CS121 Problem Set 5

Due: 23:59, May 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.

1a) Consider the sequential code shown in the figure below. List all the dependence relationships in the code, indicating the type of dependence in each case. Draw the loop-carried dependence graph (LDG).

```
for (i=1; i<=n; i++)
  for (j=1; j<=n; j++) {
    S1: a[i][j] = a[i-1][j] + b[i][j];
    S2: b[i][j] = c[i][j]
  }
```

1b) Using the LDG derived in Q1a, describe how to obtain a parallel algorithm, explaining any modifications required to improve the efficiency. Express your parallel algorithm using OpenMP.

2) Consider the loop shown in the figure below. Indicate the loop-carried dependence relationships. By restructuring the code, is it possible to eliminate this dependence and parallelize the loop? Express your answer using OpenMP.

```
a[0] = 0;
for (i=1; i<=n; i++)
  S1: a[i] = a[i-1] + i;
```

3) One way to get a numerical approximation to π is to sum the following sequence:

$$\pi = 4(1 - 1/3 + 1/5 - 1/7 + \ldots)$$

A sequential algorithm to calculate π using this formula is given in the figure below. By analyzing the loop-carried dependences in this code, show how the algorithm may be parallelized, expressing your answer using OpenMP. Indicate clearly any modifications required to the code in order to allow parallelization and improve the efficiency.

```
double factor = 1.0;
double sum = 0.0;
for (i = 0; i < n; i++) {
    S1: sum += factor/(2*i+1);
    S2: factor = -factor;
}
pi = 4.0*sum;
```

4) Consider the even-odd transposition sort algorithm, whose code is shown below. The algorithm works similarly to bubble sort, but is more parallelizable. Given *n* numbers to sort (say into nondecreasing order), it works in *n* phases. In the even phases, we take each even indexed number, and swap it with its right neighbor if necessary so that they are in nondecreasing order. In the odd phases, we do the same thing for the odd indexed numbers. An example execution is shown below.

```
for (phase = 0; phase < n; phase++)
   if (phase % 2 == 0)
      for (i = 1; i < n; i += 2)
         if (a[i-1] > a[i]) Swap(&a[i-1],&a[i]);
   else
      for (i = 1; i < n-1; i += 2)
         if (a[i] > a[i+1]) Swap(&a[i], &a[i+1]);
```

	Subscript in Array					
Phase	0		1		2		3
0	9	↔	7		8	↔	6
	7		9		6		8
1	7		9	↔	6		8
	7		6		9		8
2	7	↔	6		9	↔	8
	6		7		8		9
3	6		7	↔	8		9
	6		7		8		9

Source: An Introduction to Parallel Programming, Peter Pacheco

4a) Prove that after *n* phases, all the numbers are sorted.

4b) Show how to parallelize this program in OpenMP.