COL 380	March 28, 2017
Lab 1	
Instructor: Subodh Sharma	**Due: April 5, 23:55 hrs**

1 Problem Statement

In this lab, you will create a parallel (OpenMP) solution to the traveling salesman problem (TSP) using genetic algorithms (GA). TSP is known to be a NP-Hard problem that results in a computational explosion with a time complexity of O(n!). The TSP problem has numerous applications - particularly in scheduling-related scenarios such as computer wiring, vehicle routing, etc.; therefore a scalable solution to TSP is eagerly sought.

GA offer a scalable strategy to find *near-optimal* solutions to a whole bunch of optimization-related problems, including TSP. Broadly, a GA strategy to solve TSP appears to be the following:

- Initializing population: Begin initially with a *population*, *i.e.* a randomly chosen set of candidate solutions to the TSP problem. For instance, if we have to visit 36 cities with each city named with one of the 36 characters from the set $\Sigma = \{A, \cdots, Z\} \cup \{0, \cdots, 9\}$, then 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ is one such candidate solution (or a possible trip). We refer to a candidate solutions by the term *chromosome*. Each city in a chromosome is understood to be a *gene*.

- Selection: The next step is to evaluate the *fitness* of chromosomes in the population. This could be achieved by defining a fitness function. For this lab, consider the fitness function to be $f(i) = \frac{1}{D_i}$, where for each chromosome (or trip) i, D_i is the total Euclidean distance. Thus, maximum fitness of a chromosome i is when the i has the shortest D_i. After evaluating the fitness of the population, select a fixed number of fittest chromosomes as *parents*.

- Recombination: Perform a *crossover* on the selected set of parents. Choosing the right crossover algorithm is extremely important. There are various crossover algorithms that have been studied in depth. We will restrict the scope in this lab to the following two crossover algorithms:

  1. Partially Mapped Crossover (PMX)[1]: Consider two parent chromosomes, P_1 and P_2; randomly select a substring from parent P_1. The genes in the substring between the positions (including the genes at the positions) are exchanged with the genes within the same positions in the parent P_2. Determine the mapping relationship and legalize the offspring with the discovered *relation map*. For instance, consider two parents $P_1 = 1\,2\,3\,4\,5\,6\,7\,8$ and $P_2 = 3\,7\,5\,1\,6\,8\,2\,4$. Suppose that PMX chooses a substring $4\,5\,6$ from points 4 to 6 in P_1. Then the genes $1\,6\,8$ are selected from P_2 to match the substring $4\,5\,6$ with the relation map as $\{(4,1),(5,6),(6,8),(2,2),(3,3),(7,7),(5,8)\}$. Assume the relation map to be symmetric. The new produced chromosomes are $P_1^* = 4\,2\,3\,1\,6\,8\,7\,5$ and $P_2^* = 3\,7\,8\,4\,5\,6\,2\,1$. Care has to be taken that the genes (or cities) are not repeated in the chromosome (*i.e.*, the paths are hamiltonian).
  2. Another crossover technique is the Greedy Crossover [GX][2]: Again consider two parent chromosomes; pick the first gene of P_1 and compares the genes leaving that gene in P_1 and P_2, and choose the closer one to extend the child chromosome. If one gene has already appeared in the child chromosome, then choose the other gene. If both genes have already appeared, then randomly select a non-selected gene. Similarly, generate the second child

chromosome by taking the complement. Complement can be understood with the following example: imagine a chromosome with 8 genes, ABCDEFGH, then its complement will be HGFEDCBA. Thus, the complement relation map is $\{(A, H), (B, G), \cdots\}$

- Mutation: After introducing crossed-over children to the population, perform random mutation of genes in a randomly chosen set of crossed-over chromosomes. By mutation, we mean that two distinct genes are chosen from each chromosome with a given probability and their positions in the chromosome is swapped.

In short, the algorithm appears as following:

```
generate-initial-population()
evaluate-fitness-of-population()
while (termination-criteria-not-met) {
  breed new population: selection();
  For-every-two-fit-chromosomes: perform-recombination();
                                 perform-mutation();
                                 evaluate-fitness-of-new-population();
}
```

1.1 Statement of work

Implement the function parallel_tsp() and associated functions to complete the parallelization of TSP.

- Choose an initial population from 500 to 10000 for cities ranging from 10 to 35;
- Implement a crossover strategy that combines PMX and GX.
- Appropriately choose the number of iterations of the while loop (termination check) [anywhere from 1000 to 10^5].
- Assume the probability of cross-over to be .8 and probability of mutation of each gene is 0.1.
- Output should be a path along with the distance of that path.
- Demonstrate the speedup on threads in the range 4 to 32;
- Prepare a lab report (as .txt file) by clearly explaining your design decisions, parallelization strategy, load-balancing strategy. Make sure that explanations are bulleted and are not long paragraphs. Keep the lab report within 800 words. Compile your programs with -O3 option and make sure that while debugging, option -O3 is replaced by -g. In order to debug use DDD, GDB, or IDE-based debuggers.
- Turn in the source code file and the report. The break-down on grading is as follows:
  1. Correct execution of the program: 75 Marks
  2. Clarity in report and discussion of your solutions: 20 Marks
  3. Code modularity, code comments, coding style: 5 marks

1.2 Extra Credit

There are various other cross over techniques such as cycle crossover (CX), edge recombination crossover (ERX), etc. Try to create a hybrid recombination strategy that results in a solution closer to the *exact* solution while taking less time. (**15 marks**)

1.3 References

[1] Goldberg, D.E., and R. Lingle. Alleles, Loci, and the Traveling Salesman Problem. Proceedings of the First International Conference on Genetic Algorithms and Their Application , edited by Grefenstette J., Lawrence Erlbaum Associates, Hillsdale, NJ,1985, pp. 154-159.

[2] J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht, Genetic Algorithms for the Traveling Salesman Problem, Proc. of 1st Int. Conf. on Genetic Algorithms and Their Applications, pp.160 - 168, 1985.