# Project2 – Parallel Quicksort

Due date: 11:59 pm on April 30, Thursday

**90 points**

### Objective:

To practice writing a parallel program in MPI and run it in a parallel system.

### Problem to solve:

Implement the message-passing parallel formulation of the Quicksort algorithm described in the book (pp. 406~412) by writing a function. The function’s prototype can be defined as follows:

PQsort(int nelements, int \*elements, int pivot, MPI\_Comm comm)

where *nelements* is the number of elements that are stored locally by the calling processor, *elements* is a pointer to the array that stores the local elements, pivot is the pivot and *comm* is the MPI communicator to be used for all communication. Upon completion, your function should return exactly *nelements* entries of the final increasing order sorted array to the calling processor. The ordering of the processors is determined by the process-rank returned by the MPI\_Comm\_rank(comm) routine, where *comm* is the communicator argument of the routine.

**Other Requirements:**

* This project is to be completed with a group of 2 students. Your group formation is provided at the end of this document.
* To make the implementation easier, assume each processor has about same number of elements. For example, in a three processor case, the sub-arrays in processor P0, P1, and P2 can be [2, 1, 9], [0, 5, 3], [4, 8, 12, 7]. Upon completion, your routine should return to the different processors the following sub-arrays: P0:[0, 1, 2], P1:[3, 4, 5], P2:[7, 8, 9, 12].
* You should use a randomized pivot selection scheme that is performed in two steps: (i) Select a processor randomly; (ii) Have this processor randomly select one of its elements as the pivot. After pivot selection, this pivot needs to be broadcasted to all the processors.
* To test your function, you need to write a test program. You can assume that the entire array is initially stored at processor P0 (and the rest of the processors provide zero elements). Your test program should perform an initial array redistribution in which each processor gets approximately n/p elements. The final ordering should be displayed properly.

### What to submit:

Turn in the following documents to your **studentWorkFolder**:

[\\STCLOUDSTATE\HuskyNet\CourseFiles\Spring2020\jhu\CSCI475s02\StudentWorkFolder\yourStartId](file:///\\STCLOUDSTATE\HuskyNet\CourseFiles\Spring2020\jhu\CSCI475s02\StudentWorkFolder\yourStartId)

* Source codes (40 points)
* Copies of the input files. (5 points)
* Script files with running results using 1, 4, 16 and 64 processors (20 points)
* A document that contains the following items:
  + Detailed description of the algorithm for your PQsort function (10 points)
  + Detailed description about how your test program is designed. (10 points)
  + User document about how to run the program. (5 points)

**Group formation**

|  |  |  |
| --- | --- | --- |
| **Last Name** | **First Name** | **Group #** |
| Hoeschen | Andrew | #1 |
| Lewis | Steven | #1 |
| Clapp | Brent | #2 |
| Davis | Joshua | #2 |
| Anderson | Donivan | #3 |
| Erickson | Micah | #3 |
| Christenson | Mark | #4 |
| Al-Sharabati | Abdulrahman | #4 |
| Abshir | Misky | #5 |
| Manraj | Aneesa | #5 |