# CS121 Problem Set 2

Due: 23:59, November 11, 2021

1. Submit your solutions to Gradescope (www.gradescope.com).
2. In "Account Settings" in Gradescope, set FULL NAME to your Chinese name and enter your STUDENT ID.
3. If you submit handwritten solutions, write neatly and submit a clear scan.
4. When submitting your homework, be sure to match each of your solutions to the corresponding problem number.

1. Consider the same situation as in problem 8 of problem set 1. Give another algorithm for all-to-all broadcast which takes time ($t_s + t_w\, m$) ($p$-1).

   *Hint*: Try to embed a $p$ process ring in the tree.

2. Consider the following sequential *rank sort* algorithm for $n$ values assuming no duplicate values:

```
for (i = 0; i < n; i++) {
    x = 0;
    for (j = 0; j < n; j++)
        if (a[i] > a[j]) x++;
    b[x] = a[i];
}
```

   (a) Rewrite this as a parallel algorithm using OpenMP assuming that $p < n$ threads are used. Clearly indicate the use of private and shared variables and the schedule type.

   (b) Modify the OpenMP code in part (a) to handle duplicates in the list of values, i.e. to sort into non-decreasing order. For example, the list of values [3, 5, 7, 5, 7, 9, 2, 3, 6, 7, 8, 1] should give the sorted list [1, 2, 3, 3, 5, 5, 6, 7, 7, 7, 8, 9].

3. The following sequential code calculates a triangular matrix using a function **`calc(i,j)`**, which has no data dependences and requires a constant (but large) amount of computation.

```
for (i = 0; i < n; i++)
   for (j = 0; j <= i; j++)
      a[i,j] = calc(i,j);
```

   By inserting OpenMP directives into the sequential code, show how the following schemes for assigning work to threads may be implemented on a shared memory parallel architecture, commenting on the efficiency of each scheme:

(a) A static block assignment of contiguous rows to threads.

(b) A static cyclic assignment of single rows to threads.

(c) A dynamic assignment of single rows to threads.

4. The *Back Substitution* algorithm solves a set of linear equations in upper (or lower) triangular form, as shown in Figure Q4a. A sequential algorithm to solve such a set of linear equations is given in Figure Q4b. Design a parallel algorithm for a shared memory architecture and express it using OpenMP assuming that p < n threads are used. What schedule type would you use?

$$\begin{array}{lr}
a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 \quad \ldots \quad + a_{n-1,n-1}x_{n-1} & = b_{n-1} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad . & \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad . & \\
a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 & = b_2 \\
a_{1,0}x_0 + a_{1,1}x_1 & = b_1 \\
a_{0,0}x_0 & = b_0
\end{array}$$

**Figure Q4a**

```
/* Back Substitution */
for (i = 0; i < n; i++) {
    x[i] = b[i]/a[i][i];
    for (j = i+1; j < n; j++) {
        b[j] = b[j] - a[j][i]*x[i];
        a[j,i] = 0;
    }
}
```

**Figure Q4b**

5. We discussed barrier synchronization as a way to ensure all threads reach a point in the code before any of them progress beyond the point. The figure below shows a barrier for N threads implemented using locks. Explain clearly how this code works. What should the initial values of `count` and the `arrival` and `departure` locks be? Why is it necessary to use two lock variables?

```
void barrier()
{
    set_lock(arrival);
    count++;
    if (count < N)
        unset_lock(arrival);
    else
        unset_lock(departure);
    set_lock(departure);
    count--;
    if (count > 0)
        unset_lock(departure);
    else
        unset_lock(arrival);
}
```

6. GPU computing has been used to speed up many real-world applications. However, not all applications are suitable for GPU acceleration. Consider the following operations / applications and decide whether each is suitable for GPU acceleration or not. Briefly justify your answer. Assume all the input data are initially in the main memory.

   (a) Matrix multiplications on two matrices *A* and *B*, each of size 32000 by 32000.

   (b) Matrix multiplications on two matrices *A* and *B*, each of size 32 by 32.

   (c) Binary search on a sorted array with 1 billion elements.

   (d) Binary search on a sorted array with 1 thousand elements.

7. Compared to CPU threads, GPU threads are designed to be "light-weight". Describe some potential disadvantages associated with increasing the amount of work done in each CUDA thread (e.g. using loop unrolling) and thereby decreasing the total number of threads used.