

## 1. General Remarks

This is a warm up to pthreads and openMP programming. In the assignment you will permute an integer square array **A** in two different ways.

## 2. Permutations

Way 1 (by finding maximal elements in a column, think pivoting):

```
for i = 0 to n-1
    find max over k of abs(A[i,k]) in A[i:n,:]
    swap A[i,:] with A[k,:]
```

For example

$$\left( \begin{array}{c|ccc} 12 & 3 & -7 & 5 \\ 14 & -1 & 20 & 4 \\ 2 & 21 & 6 & 9 \\ -20 & 30 & 25 & 11 \end{array} \right) \xrightarrow{\text{column}(1)} \left( \begin{array}{c|ccc} -20 & 30 & 25 & 11 \\ 14 & -1 & 20 & 4 \\ 2 & 21 & 6 & 9 \\ 12 & 3 & -7 & 5 \end{array} \right) \xrightarrow{\text{column}(2)} \left( \begin{array}{cc|cc} -20 & 30 & 25 & 11 \\ 2 & 21 & 6 & 9 \\ 14 & -1 & 20 & 4 \\ 12 & 3 & -7 & 5 \end{array} \right) \text{ done}$$

Way 2 (by sorting subdiagonal elements in columns)

```
for i = 1 to n
    sort A[i:n,i] in decreasing order
    permute rows of A[i:n,i:n] so new column A[i:n,:] is sorted
```

For example

$$\left( \begin{array}{c|ccc} 12 & 3 & -7 & 5 \\ 14 & -1 & 20 & 4 \\ 2 & 21 & 6 & 9 \\ -20 & 30 & 25 & 11 \end{array} \right) \xrightarrow{\text{column}(1)} \left( \begin{array}{c|ccc} 14 & -1 & 20 & 4 \\ 12 & 3 & -7 & 5 \\ 2 & 21 & 6 & 9 \\ -20 & 30 & 25 & 11 \end{array} \right) \xrightarrow{\text{column}(2)} \left( \begin{array}{cc|cc} 14 & -1 & 20 & 4 \\ -20 & 30 & 25 & 11 \\ 2 & 21 & 6 & 9 \\ 12 & 3 & -7 & 5 \end{array} \right) \text{ done}$$

## 3. Codes

There are two codes to be written, one in pthreads and the other in OpenMP.

- A **pthread** code for implementing the Way 1 permutation scheme.
- An **openMP** code for implementing the Way 2 permutation scheme.
- Your codes must be written in standard C language and compiled by the **gcc** compiler (use the **-O3** optimization flag).

- Your code must be well documented so any person with limited knowledge of C could understand what the code is doing.
- The first line in the codes must show how the code should be compiled. The codes must be verified on the Linux machines in Phillips 314.
- Codes must be saved in separate files named  
`your_net_id_hw2_pthreads_max.c`,  
`your_net_id_hw2_openmp_sort.c`,  
 respectively.
- Your codes must be described in a file `your_net_id_hw2_codes.pdf`. For clarity and to save space you can refer to relevant lines in the codes.
- Your findings and discussion should be described in a file `your_net_id_hw2_writeup.pdf`
- All files need to be packed with the `tar` or `gzip` facilities. The packed file must have the name `your_net_id_hw2_2.suffix` where `suffix` is either `tar` or `zip`.
- Submit your work on Blackboard but also please e-mail your work to `awb8@cornell.edu`
- For pthreads directives consult

<https://computing.llnl.gov/tutorials/pthreads/>

or any other convenient source. Try to use only the simplest pthread constructs.

- For OpenMP directives consult

<https://computing.llnl.gov/tutorials/openMP/>

or any other convenient source. Try to use only the simplest OpenMP constructs.

- If you rely on resources outside lecture notes but publically available, you need to cite sources in your write-up.

## 4. Benchmarking.

The assignment is about finding fast methods for

- finding maximal in absolute value element in a 1D array
- sorting elements of a 1D array
- splitting the work among threads so data movements can be accomplished efficiently
- decide for what sizes of `A` creating multiple threads is beneficial (you may want to vary the number of threads within the code so the execution is as fast as possible)

Parameters:

- (a) To assess the quality of your codes you need to benchmark your code for speed-ups over sequential execution over a range of threads from 2 to twice the number of cores of the systems you will use. (Do not create a single thread but rather write a sequential code.) Do not increase the number of threads if it results in slowdown (you need to provide an evidence of slowdown).
- (b) It may be beneficial to decrease the number of threads when your submatrices become smaller and smaller.
- (c) Please benchmark your code over a range of dimension of the matrix **A**. Start with  $n = 256$ , and increase the size by a factor of 2 as long as your computer will allow you (but at least to  $n = 2^{10}$ ). (You may want to test your code on a small matrix, say  $n = 16$ , to check for correctness of your code.)
- (d) You are asked to graph and tabulate your results and discuss your findings in the file `your_net_id_hw2_writeup.pdf`

Please present your tables and graphs in a way so they are easily readable. For example, if certain combinations of (number of threads, matrix dimension) do not bring any new information, you may omit them from your graphs. But then explain why you are omitting them.

## 5. Remarks.

- (i) To generate **A** use standard gcc number generator `rand()` but set the seed first to the last digit of your net id.
- (ii) Your code will consist of parallel, sequential and mixed sections. You may want to time sections to see which ones take most time and then work on speeding them up.

## 6. Timing C codes.

You will need to use a fine resolution clock to time your codes. You can find examples how to do it at

<https://www.cs.rutgers.edu/~pxk/416/notes/c-tutorials/gettime.html>

Check also

[http://linux.die.net/man/3/clock\\_gettime](http://linux.die.net/man/3/clock_gettime)