19th Marathon of Parallel Programming SSCAD – 2024

Calebe P. Bianchini[1] **and Hélio C. Guardia**[2]

[1]Mackenzie Presbyterian University

[2]Universidade Federal de São Carlos

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 01 to 07.

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=g++

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 9.4.0 (*default*)	C = gcc C++ = g++
MPI	Open MPI 4.1.1 (*default*)	C = mpicc C++ = mpic++
	GCC 9.4.0	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 *mpicc/mpic++*. The script must call *mpirun/mpiexec* with the correct number of processes (max: 160).

```
#!/bin/bash
# This script runs a generic Problem A
# Using MPI in the entire cluster (4 nodes)
mpirun -np 4 ./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
Optimizing Media Storage on Blu Rays

Leonardo M. Takuno

Anyone who has ever recorded a variety of media onto CDs, DVDs or Blu Rays, knows that optimizing the use of space on each disk was never an easy task. Sometimes, it was necessary to adjust and reorganize movies, photos, and music to fit everything properly onto the disks.

Isabela needs to record a mix of movies, photos, and music onto her Blu Rays. She has a collection of digital files on her computer and would like to distribute them across several Blu Rays. She knows that each Blu Ray has a maximum storage capacity and is aware of the size of each type of file. However, she is having trouble deciding which files to put on which Blu Ray to maximize the use of each disk.

Write a parallel version of the this solution.

Input

The first line of input consists of two positive integers N and K, which represent the total number of files (movies, photos, and music) on Isabela's computer and the number of Blu Rays she has. The second line of input consists of N positive integers, which represent the size in gigabytes of each file. The last line of input consists of K positive integers, which represent the maximum storage capacity in gigabytes of each Blu Ray. No file is larger than 50 gigabytes, and no Blu Ray has a capacity greater than 50 gigabytes.

The input must be read from the standard input.

Output

Print a single line containing the maximum total number of gigabytes of files that can be recorded on the Blu Rays.

The output must be written to the standard output.

Example

Sample Input 1	Sample output 1

8 3
37 46 37 47 1 10 46 17
1 7 17
	18

Problem B
Traveling Salesman Problem Solver

Lucas S. Rosa, Alfredo Goldman

The Traveling Salesman Problem (TSP) is a classic optimization challenge in computer science and operations research. It asks the question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?".

This problem is of great importance in various real-world applications, including:

1. Logistics and transportation planning
2. Circuit board drilling in manufacturing
3. DNA sequencing in bioinformatics
4. Computer wiring
5. Delivery route optimization.

The TSP is known to be NP-hard, meaning that as the number of cities increases, the time required to find the optimal solution grows exponentially. This makes it an excellent candidate for exploring heuristic algorithms.

This solution implements a TSP solver using the Shotgun Hill Climbing algorithm. Here's a high-level overview of its functionality:

1. It reads a distance matrix from a CSV file, representing the distances between cities.
2. It uses a ”shotgun” approach, which involves multiple restarts of a hill climbing search.
3. For each restart:
    - It generates a random initial tour.
    - It then repeatedly attempts to improve this tour using 2-opt swaps.
    - If no improvement is found, it moves to the next restart.
4. After all restarts, it returns the best tour found across all attempts.

The algorithm balances between exploration (through multiple random restarts) and exploitation (through hill climbing), aiming to find a good approximate solution to the TSP in a reasonable amount of time.

Write a parallel version of the this solution.

Input

The first line of the input contains:

1. the umber of iterations for each hill climbing attempt
2. the number of restarts (shotgun attempts)
3. a seed for the random number generator.

The following lines contains the distance matrix describing the distances of the cities.

The input must be read from the standard input.

Output

The program outputs:

1. The best tour found, represented as a sequence of city indices
2. The total length of this tour

In this output, the sequence of numbers represents the order in which cities are visited in the best tour found; and the tour length represents the total distance traveled in this tour.

The output must be written to the standard output.

Example

Sample input 1	Sample output 1
`2000 20 17`	`Best tour found:  0 2 1`
`0.1,0.2,0.3`	`Tour length:  0.3`
`0.0,0.1,0.0`	
`0.2,0.0,0.1`	

Problem C
Zeros of Riemann Zeta Function

Luiz A. Steffenel

Zeros of the Riemann zeta function, denoted as $\zeta(s)$, are the values of the complex variable s for which $\zeta(s) = 0$. There are two types of zeros: trivial and non-trivial.

Trivial zeros occur at negative even integers $s = -2, -4, -6, ...$ These are considered "trivial" because their existence is relatively easy to prove using the functional equation of the zeta function.

Non-trivial zeros are the complex values of s for which $\zeta(s) = 0$ and have real part between 0 and 1. They are called "non-trivial" because their distribution is less understood, and their study is important for understanding prime numbers and related objects in number theory.

Riemann Hypothesis is one of the most famous and long-standing unsolved problems in mathematics, proposed by Bernhard Riemann in 1859. It conjectures that the Riemann zeta function has its zeros only at the negative even integers and complex numbers with real part $1/2$.

The code related to this challenge computes the number of zeros on the critical line of the Zeta function. The objective is not to compute the zeros: we count them to check that they are on the Riemann Line.

The exercise is to sample a region on the critical line to count how many times the function changes sign, so that there is at least 1 zero between 2 sampling points. Here we use a constant sampling but you can recode entirely the way to proceed.

Write a parallel version of the this solution.

Input

In the first line of the input you'll receive three integers: the first, L, means the lower bound; the cedont, U, mena sthe upper bound and the last, S, means how many samples will be tested.

The input must be read from the standard input.

Output

The output contains a single line. It has the total number of zeros.

The output must be written to the standard output.

Examples

Sample input 1	Sample output 1
`10 1000 100`	`I found 649 Zeros`

Problem D
The Tree Center

Tiago A. O. Alves

The eccentricity of a node u in a Graph $G(V, E)$ is the maximum distance between u and another node v, $v \in V$. In a tree, the center is the node with the minimum eccentricity. You can see an example on Figura .

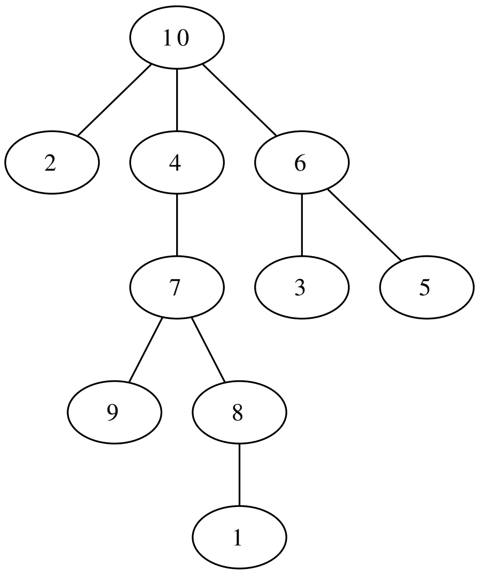

Figure 1. A tree in which the center is the node 4, with maximum distance 3.

This problem regards taking a tree as input and returning its center. Your assignment is to develop a parallel solution 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`