Report_CE21BTECH11008

Reading The file

My input file name is “read.txt” and output file name is “OutFile.txt” . For **OutFile** I have made a global file pointer as it can be written by any thread while for input file made file pointer in the main () function. “Read.txt” contain a line. Extracting the M and N from this by simple reading and casting into integer using **atoi**. At last using fclose closed the all the file.

To measure Time

Used clock_t data type and made initial and final counter of the time then subtracted and typecast to double for better accuracy. Used clock.h library.

Vampire Number

Vampire number is the composite number of even digit such that it can be factored into half as many digit as original number such that each digit in factor will be present in number exactly ones. Also both the factor shouldn’t have trailing 0

Removal of extra Number

As the definition it should have even digit. So I have made one function name **digit_count** which simply works on the division and then checking number become zero or not. This I have used to checking the number. If number have odd digit then I have discarded that. Also Vampire number cant be prime thus made **check_prime()** to decide the thing.

Threading Methodology

Think various method. First method directly come was dividing into equal and assign the continuous set of number to given set but it will be inefficient. As my method is based permutation that is finding recursion. It will have order of n^2(where n is the number of digit) . Thus, larger number will take more resource and time. Lower thread number will be having very less work than higher thread number thus last thread will be doing its work while initial thread will stop execution after some time and become idle.
To encounter this, I came up with the idea of modulus. In my case, taking the modulo of the given number with maximum number of thread and assigning the things to each thread. Now almost all thread will get approx. same type of data and same count of data thus provide better utilization of the thread.
In code made 2d array that takes the number come after modulus and thus store uniquely such that that it is taking 0(n)

Implementation of vampire number

For this I have used the concept of recursion and find the permutation of the given number. For the given number stored as character that essentially using the recursion name permutate that is of type bool that will tell return true if found else false. For permutation in this make one **swap_fun()** function to permutate passing by address. When it is reaching to the base case it means complete permutation (unique) has been created and thus passing in **“check_vampire()”** function. That partition into the two parts of equal digit and I am also passing character array thus has kept unaltered value of the number and comparing with that.

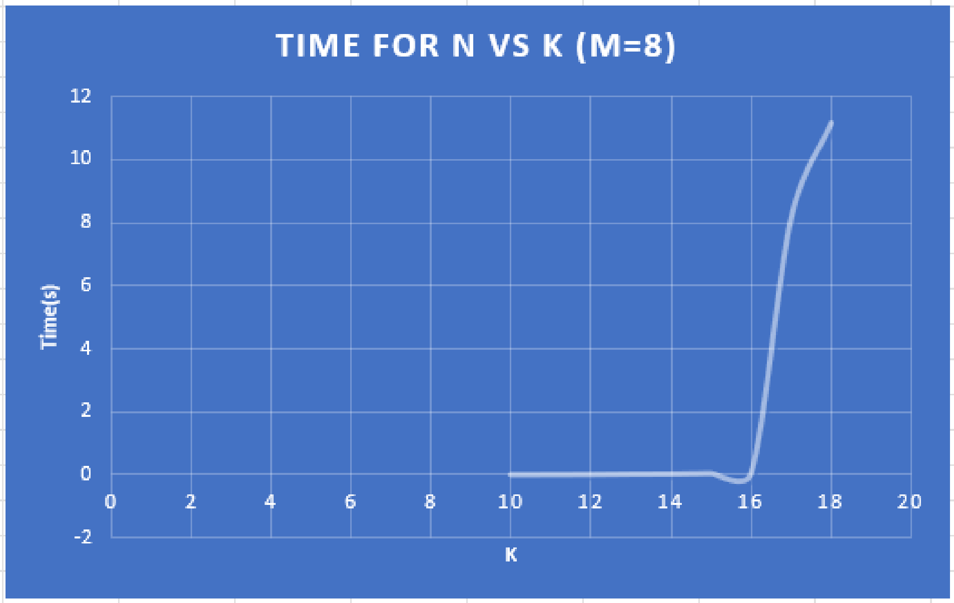
Inputting to the “OutFile.txt”

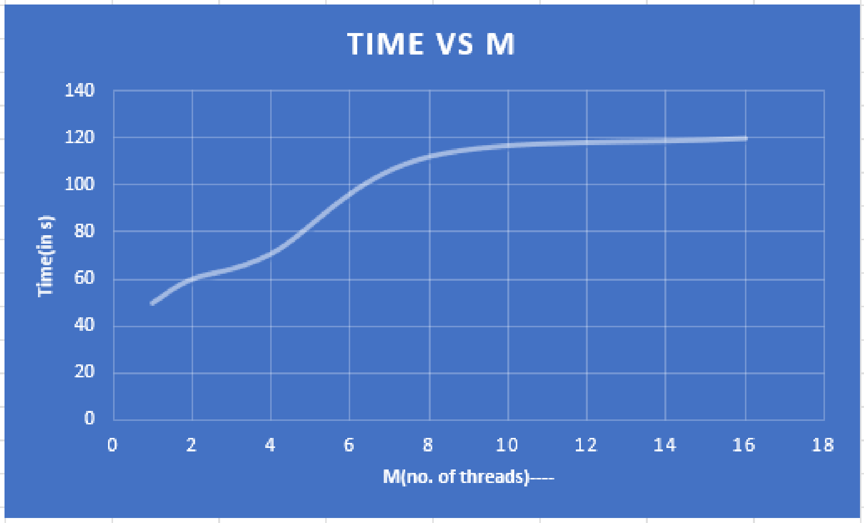
using **fprintf** function that is printing if found and corresponding thread number (note adding 1 to thread number as our index of the thread variable start with 0 but actually it will be thread 1.

Implementation of threading

First using **find_vampire** function passed the argument set in the main function. Introduced the library **“pthread.h”.** Making one array of type threads and size number of thread. As argument passing function named **“find_vampire_no”** and passing argument to the function as void pointer. Used struct to make argument as user defined data type that is containing three integer value.
After this completes then joining the thread and removing the partition.

In **“find_vampire_no”** function restore the value of the argument using dereferencing the pointer. And size of each array processed by each thread will be n/m approximately. After finding the digit using **digit_count** dynamically allocated the memory using malloc and **sprintf** to cast into character array and finally passed through the “permutate()” that is doing further implementation.
At last free the dynamically allocated memory.
Rest things are commented beside code.


TIME FOR N VS K (M=8)
Time(s)
K


TIME VS M
Time(in s)
M(no. of threads)----