Programming Assignment 1

Student ID:

Name: 王順興

**Serial version:**

num\_city: number of cities to connect

2opt\_swap(start, end);

for (i = 1; i < num\_city - 1; ++i) {

for (j = i + 1; j < num\_city; ++j) {

2opt\_swap(i, j);

}

}

**Can be changed to:**

start from (1) to (num\_city - 1)

2opt\_swap with \*depth\* (2) to (num\_city - i)

for (i = 1; i < num\_city - 1; ++i) {

for (depth = 1; depth < num\_city - i; ++depth) {

2opt\_swap(i, i + depth);

}

}

*depth* is the length between the 2opt\_swap indices

**Idea: Split *depth* across threads**

num\_thread: number of thread available

num\_city: number of cities to connect

maximum\_depth = num\_city - 1

depth\_for\_each\_thread = maximum\_depth / num\_thread

2opt\_swap(start, end):

\_read\_lock

create new\_route

\_unlock

assert(distance(new\_route) < distance(current\_route))

\_write\_lock

current\_route = new\_route

\_unlcok

end

* A race condition may occur when a thread pass through assertion, but not yet change the current\_route; a better current\_route may be overwritten, so the assertion have to be done again after applying \_write\_lock
* The aforementioned race condition also prevents partial update of the current\_route, so a new array(new\_route) is created every opertaion.
* distance(current\_route) can be cached.

1. Performance (run-time) analysis with 1, 2, 4, 8, 16 core(s)
2. Record your distance every 30 seconds with **test11**

|  |  |
| --- | --- |
| Time | Distance |
| 0m 30s |  |
| 1m 00s |  |
| 1m 30s |  |
| 2m 00s |  |
| 2m 30s |  |
| 3m 00s |  |
| 3m 30s |  |
| 4m 00s |  |
| 4m 30s |  |
| 5m 00s |  |
| 5m 30s |  |
| 6m 00s |  |
| 6m 30s |  |
| 7m 00s |  |
| 7m 30s |  |
| 8m 00s |  |
| 8m 30s |  |
| 9m 00s |  |
| 9m 30s |  |
| 10m 00s |  |

1. Discussion
2. Feedback