Introduction:

We are implementing a code for searching the nearest point to the source from the given n destination points using the concept of multithreading. We allot equal number of destination points to each thread and each thread uses the Euclidean distance formula to find the distance between source and each destination point that is allotted to this thread. And then it finds the least distance and stores this point and this distance in a struct nearest_Point variable, which in turn is stored in a min_array of type struct nearest_Point. This min_array stores the information of closest point and its corresponding distance (i.e., the struct) from all threads. Then the main thread finds the nearest point among all the points in min_array and thus we are done with our job.

Low-Level Design:

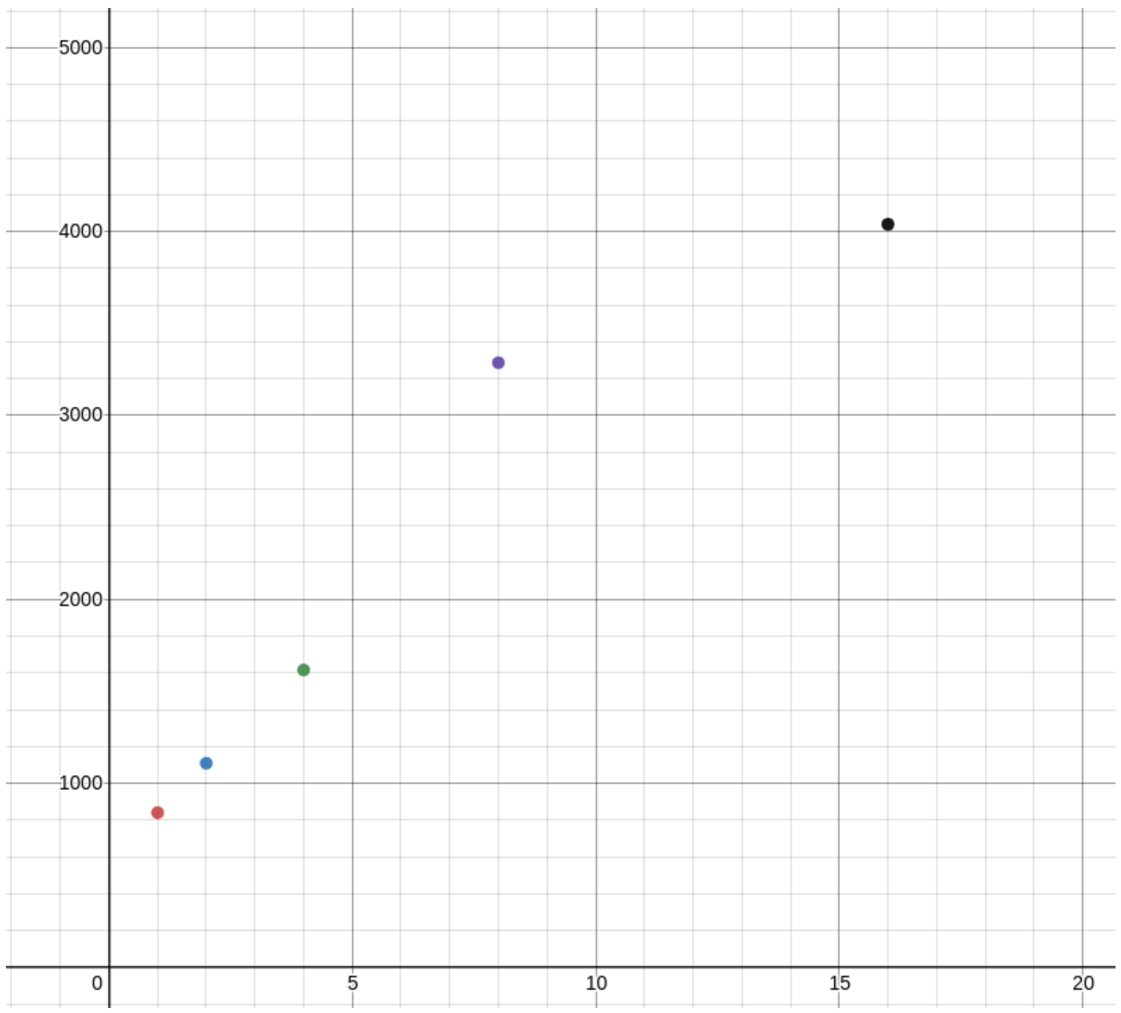
Firstly, the program reads the data from a file named input.txt. The data includes the number of threads, the source, number of destination points and all the destination points. We then store the value of source and destination points in strings and then extract the x-coordinate and y-coordinate by using functions: x_coordinate() and y_coordinate() and store them in int variables for source and in an int arrays for destination points. Also, these inputs are stored in global variables so that all the threads can access them easily.
Then we calculate the number of points that each thread must be allotted by dividing number of destination points by number of threads and store it in a global variable points_per_thread. Then set the thread attributes, set the default attribute of the thread, and then create threads. A function distance() is passed to the pthread_create function along with the thread identifier and the attributes of the thread. distance() calculates the values of the start and end, then it

calls the nearest_point() function. It implements the Euclidean distance formula, finds all distances, and then finds the minimum distance among them and stores it in min_array using stuuct nearest_Point variable. This procedure is done by all threads.
Then we join all the threads by using pthread_join and then the main thread finds the nearest point to the source from the points that are in min_array. Finally, the nearest point is found. This point and the time taken in microseconds for searching this point gets printed on the terminal as output, and we are done.

For calculating the time taken for searching: We have started the clock just before creating threads and stopped the clock just before writing the output into the file. We then subtract these values which gives us the number of clocks that is required for the binary search. To get the time(in seconds), divide the above difference amount by CLOCKS_PER_SECOND which results in the time taken for searching the nearest point.

COMPARISON GRAPHS

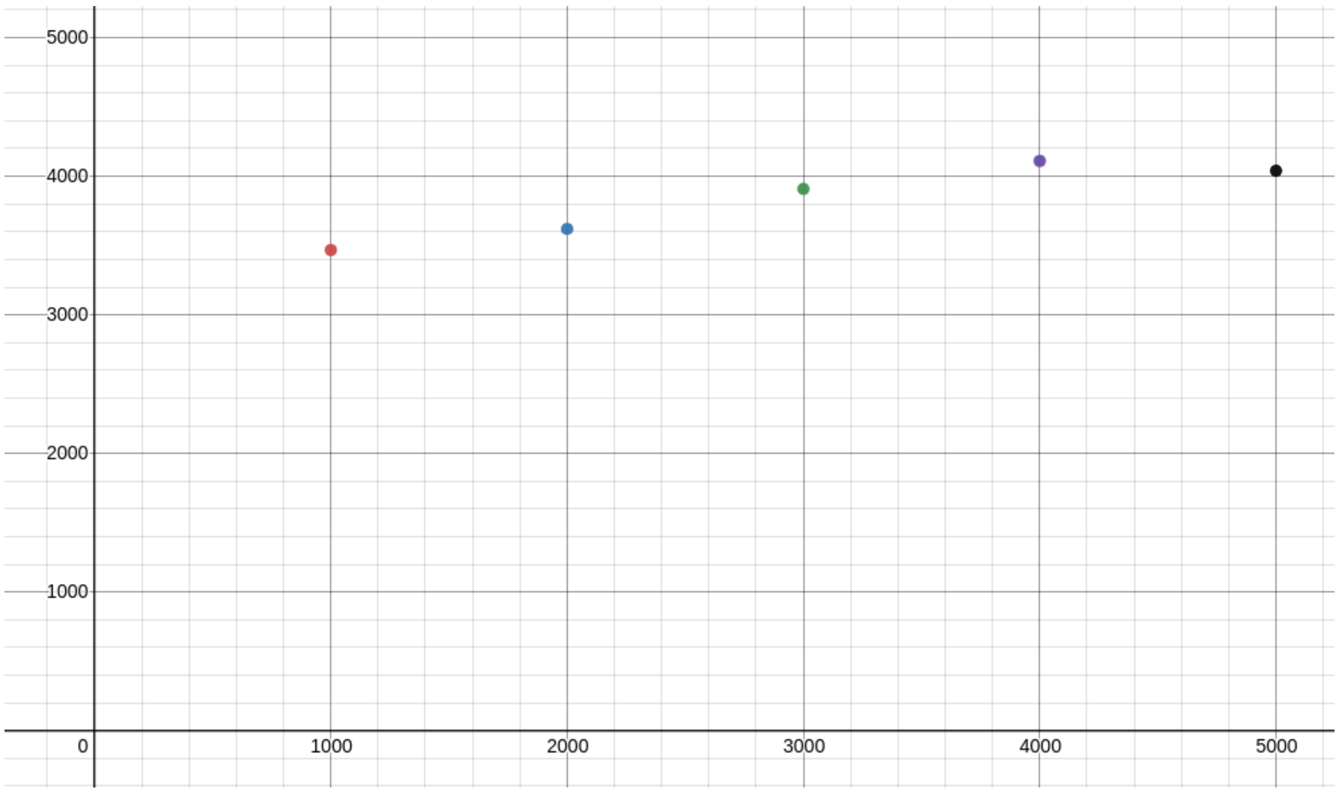
a) Varying the number of threads:



Varying number of threads

x-axis:Represents the number of threads
y-axis: Represents the time taken to search the nearest point from 5000 destination points.

b) Varying the input array size:



x-axis: Represents the size of array
y-axis: Represents the time taken to find the nearest point from varying sizes of destination points using 16 threads in seconds.