Programming Assignment 1 Report

Aayush Patel

CS20BTECH11001

Approach:

- .There are two ways to execute this program. The data can be given in the input.txt file in the format specified by the question and the output will be printed in the terminal. Set the useFileReading variable to **true** at **line 13** for this.
- Another way is to provide length of the array and the number of threads from the command line. Set the useFileReading variable to false at **line 13**. Running the code will randomly generate the array of specified length and output will be printed(this will be used to run the simulations for the plots that will be shown below).
- From line 70-102, the code performs one of the two operations specified above.
- Next at line 109 the clock function is used. This will be used later to calculate time of execution.
- Line 112 creates N-length array for storing the tid's of the N threads which will be created later.
- Lines 113 creates pointers to N pointers to params struct. These parameters pointer struct are the ones which will be passed to *pthread_create()* function for each thread.
- Next a loop is defined which will run N times. On each iteration, it will create a new thread. Each thread will run perform the findNearest function on the array from start index to end index(both included). The variables start and end are updated such that approximately uniform sized subarrays are formed on which findNearest function is performed.
- Next a loop is defined which will wait for all N threads to join.
- Now we have N minDistances from the N subarrays. To get the minDistance on whole array, we use the standard brute-force finding minimum operation.
- At line 164 the time taken for this whole process is calculated in seconds.
- In the end, we print the time taken as the output.

Optional:

- There is a *simulator.py* file provided which will average the time obtain for specified number of threads and number of elements over n_sim number of simulations. Make sure before running this, the useFileReading on **line 13** of **c file** is set to **false**.

Sample Input in input.txt file:

2
(1,1)
4
(2,2)(4,5)(0,3)(1,2)

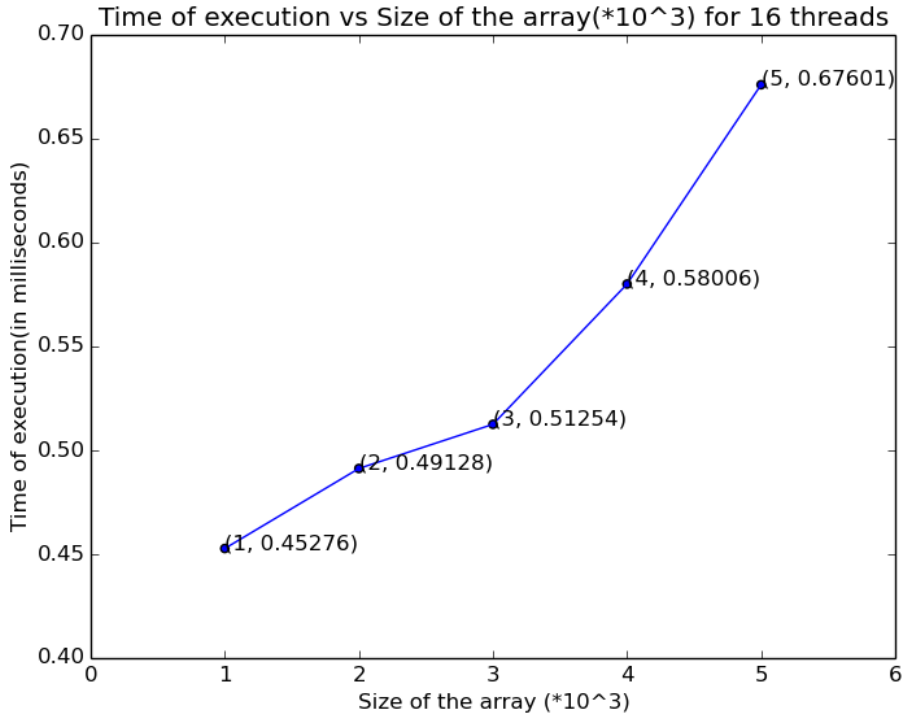
Sample Output in output.txt file:

Time taken for TSP of 4 elements with 2 threads: 0.168000 milliseconds
(1,2)

Comparing sequential execution time of findNearest(N=1) with execution time of it with multiple threads

As we can see from the plot, time of execution increases from N=1 to N=16. This may seem unusual but there is might be an explanation for that. Since the number of elements(=5000) is reasonably small for computation, a sequential program executes it in a lesser time. As the number of threads increases, the overhead for managing multiple threads also increases. Therefore, multiple threads(>=2), provide a large overhead compared to small number of elements over a sequential execution. *Maybe* for a larger sized array, multithreading might be efficient.

In conclusion, taking the overhead of creating and managing threads into account, more number of threads doesn't necessarily mean lesser execution time.



As expected, for the same number of threads, the time of execution increases as the size of the array increases.

Note: The times calculated may vary from system to system as the clock() function is dependent on that. Though the values of time might be different, the analysis should stay same on other systems/architectures.