**Sabancı University**

Faculty of Engineering and Natural Sciences

**CS406-531 Parallel Computing / Parallel Processing and Algorithms**

**Spring 2016-2017**

**Homework 1**

**Due: 23/10/2017 - 23:00**

(One day late submission penalty: -10%)

|  |
| --- |
| **PLEASE NOTE:**  **Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!**  **You HAVE TO write down the code on your own.**  **You CANNOT HELP any friend while coding.**  **Plagiarism will not be tolerated!** |

1. **Introduction**

In this assignment, you will implement a breadth-first search algorithm with multiple threads. The graphs will be sparse and they will be in Matrix Market Format. You will read these graphs into a Compressed Row Storage (CRS). For further information about MM and CRS formats, you can check the links below.

**http://netlib.org/linalg/html\_templates/node91.html**

**http://math.nist.gov/MatrixMarket/formats.html**

The functions for reading the sparse graph is given as well as the structure of CRS. You will just implement (and optimize) the BFS step. Any preprocessing on the graph is allowed. A report of at least 1 page should also be submitted including:

* A general explanation of the implementation
* Execution times and speedups
* Tricks for parallelization (scheduling, preprocessing etc.)

If you don’t know how BFS algorithm works, you can check

https://en.0wikipedia.org/wiki/Breadth-first\_search

1. **Program Flow**

You will read the sparse graph (which is in Matrix Market format) into a Compressed Row Storage (CRS) data structure. Then a Breadth-First Search (BFS) will be performed starting from a given source node. The index of this source node will be a command-line input. At the end your program should print the source node and the distances of each node into a file. The first line should be the index of the source node and the second line should contain distances separated by spaces.

For each graph, you should carry out experiments with {1,2,4,8,16} threads and report the runtimes.

Your report will be as important as the implementation itself. A report of at least one page should be submitted with the code, including:

* A general explanation of the implementation
* Execution times, speedups and efficiency etc.
* Tricks done for parallelization (scheduling, preprocessing etc.)
* How you compile & run your program

1. **Working on Nebula**

After connecting to Nebula, you should (optionally) create a new directory for your homework. To create a new directory for this homework you can use the command mkdir. If you are using Windows, you should first download WinSCP and use that to copy the codes to Nebula.

There is a Makefile included in the homework. If you type **make** in the directory in which you copied the given codes, then an executable with the name *bfs* will be created automatically. You can change the Makefile if you wish, but it’s not recommended.

After compiling, to run your program you can use the line below:

.**/bfs <sparseGraphFileName> <sourceNodeIndex>**

To download a matrix from a link, you can copy the link address and use the command **wget** to download the *tar* file. After that, you can use **tar -zxvf** to decompress the tar file.

A sample matrix and the result file is also attached in the homework.

In your homework you should test your code on four graphs:

* Usroads (<https://sparse.tamu.edu/Gleich/usroads>)
* M6 (<https://sparse.tamu.edu/DIMACS10/M6>)
* Hugetrace-00020 (<https://sparse.tamu.edu/DIMACS10/hugetrace-00020>)
* Europe\_osm (https://sparse.tamu.edu/DIMACS10/europe\_osm)

If you want to test your implementation on additional graphs, you can find sparse matrices at the link below.

<https://www.cise.ufl.edu/research/sparse/matrices/>

Note: It would be nice if you delete the matrices after submitting the homework since 25 copies of each graph would take a lot of space on Nebula 😊

1. **Some Remarks**

In the grading process three things will be checked:

* Correctness of your implementation
* Speedup and efficiency values
* How well the report is written

The fastest running codes will also get a bonus point. Please note that we are aware of the fact that there are multiple parallel BFS implementations online and will realize if you try to use one of them.

Please don’t forget to submit your code and the report together.

**What and where to submit (PLEASE READ, IMPORTANT):**

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Name your cpp file that contains your program as follows:

***“SUCourseUserName\_YourLastname\_YourName\_HWnumber.cpp”***

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

***Cago\_Ozbugsizkodyazaroglu\_Caglayan\_hw2.cpp***

Do not add any other character or phrase to the file name. Make sure that this file is the latest version of your homework program. Compress this cpp file using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. **Please make sure that you also include your report to the compressed file.** Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct file. The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows:

***SUCourseUserName\_YourLastname\_YourName\_HWnumber.zip***

For example zubzipler\_Zipleroglu\_Zubeyir\_hw1.zip is a valid name, but

***hw1\_hoz\_HasanOz.zip, HasanOzHoz.zip***

are **NOT** valid names.

**Submit via SUCourse ONLY!** You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

CS406-531 Team (Mustafa Kemal Taş, Kamer Kaya)