Data Center Capacity Exercise

A Data center hosts a distributed service that processes requests to a large set of data which is split in *N* subsets so that each subset can fit entirely on one machine. Each machine can host only one subset of the data, and process 100 requests per second. Each request is sent in parallel to *N* machines, each hosting a different subset, and the results are then aggregated with almost zero overhead; that means a group of *N* machines, each hosting a different subset, can jointly process 100 requests per second. To scale the processing up, additional machines responsible for the same subsets are introduced. Each additional group of *N* machines can potentially add another 100 requests per second to the overall capacity of the service. The entire data center therefore may be viewed as *M* machine groups, each containing *N* machines. The first machine in each group hosts the first subset of all data, the second serves the second subset, and so forth.

The task is to compute the overall capacity of the *M* by *N* service. Without additional restrictions, it is simply *M* x 100 requests per second, but there's an extra requirement for the service to maintain consistency across data updates. To do so, each subset is versioned with a 32-bit signed integer, and only subsets that have the same version can be used together for result aggregation. Versions may have arbitrary values; there can be no assumption about versions being positive, equal, or falling into a narrow range. Also, the same subset on different machines may have different versions which, while incompatible between themselves, may be compatible with versions for other subsets hosted by other machines.

You need to write a program that:

- Reads *M* and *N* from standard input.
- Reads versions of subsets hosted by all of *M* x *N* machines. Each line represents a group of *N* machines; *i*-th machine hosts *i*-th subset, and *i*-th number on the line is the version of the subset on that machine.
- Computes the overall capacity of the service and writes it to the standard output.
  You may write the program in either C, C++, or Java and use the standard libraries for collections and algorithms (e.g. STL). Please optimize for computational complexity in the worst case, and provide an upper boundary for it in terms of the basic operations - comparisons, copies, loop iterations, memory allocations (so if you use a standard container, you need to mention associated computational costs).

Example input:

```
4     5
103   107   103   107   107
107   103   107   107   103
103   107   107   103   103
103   103   107   107   107
```

Expected output:

```
200
```