CS433: Assignment#1 (Due by 10th March 2022 midnight)

1. In this problem, you will write a shared memory parallel program to solve the traveling salesman problem (TSP) for undirected graphs with positive integer edge weights. The problem is to find the minimum weight tour (i.e., the sum of weights of the edges in the tour is minimum) that visits every vertex exactly once and returns to the starting vertex. The program will accept an input file name, an output file name, and the number of threads from the command line exactly in that order. The input file describes the edge weights in a certain format. I have included a sample input file so that you can generate similar files for testing. The first line in the file contains the number of vertices in the graph. The next line contains the weights of the edges connecting vertex#1 to vertex#i in increasing order of i. In general, a particular line shows the weights of the edges connecting vertex#k to vertex#i in increasing order of i such that $i > k$. The sample file shows the edge weights for a complete graph of four vertices. The second line shows the weights of edges (1, 2), (1, 3), (1, 4). The next line shows the weights of edges (2, 3), (2, 4). The last line shows the weight of the edge (3, 4). It is very important to adhere to the input file format because we will test your submission on graphs generated by us. So, even if you have written a correct and efficient parallel program, if we cannot run your program on our graphs, we won't be able to give you any score. The program should output the minimum weight tour through all vertices along with the total weight of the tour in the output file in the following format: the first line of the file will show the tour as a list of vertices and the second line will show the total weight of the tour. Note that depending on the graph structure, it is possible to have multiple different minimum weight tours. It is sufficient to output any one of them. Please feel free to choose OpenMP or POSIX thread for your implementation. **(10 points)**

2. In this problem, you will write a shared memory parallel program to solve a lower triangular system of equations $Lx = y$ where L is an $n \times n$ lower triangular matrix of reals and x, y are real-valued $n \times 1$ vectors. The problem is to find x given L and y. Row i of a lower triangular matrix has zeroes in columns $i + 1$ to $n - 1$ for $0 \leq i \leq n - 1$. Additionally, we will assume that no diagonal element of L is zero. The program will take an input file name, an output file name, and the number of threads from the command line exactly in that order. The first line of the input file specifies the value of n. The second line onward it lists the non-zero elements of L row-wise. The last line of the file lists the elements of y. I have included a sample file specifying the following L and y.

$$L = \begin{bmatrix} 1.0 & 0 & 0 \\ 1.0 & 2.2 & 0 \\ 3.0 & 4.1 & -1.3 \end{bmatrix}, \; y = \begin{bmatrix} 2.3 \\ 4.5 \\ -7.2 \end{bmatrix}.$$

Note that reading large matrices from the file may take a significant amount of time. So, for testing your program on large matrices quickly, you can initialize L and y with arbitrary values in your program. Please put this code in a clearly marked function named InitializeInput. Comment out the call to this function in your submission so that we can test your program with our input test files. The program should output the elements of x in the first line of the output file. The output file should not have any other lines. Please feel free to choose OpenMP or POSIX thread for your implementation. Hint: Do not invert L and then multiply by y. Use simpler ways of finding x. **(10 points)**

What to submit and how. For each question, submit the parallel program. Prepare a report describing your parallel algorithms, the optimizations you applied (if any) for improving performance, and the performance results. Please clearly specify in the report whether your program is written using OpenMP directives or POSIX thread libraries for each program.

For the first question, report results for at least five graph sizes. All five graphs must be complete graphs. For each graph, prepare a table showing how the measured time varies with the thread count (the range of thread count is left to you). Try to pick largish graphs that your program can run within reasonable time. Remember to specify the number of vertices in each graph clearly. Discuss the observed trends with explanation.

For the second question, report results for at least five values of n. For each n, prepare a table showing how the measured time varies with the thread count. Remember to specify the values of n clearly. Try to pick large n that your program can run within reasonable time. Discuss the observed trends with explanation.

Note that your marks will depend on how good your program is in terms of performance, how well the algorithm is described in the report, and the discussion on the results. In your measured time, do not include any time that is needed to allocate memory or initialize arrays. Please refer to the examples discussed in the class. You will not get any marks if your program does not compile or crashes during execution. Only the programs that compile and run to completion without any error will be considered for grading.

Please send your submission (two programs and a PDF report) via email to cs433submit2022@gmail.com with subject “Assignment#1 submission Group#X”. Replace X by your group number in the subject line.