runs your solution the way you design it. There is a simple script inside you problem package that should be modified. Example:

```
#!/bin/bash
# This script runs a generic Problem A
# Using 32 threads and OpenMP
export OMP_NUM_THREADS=32
OMP_NUM_THREADS=32 ./sum
```

Submitting MPI

If you are planning to submit an MPI solution, you should compile using *mpiicc*/*mpiicpc*. The script must call *mpirun*/*mpiexec* with the correct number of processes (max: 4). It must use a file called `machines` that are generated by the *auto-judge* system - **do not** create it.

```
#!/bin/bash
# This script runs a generic Problem A
# Using MPI in the entire cluster (4 nodes)
# 'machines' file describes the nodes
mpirun -np 4 -machinefile machines ./sum
```

Comparing times & results

In your personal machine, measure the execution time of your solution using *time* program. Add input/output redirection when collecting time. Use *diff* program to compare the original and your solution results. Example:

```
$ time -p ./A < original_input.txt > my_output.txt
real 4.94
user 0.08
sys 1.56

$ diff my_output.txt original_output.txt
```

Do not measure time and **do not** add input/output redirection when submitting your solution - the *auto-judge system* is prepared to collect your time and compare the results.

Problem A
Number of Submatrices that Sum to k

Leonardo M. Takuno

Given an $M \times N$ matrix $A = (a_{ij})$ of integers and $k \in \mathbb{Z}$, this problem[1] involves counting the number of non-empty submatrices whose sums result in a given value k.

Note:

- The matrix may contain some negative numbers.
- Submatrix: A matrix obtained by removing some rows and/or columns from the beginning or/and from the end of a matrix.

For example 1, given the following 2×2 matrix and $k = 1$:

1	0
0	1

The output is 6.

Explanation: Sum the element $matrix[0][0]$, plus the element $matrix[1][1]$, plus the two submatrices line (1×2), plus the two submatrices column (2×1).

As a second Example 2, given the following 2×2 matrix and $k = 0$:

1	-1
-1	1

The output is 5.

Explanation: Sum the two submatrices line (1×2), plus the two submatrices column (2×1), plus 2×2 submatrix.

Your assignment is to develop a parallel program for solving this problem.

Input

The input contains only one test case. The first line contains three integers: the value k ($-10^8 \leq k \leq 10^8$), the number of rows (M) and the number of columns (N) of a matrix A separated by a blank space ($1 \leq M, N < 1000$). The next M lines contain N integers in each line separated by blank space representing a_{ij} element of A ($0 \leq i < M, 0 \leq j < N$, $-1000 \leq a_{ij} \leq 1000$).

The input must be read from the standard input.

Output

For the test case print one line of output, and integer denoting the number of submatrices that sum to k.

The output must be written to the standard output.

[1] based on LeetCode problem 1074.

Example

Sample input 1	Sample output 1
1 2 2 1 -1 -1 1	2

Problem B
Highest Sum Path in Generic Tree

Arthur F. Lorenzon

Given a generic tree, your objective is to identify and calculate the path from the root to a leaf that has the largest sum of node values. Each node in the tree has an associated value, and a path is defined as a sequence of nodes from the root to a leaf.

Your assignment is to develop a parallel program for solving this problem.

Input

The first input line has a unique integer N ($1 \leq N < 1500000$), the number of nodes in the tree. The next N lines describe the nodes. Each line begins with a decimal value V ($0.1 \leq V \leq 100.0$), the node value, followed by an integer K ($0 \leq K \leq 1000$), the number of children. Then, there are K integers representing the children indices (starting by 1).

The input must be read from the standard input.

Output

The first line must have the highest decimal value found. The second line must have the indices that compose the path from the root to leaf with the highest sum. The indices must be split by Spaces and start by 1.

The output must be written to the standard output.

Example

Sample input 1	Sample output 1

`5`
`10.5 2 2 3`
`5.2 2 4 5`
`3.3 0`
`4.1 0`
`2.4 0`
	`Max Sum:  19.80`
	`Path:  1 2 4`

Problem C
NPbonacci

Rafael Castro, Wellington Martins

The Fibonacci Sequence is a very well known recurrence, and we can describe any term F_i in the sequence, with $i \geq 1$, by the following:

$$F_i = \begin{cases} 0, & \text{if } (i = 1) \\ 1, & \text{if } (i = 2) \\ F_{i-1} + F_{i-2}, & \text{otherwise} \end{cases}$$

In the Fibonacci sequence the size of the base cases is two, the first two values, $F_1 = 0$ and $F_2 = 1$, are referred as base term, and any other term are constructed by using the sum of the immediate two terms before.

Another sequence called here as $NPbonacci$ consider a different set of base cases A, with size N. By consequence, any term that is not a base term, will consider the last N terms in the sequence to construct its own value. But, instead of only a simple sum to construct the next term value, here is applied a weighted sum, multiplying the terms by a value given by a list of integers P, where $|P| = N$. The $NPbonacci$ can be described by the following:

$$Fb_i = \begin{cases} A_i, & \text{if } (i \leq N) \\ \sum_{k=1}^{N} Fb_{i-k}.P_k, & \text{otherwise} \end{cases}$$

We can notice, for example, that the $NPbonacci$ sequence will be equal to the original Fibonacci, if we set $N = 2$, $A = [0, 1]$ and $P = [1, 1]$.

Given the values N, H, the base cases A and the list P, compute the term Fb_H modulo $10^9 + 7$.

Your assignment is to develop a parallel program for solving this problem.

Input

In the first line of the input you'll receive two integers, N and H. In the second line, will be given N integers, $A_1A_2A_3...A_N$, representing the base cases A. In the third line and last line, will be given N integers, $P_1P_2P_3...P_N$, representing the list P.

Constraints:

- $1 \leqslant N \leqslant 1020$;
- $N < H \leqslant 10^9$;
- $0 \leqslant A_i, P_i \leqslant 10^9$, $i = 1, \ldots, N$.

The input must be read from the standard input.

Output

Compute the value of Fb_H modulo $10^9 + 7$.

The output must be written to the standard output.

Examples

Sample input 1	Sample output 1
2 10 0 1 1 1	34

Sample input 2	Sample output 2
3 50 2 2 2 1 2 1	783060858

Problem D
Number of ways to traverse a grid

Salles V. G. Magalhaes

Given a matrix M (with dimensions NxN), the objective is to count the number of possible ways to traverse the matrix from cell (0,0) to cell ($N-1$,$N-1$), considering each move can only be done by either going to the right (increasing the x coordinate) or up (increasing the y coordinate).

In this version of the problem we have another restriction: some cells are blocked and, thus, no path can traverse them.

The proposed baseline algorithm employs dynamic programming to solve the problem. Your assignment is to develop a parallel program for solving this problem.

Input

The input has only one test case. The first line contains two integers N and M ($0 < N < 2 \times 10^5$). N represents the dimension of the matrix and M is the number of blocked cells ($M < 0.01 \times N^2$ when N is large (greater than 1000)). Then, there are M rows, each containing the y and x coordinates of a blocked cell.

The input must be read from the standard input.

Output

The output contains a single line. Print the number of paths from cell (0,0) to cell ($N-1$,$N-1$). You should print the answer modulus 1000000007 (since it may be big).

The output must be written to the standard output.

Example

Sample input 1	Sample output 1
3 1 1 0	3

Sample input 2	Sample output 2
3 0	6

Sample input 3	Sample output 3
4 0	20