PROGRAMMING ASSIGNMENT-2
VALIDATING SUDOKO

DASARI SRINITH
CS21BTECH11015

REPORT :

OVERVIEW OF CODE :

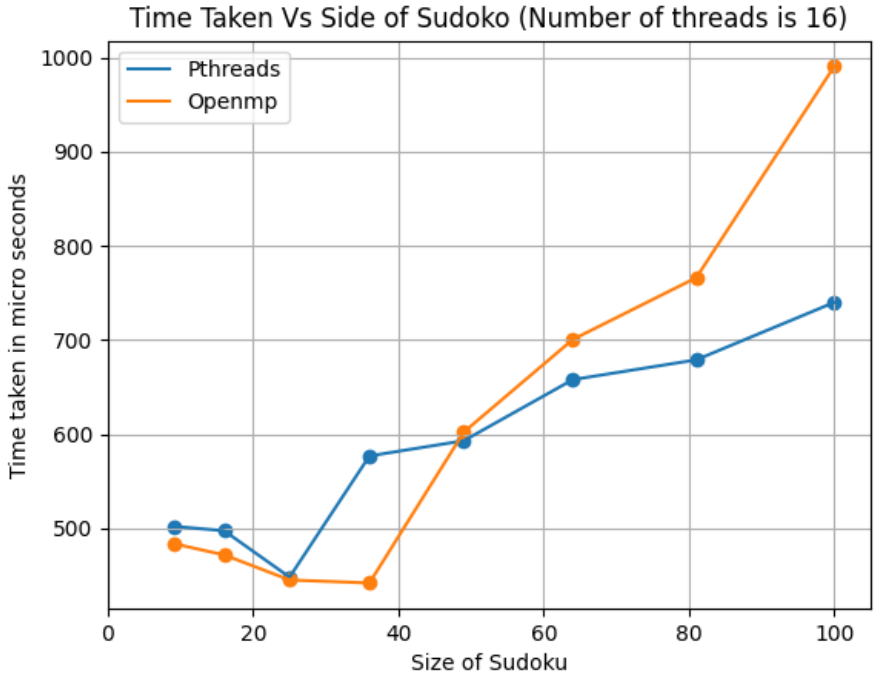
- The code for the assignment, takes 2 inputs, one being K (the total number of threads to be used in the experiment) and other N (the size of sudoko to be used)
- The basic idea used in both the files, one with Pthreads and one with OpenMP is that ,
  - We divide the work of checking 3*(length of side of sudoko) elements(rows,columns,grids) to K threads , equally if possible or at max each thread differs with any other thread by 1.
  - Each thread , updates their work by changing values in an array 'arr' , such that
    1. 0 − That row/column/grid has been checked and is invalid
    2. 1 − That row/column/grid has been checked and is valid

    Here , we are not considering "early termination of threads case"
  - Then , after joining of all the threads , the main thread , checks the array 'arr' and then prints the "Outmain.txt" file which contains the log.

ANALYSIS :

1. **EXPERIMENT 1 :**

   - In this , we are varying the size of Sudoku from 9*9 to 100*100 keeping the number of threads to be 16.
   - We expect the graph to be increasing as the value of size of sudoku increases , which is almost the scenario for both pthreads and openmp in this case
   - The following is the plot for both Pthreads and OpenMP , for experiment 1.



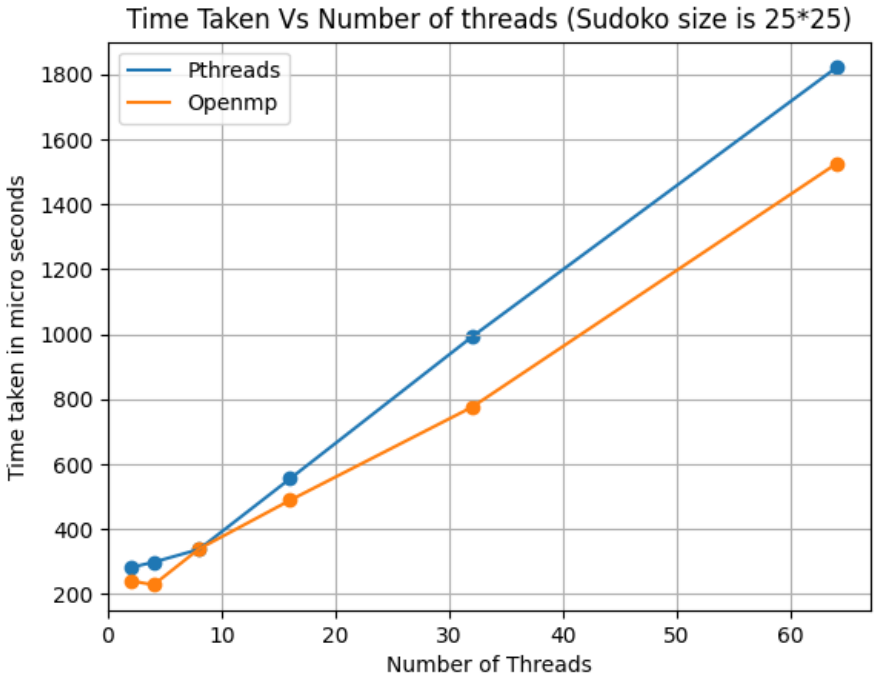
   - For **Pthreads** , it can be observed that , as the size of sudoku increases , the time decreases till n = 25 and then increase till n = 100.
   - This can be explained as follows :
     - For lower size of sudoku's as the size increases , the increase in work for each thread will be very less , for example from change from 9 to 16 the increase in work for each thread will only be one element or atmax 2 elements
     - So , the time for these type of cases will have to almost similar or increase little bit or decrease little bit .
     - And from 9 to 25 , that is what we observe ,"slightly decreasing in this scenario" (The change is in 50-60 micro seconds in the plot , can be seen from data)(it may be increasing or constant , for other data).
     - The data is [502.400000, 497.600000, 448.400000, 577.000000, 593.200000, 658.000000, 679.000000, 740.000000] in micro seconds for pthreads.
   - For **OpenMP** , it can be observed that , as the size of sudoku increases , the time decreases till n = 36 and then increase till n = 100.
   - This can be explained as same as said above and .
     - The increase in time for openmp is higher than pthreads at higher size of sudoku , because OpenMp will work better for sequential and i have used a

sort function which is recursive which will effect the performance of OpenMp , making pthreads better at higher values of size of sudoku.

- The data is [484.000000, 472.000000, 445.400000, 442.200000, 602.400000, 700.600000, 766.200000, 990.400000] in micro seconds for OpenMp.
- It can be observed that Pthreads and OpenMp almost have the same performance , but for higher size , Pthreads was more faster.

2. **EXPERIMENT 2 :**

- In this , we are varying the number of threads from 2 to 64 in the powers of 2 , keeping the side of sudoku to be 25*25
- We expect the graph to be decreasing as the value of size of sudoku increases , but that is not the case here.
- The following is the plot for both Pthreads and OpenMP , for experiment 2.



- The reason for the exception could be because of other latencies.
- For **Pthreads** and **OpenMp**, it can be observed that , as the size of sudoku increases , the time increases till n = 100.
- This can be explained as :
  - The values which we took for the experiment N=25 is very small for higher values of number of threads
  - We dont need so many threads to do small computations.
  - It will only result in more overhead from thread creation and more stack duplication for those threads to work on.
  - So , the time for these type of cases will have to increase as the number of threads increases.
  -

- It can be observed that Pthreads and OpenMp almost have the same performance , Pthreads is computing more faster than OpenMp , for the same reason said above recursive code.