Assignment 2:Vampire nummbers

Gaurav Choudekar - CS22BTECH11015

December 3, 2023

1 Introduction

This report is regarding assignment 2 conducted on finding vampire numbers with threads.

2 About Code

The code is written in C.all vampire numbers are found till given input number and represented in OutMain file with their thread numbers.

3 Input format

A file is given as input,name of file is modified accordingly.there are only two numbers in file seperated by space bar,first represent upper limit to find vampire numbers and second one to represent number of threads used.

4 Basic idea of resource usage

For each thread,one node is made and in it with help of array,all numbers are stored.also total attribute is used in it to store number of numbers in that array.when a thread is created,function is called for computing numbers stored for that node/process,for this a node is given as input for each thread create call.

5 Code Flow

5.1 Main Function

In the main function take input from file and make our nodes ready according to that,they are first initialised,then numbers are evenly distributed in it.for distributing numbers in it,we start from last,that is,end point of computing vampire numbers and consecutive numbers to different/consecutive threads.in this way we end up assigning with mod(m) logic .after each thread create,we send that specific node to check function for checking vampire numbers.after joining particular thread,it will print numbers it found as vampire in output file.

5.2 Check Function

Aim of check function is to check each number in node if it is vampire number or not.first,it will one by one call all numbers assigned to it and call checkvamp function,checkvamp will return 1 or 0 according to its status about being vampire number and it can store it,if it is,in local array.

5.3 checkvamp function

It will first calculate number of digits in that number,if it is odd then it cannot be a vampire number.it will form array and store its digit in it,then it will form every combination for 2 numbers and check if their multiplication is equal to original number or not.it will form first number from array and second number from remaining numbers from array.for different number of digits,different if clauses are made.

5.4 Accuracy Check complications

5.5 Accuracy check

Before giving big inputs,i gave small inputs so that it became easy to verify from output file if it is giving correct output or not.whenever i got error or wrong output,i gave printf statements in between to check if that function or part of function is reached or not,also if needed,i printed variable values to check if their correct value is reached or not to confirm that code till that part is correct.at the end,when i became confident about small values,and later big values i cross-checked my output values/result with peers to become double sure about it.

5.6 complications

First of all while coding for this problem,i was giving direct input in code through variables rather than file input,so after shaping it in required format,i got many segmentation faults and wrong values for given inputs.this was my first time working with threads,so it was little interesting and difficult for me to use it and give parameters to it.As i used node,i would get segmentation fault or no output for wrong result,i required to correct it and keep the current sizes of variables.it was little difficult for me to record execution time,so i used some online resources to see what can i use for it.forming combinations of numbers also was little difficult or lengthy for me.

5.7 comments in my code

I am intentionally not deleting commented parts of my code so that it indicates my printf statements where i was needed to use them for accuracy check and dealing with complications.

6 Graphs

Include the graphs as specified in the assignment.

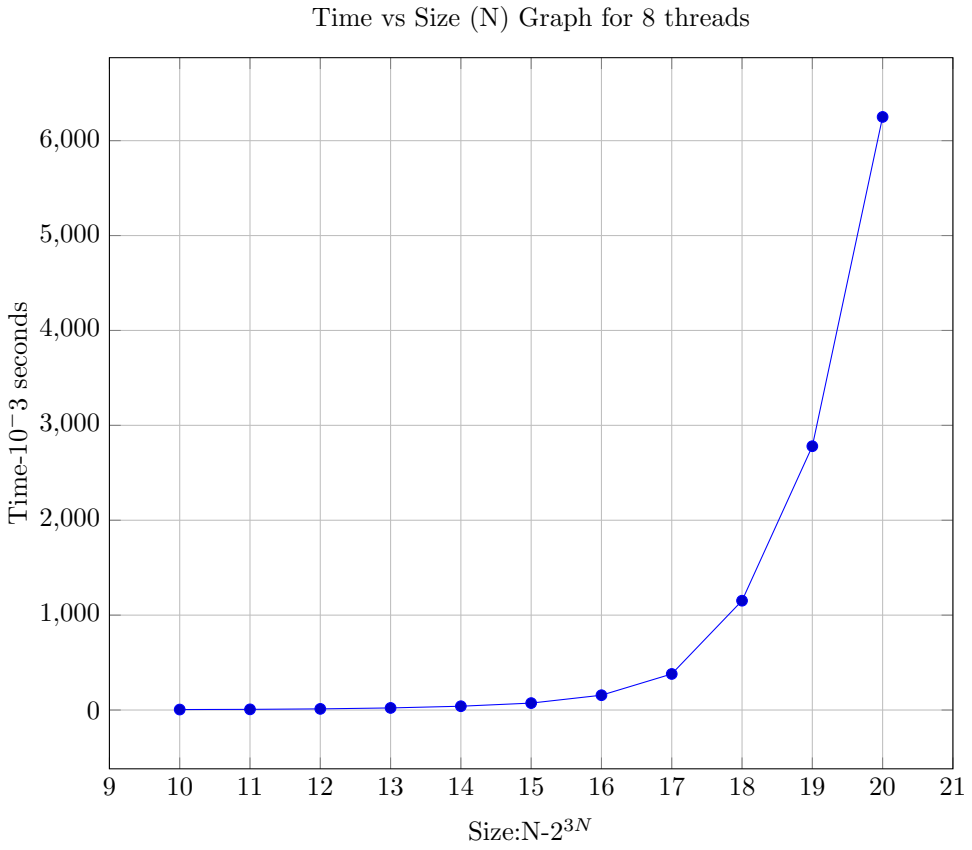

Figure 1: Time vs Size (N) Graph

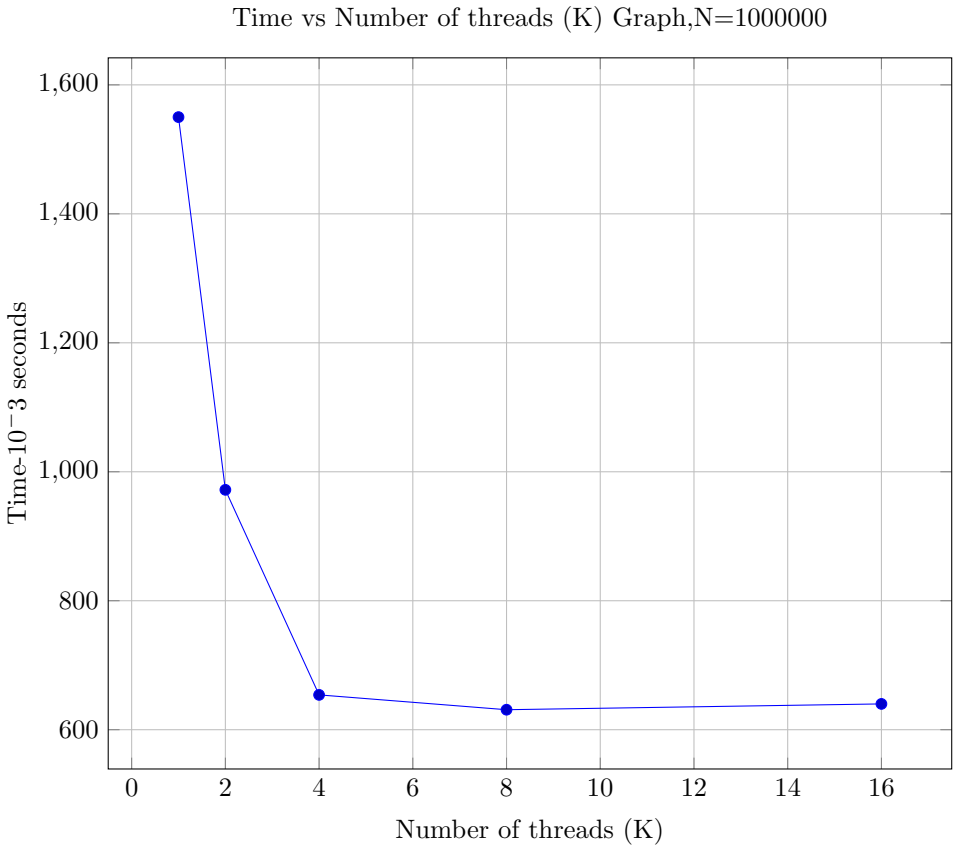

Figure 2: Time vs Number of threads (K) Graph

7 Output analysis

for fixed number of threads,as we increase number of operations,that is,number of numbers to compute, time required for execution/their computation increases.this is due to that fact that one thread requires to compute more number of numbers.For fixed number of numbers,as we increase number of threads,time required for their exection decrases.this is due to that fact that one thread requires to compute less number of numbers.this facts can be seen easily with the help of graphs.as number of numbers to compute increases,time execution increases more and more rapidly.growth/decrase in these graphs is not linear,it is exponential or rapid.for odd number of digits vampire numbers are not present in its set.

8 Improvement scope

Output obtained is not equal in threads,which means it could be divided more properly between different threads.memory allocation can be more proper,that is,it can be allocated as per needed,not how much we can.due to these there is scope for better time complexity and space complexity.error handling could be implemented better.memory cleanup could also be done better.