# Project #2 - Parallel Radix Sort using OpenMP

SWE3021 Multicore Computing - Fall 2020

Due date: November 6 (Fri) 17:00pm

For your second project, you will implement a parallel radix sort algorithm, and then evaluate its performance. You may want to compare the performance of your implementation against the standard sequential `qsort()` of `stdlib.h`. Again, this project will consist of not just coding, but also testing your code using large data sets.

## 1 Parallel MSD Radix Sort

Radix sort was developed for sorting large integers, but it treats an integer as a string of digits, so it is really a string sorting algorithm. There are two types of radix sorting: MSD (most significant digit) and LSD (least significant digit) radix sort. MSD radix sort starts sorting from the beginning of strings, thus it is suitable for lexicographic sorting of variable-length strings. In this project, you are going to parallelize the following MSD radix sort for variable-length strings.

```
void msd(string[] a) {
    msd(a, 0, a.length, 0);
}

void msd(string[] a, int lo, int hi, int d) {
   if (hi <= lo + 1) return;

   for (int i = 0; i < N; i++)
       count[a[i].at(d) + 1]++;
   for (int k = 1; k < 256; k++)
       count[k] += count[k-1];
   for (int i = 0; i < N; i++)
       temp[count[a[i].at(d)]++] = a[i];
   for (int i = 0; i < N; i++)
       a[i] = temp[i];
   for (int i = 0; i < 255; i++)
       msd(a, l + count[i], l + count[i+1], d+1);
}
```

Your program should accept five arguments: the name of an input file that has a string per line, the number of strings your program takes as input, the start index and the number of sorted strings your program should print out, and the number of concurrent threads. For example, suppose you have an input file as the following.

```
$ cat input.txt
Your
program
should
accept
five
arguments:
the
name
of
```

```
an
input
file
that
has
a
string
per
line,
the
number
of
strings
...
```

Suppose you want to sort the first 22 strings. Then, you should run your program as the following:

```
$ ./a.out input.txt 22 0 22 16
a
accept
an
arguments:
file
five
has
input
line,
name
number
of
of
per
program
should
string
strings
that
the
the
Your
Elapsed time: 0.00000079 sec
```

If you want to sort only the first 10 strings and print out a subset of the sorted array, i.e., five strings from the second smallest string, then, you should run your program as the following:

```
$ ./a.out input.txt 10 1 5 16
program
should
accept
five
arguments:
Elapsed time: 0.00000032 sec
```

Note: you should not include the time to read the given input file from disks. I.e., your code must have the following structure.

```
int main(int argc, char* argv[]){
   // declarations
   struct timespec start, stop;
```

```
    ...

    // read an input file
    ifstream inputfile;
    inputfile.open(argv[1]);
    for(int i=0;i<atoi(argv[2]);i++){
        inputfile >> array[i++];
    }
    inputfile.close();

    // now we measure the time
    clock_gettime( CLOCK_REALTIME, &start);

    // OpenMP initialization
    omp_set_num_threads(atoi(argv[5]);
    ...

    // parallel radix sort
    ...

    // we are done with measuring the time
    clock_gettime( CLOCK_REALTIME, &stop);

    // print out the result
    for(...) cout << array[...] << endl;

    cout << "Elapsed time: " << (stop.tv_sec - start.tv_sec) +
           ((double) (stop.tv_nsec - start.tv_nsec))/BILLION << " sec" << endl;

}
```

## 2 How to Submit

To submit your project, you must name your code as project2.cpp and run `multicore_submit` command in your project directory in swji.skku.edu server.

```
$ multicore_submit project2 project2.cpp
```

For any questions, please post them in Piazza so that we can share your questions and answers with other students. Please feel free to raise any issues and post any questions. Also, if you can answer other students' questions, please don't hesitate to post your answer. You would get some credits for posting questions and answers in Piazza.