Shear Sort

The Shear Sort algorithm (also called Snake Sort) is a parallel sorting algorithm originally designed to run on a two-dimensional mesh of processors. However, the same algorithm can be used to sort N values (with N a square number) on a multiprocessor computer using any number of threads. Shear Sort arranges the original array of $N = M^2$ elements into a square $M \times M$ matrix A. Then, it proceeds to execute $log_2(N)$ stages. In each stage, the rows of the matrix are sorted (alternating increasing and decreasing order) and then the columns are sorted (all in increasing order). The following pseudo-code summarizes Shear Sort:

```
function shearSort(A, M):
    repeat log2(M*M) times:
        sortRowsAlternateDirection(A,M)
        sortColumns(A,M)
```

The final matrix A contains the elements sorted if the matrix is traversed row by row alternating directions (starting left-to-right). The figure below presents an example with the first 3 stages of Shear Sort on a 16-element input.

Write a parallel OpenMP program that implements Shear Sort.

Make sure you follow these instructions:

a) You should implement a sorting algorithm (may be parallel) to sort each row and column.

b) You should add OpenMP directives into the code in order to find the *best* way to parallelize it.

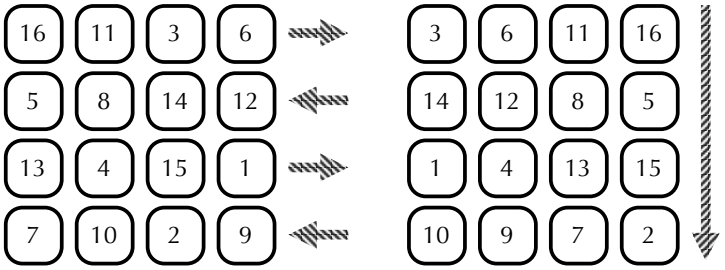
c) Write your code in the provided file `shear.cpp`.

d) Generate a sequential version of your code by compiling without flag `-qopenmp`.

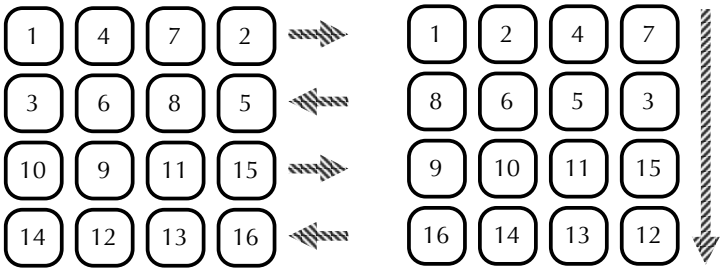
e) Your program must run with the following parameters:
`./shear <N>`
or
`./shear <N> <file>`
The first case will create a random array (stored as matrix) size N, while the second case will read the array from a file. In any case, N must be a square number.

f) Use the following command to analyze the performance of your code using different numbers of threads:
`OMP_NUM_THREADS=16 ./shear <N>`

Stage 1



Stage 2



Stage 3