

## Multicore Processors: Architecture & Programming Lab Lab 1

In this lab you will implement a method for solving a modified version of the traveling salesman problem in Pthreads.

### What is the problem definition?

You have a group of  $n$  cities (from 0 to  $n-1$ ). The traveling salesman must start from city number 0, **visit each city once**, and does not have to come back to the initial city (This is why we said it is a modified version of the original traveling salesman problem). You will be given the distance between each city (cities are fully connected to each other). You must **pick the path with shortest distance**. Note that there may be several paths that satisfy the above conditions, you just need to find one of them.

### What is the input?

The input to your program is a text file that contains an  $N \times N$  matrix, one row per line, representing the distance between any two cities. Your program will be called `ptsm` (for parallel tsm). The command line then will be:

```
> ptsm x t filename.txt
```

Where :

**x** is the number of cities

**t** is the number of threads

**filename.txt** is the file that contains the distance matrix

### What is output?

Your program must output, to screen, the best path you found and the total distance. Here is an example output:

```
Best path: 0 1 3 4 5 2
```

```
Distance: 36
```

This means the best path your program found is  $0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2$  with total distance of 36.

### How will you solve this?

This problem is a combinatorial optimization problem ( $n!$  possible solutions for  $n$  cities). One of the straightforward ways to solve it is through branch and bound. One way you might think about evaluating every possible route is to construct a tree that describes all of the possible routes from the first city. However, the tree may get too large. One way to

reduce the size is to follow one path till the end. This path has a cost of  $w$ , for example. When you explore a second path, as soon as the cost becomes larger than  $w$ , then don't pursue it and move to another branch. If another full path turns out to be of smaller cost, then it becomes your best path and record its cost.

The above scheme is just a suggestion; so feel free to optimize/modify it in any way you want. Also you have to decide about the total number of threads needed given the number of cities.

We will not test your submission with more than 10 cities.

### **How will we grade this?**

The total grade is out of 20

- 5 points if your program outputs a legal solution (i.e. each city visited once) but not necessarily one of the shortest paths.
- 5 points for the report
- 10 points if your solution is within 10% of the optimal solution.

### **To help you:**

We are providing you with two executables:

- `./tsm` : This program generates test files (usage: `./tsm numcities`). The output of this program is `citiesx.txt` that contains the matrix of the  $x$  city distances. It will also print a (sub) optimal solution on the screen.
- `./tsmoptimal` : This programs prints the optimal solution on the screen (usage: `./tsmoptimal x citiesx.txt`) for the  $x$  cities problem.

### **The report**

You need to write a report that includes the following:

- A graph that shows speedup over single-thread version for a problem of size 5 cities and another problem of size 10 cities. For each graph, the x-axis the number of threads and y-axis is the total time generated by *time* command. You may need to execute time command several times (~5) and take the average.
- Got to `/proc/cpuinfo` on CIMS machine to get the specs for the processors you are running your code on. Based on this, explain whether the above graphs make sense or there are anomalies, and justify.

### **What to submit?**

Add the source code as well as the pdf file that contains your graphs and conclusions to a zip file named: `lastname.firstname.zip`

Where `lastname` is your last name, and `firstname` is your first name.

**How to submit?**

email the above zip file to our grader (not to me, not to the mailing list!) with subject line: **lab1 submission**

**Have Fun!**