THE ARAB AMERICAN UNIVERSITY

FACULTY OF ENGINEERING

Parallel and Distributed Computing

Parallel and Distributed Computing PROJECT 2

ID: 202010285_____

Name: Yahya Azzam_____

Section: _____

1- Brief Description of the Parallelization Approach Using OpenMP:

Using Openmp the binary search algorithm parallelized by dividing the array into chunks, one for each thread. Each thread performs a binary search independently on its chunk

Key Details:

- Work Division: The array is split into chunks → chunkSize = SIZE / thread_count.
- Each thread searches its chunk using binarySearch (start, end).
- When a thread finds the target, it sets the shared result index (resultIndex).
- The critical section prevents race conditions when updating the result.
- Threads check the result to stop early if another has found it .

2- Instructions on how to compile and run your OpenMP code:

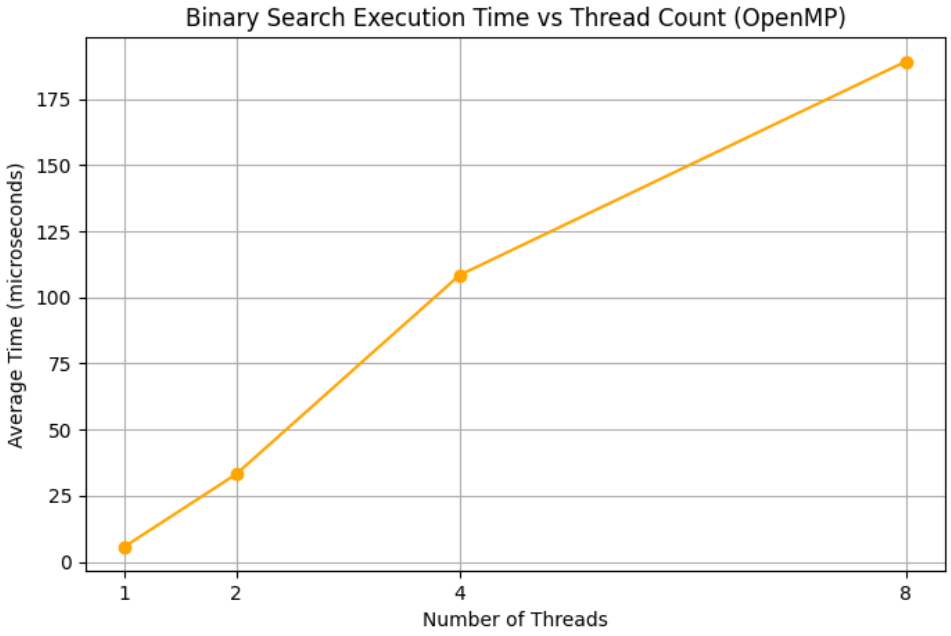
Compiling and running on linux/GCC
g++ -fopenmp binary_search_omp.cpp -o binary_search_omp

run:
./binary_search_omp

3- Instructions on how to compile and run your OpenMP code:

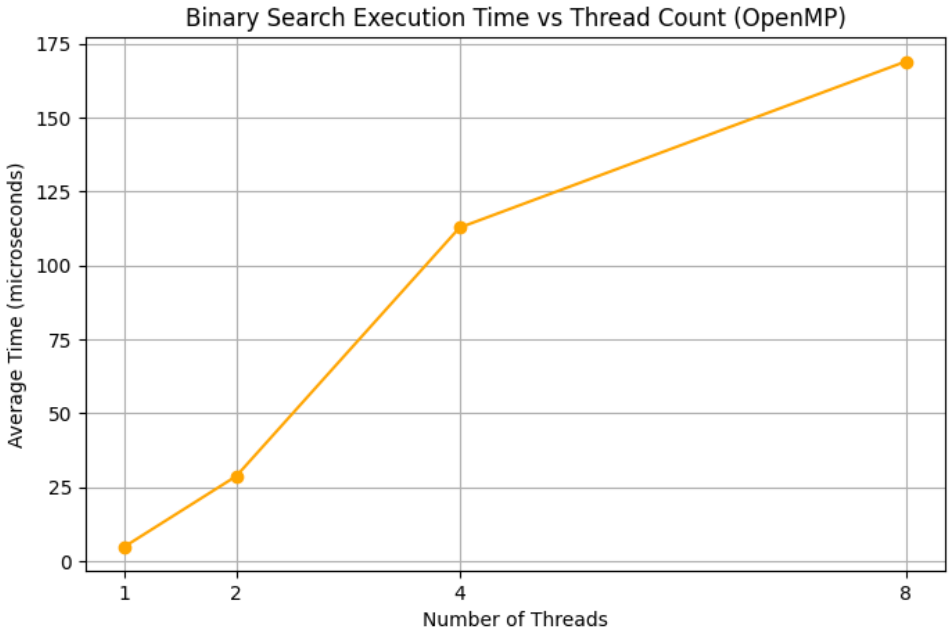
Input size: 10M

Threads count	openmp	sequantial
1	5.8 µS	3 µS
2	33.2 µS	3 µS
4	108.4 µS	3 µS
8	189.2 µS	3 µS



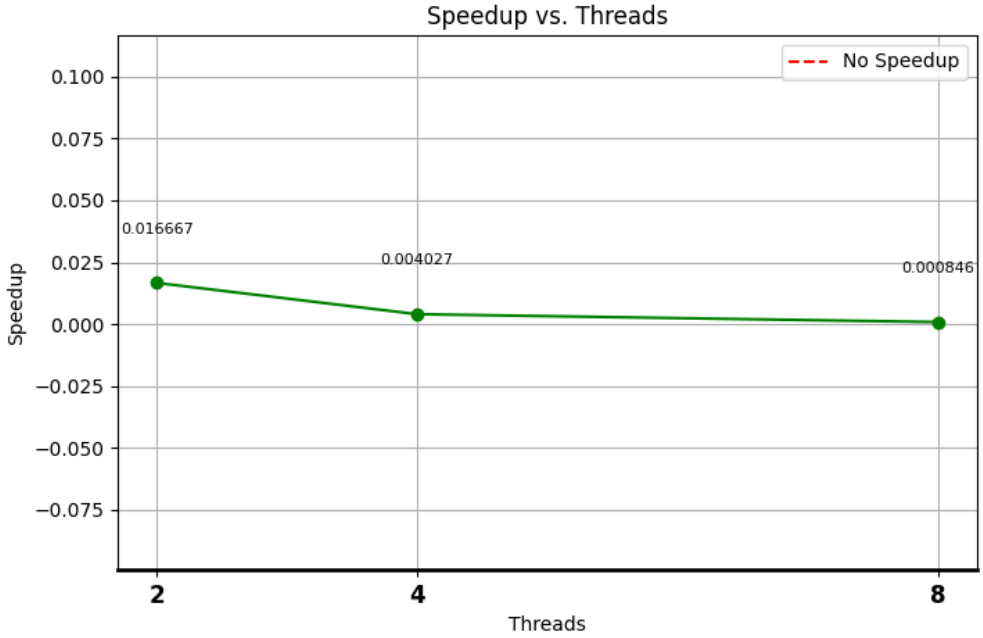
Input size: 0.5M

Threads count	openmp	sequantial
1	5 µS	2 µS
2	28.8 µS	2 µS
4	112.8 µS	2 µS
8	169 µS	2 µS



Speedup

Number of threads	speedup
2	0.016667
4	0.004027
8	0.000846



Hardware:

OS: Vbox linux .

CPU: 2 core cpu

RAM: 4 GB

Compiler: g++ on linux

4- challenges faced:

- **threads terminate** when a thread found the target others threads keep executing.
  Solution :using other condition to check if resultIndex == -1 in the while loop to allow early stopping.

- **No Speedup or Worse Time**
  The reason: Binary Search is Already Fast Binary search has O(log n) complexity due to very small workload, In binary search, only one thread finds the element, while others do redundant or wasted work. Threads may finish unevenly, and some threads may do no useful work at all, creating, and managing multiple threads adds significant overhead. In addition to bad performance of PC used to do the project

Conclusion:

In conclusion, Binary search is not a good candidate for parallelization in practice because it's already efficient and the added threading overhead reduces performance instead of improving it.