CS F422

PARALLEL COMPUTING

ASSIGNMENT – 1

REPORT

Anirudh Kumar Bansal	2014A7PS081P
Lakshit Bhutani	2014A7PS095P

March 5, 2017

<u>Problem – 1</u>

Longest Common Subsequence of Documents

Table of Contents

Abstract
The problem involves computing the pair-wise length of LCS (longest common-sub-sequence) of N documents by considering each line as a sequence. Top K pair of documents with longest LCS lengths are selected and pairwise LCS of these documents computed. Finally intersection of the content in the K' documents is calculated. The problem has been done using OpenMP (*Open Multi-Processing*).

Usage Instructions
The source code is in 'lcs_documents.c'. To compile the code -
gcc -fopenmp lcs_documents.c

[Make sure 'test1', 'test2', 'test3' , ... files are present in the same directory before running. The maximum number of documents is 100 but user can work with any number of documents by changing the MAX_N parameter in the code.*]*

*T*o run the code -
./a.out

Then enter the value of 'N' and 'K' as prompted. Do ensure that value of K is less than (N * (N + 1)) / 2.

The result on the terminal displays the time taken by the code. The output file of the intersection of documents will be created with name 'out_lcs'.

Make sure that the documents are named as *text1* , *text2, text3, ... , textn*. The documents used for testing are provided in '*test_cases_1*' folder.

Program Structure
The program follows shared memory model. In this specific case the shared memory is the hash values of the lines in all documents stored in *all_hash* the pair wise LCS lengths stored in *lcslen.*

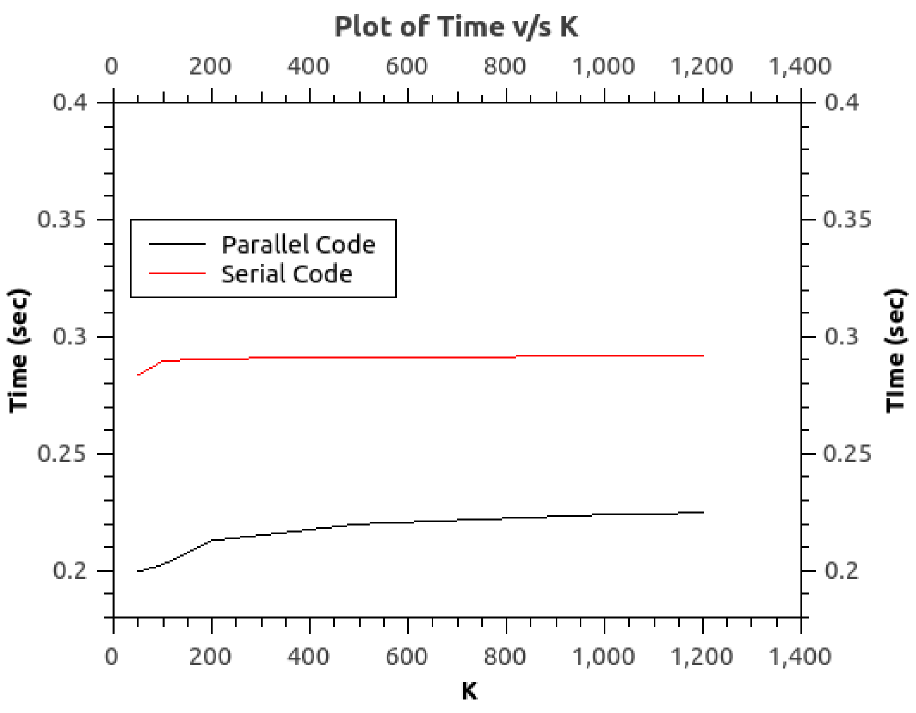
- The documents are read by individual threads and hash value of individual strings stored in *all_hash*.
- Then individual threads calculate the pair-wise LCS of the documents. And store it in *solution* array.The *solution* array also stores the indices of the two documents whose LCS is being computed.
- The solution array is sorted using quick sort to find all the documents involved in the top k pairwise LCS values.The indices of these documents are stored in *topk* array.
- The document in *topk* with the least number of lines is found and its lines are matched with all the lines of the remaining documents.Two lines are compared using their hash values stored in *all_hash* array.The lines present in the intersection of all these documents is written to the file *out_lcs.*

Experimental Results

Test document generation : A random paragraph of 100 lines with line size less than 550 characters was saved as *test* file. Test files *test1, test2, ... , test100* was generated by picking random lines from *test* file. It was ensured that there are some lines common to all test files for checking purposes. The document also contains duplicate lines. Although the length of all documents is the same *i.e.* 100 lines each, but the code is generic to handle variable size files.

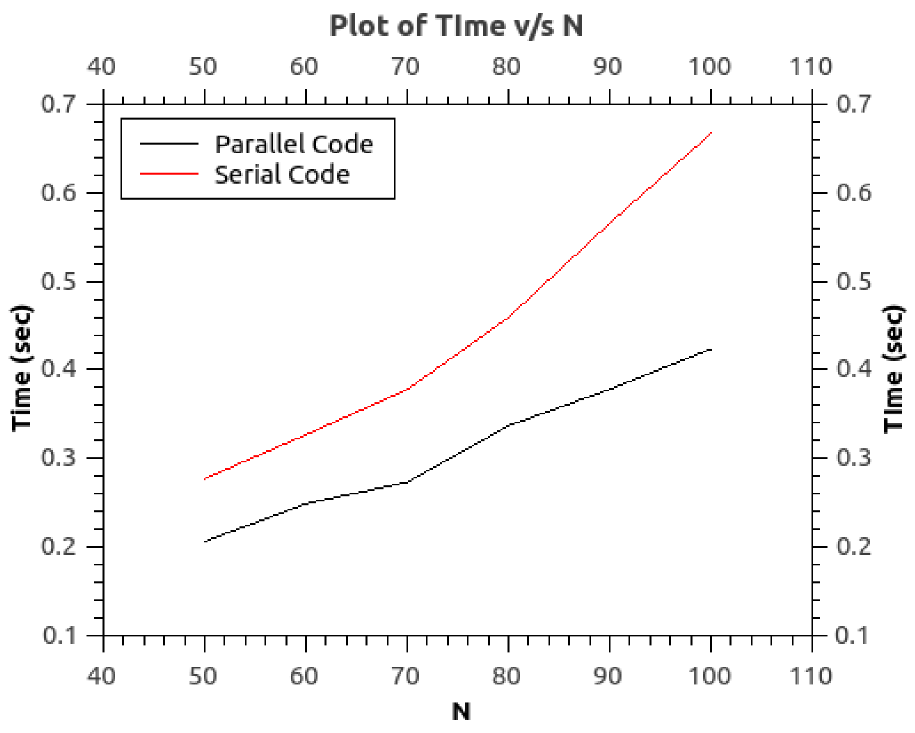
1. Variation of K keeping N constant

N	**K**	**Serial code time (sec)**	**Parallel code time (sec)**	**Speedup**
50	50	0.283591	0.199634	1.42
50	100	0.289639	0.202635	1.43
50	200	0.290595	0.212811	1.36
50	500	0.291091	0.219922	1.32
50	1000	0.291662	0.223968	1.30



2. Variation of N keeping K constant

N	K	Serial code time (sec)	Parallel code time (sec)	Speedup
50	500	0.277310	0.206526	1.34
60	500	0.327013	0.248946	1.31
70	500	0.377949	0.273272	1.38
80	500	0.459250	0.337102	1.36
90	500	0.565748	0.378108	1.50
100	500	0.666870	0.423691	1.57



Best parallelism and speedup

With N constant at 50 , the best speedup obtained is 1.43 when K is 100.

With K constant at 500, the best speedup obtained is 1.57 when N is 100.