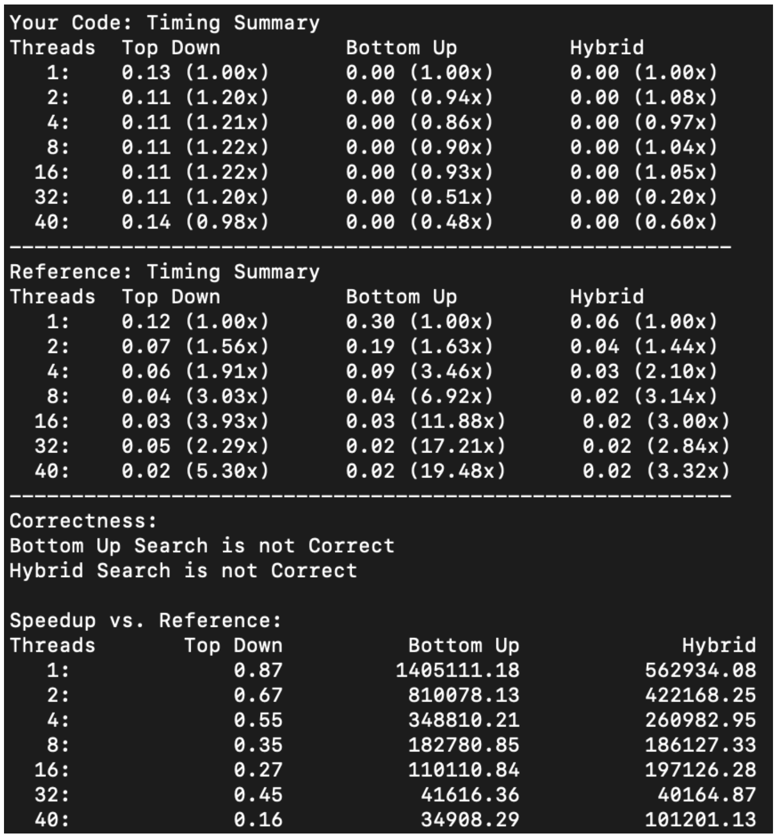
并行与分布式计算导论 作业 2
PDC 2023s Homework 2

截止期限 2023 年 4 月 2 日 23:59
DDL: 2023 Apr. 2 23:59 (GMT+8)

利用 **OpenMP** 并行编程，
Programming in **OpenMP**,
本次作业需要实现并利用 OpenMP 加速常见的图算法：宽度优先搜索算法（BFS）。在多核机器上，好的实现和加速可以在数秒内完成包含百万个边的图的 BFS。本次作业需要实现自顶向下、自底向上和混合（可选）的三种 BFS 搜索，可以参考 BFS 重新熟悉这一算法。代码利用 BFS 计算所有节点与 0 号节点的距离。
In this assignment, you will implement a graph processing algorithm: breadth-first search (BFS). A good implementation of this assignment will be able to run these algorithms on graphs containing hundreds of millions of edges on a multi-core machine in only seconds. You need to implement and accelerate three kinds of BFS algorithms: top-down, bottom-up, and hybrid (Optional). You can see here and for helpful references on BFS. The code uses BFS to compute the distance to vertex 0 for all vertices in the graph.
请实现以下四个任务，并撰写书面报告，简述你的实现思路，测试结果和分析。将书面报告和代码源码共同打包为压缩包上传至教学网，同时在课程服务器中建立文件夹 hw2，并将代码上传至文件夹。
Please finish the following four tasks and write a report containing your programming idea, test results and analysis. You need to pack up your report and source code together and upload to Course website. Please create a directory named "hw2" on the course machine, and upload your code to the directory.

任务 1
完成环境配置。将附件中压缩包上传至教学服务器（或你要使用的机器），使用命令 tar -xzvf hw2.tgz 解压。进入 /hw2/bfs 目录，使用命令 make，编译生成可执行文件 bfs。本次作业需要使用的数据集存放在可公共读取的目录 /pdchome/hw2_graphs 中，为节约磁盘空间，请各位同学不要复制数据集到自己目录下。可以在 hw2/bfs 目录中，使用命令 ./bfs <path_to_graph> 测试可执行文件 bfs 是否可以正确运行，我们已经编写了自顶向下的串行版本 bfs，你应该可以看到类似下图的运行结果。
Task1
Set up the environment. You need to upload the attached file to course machine (or your own machine), and untar the file with "tar -xzvf hw2.tgz". Execute the following command: "cd <path_to_hw2>/bfs; make" to generate executable file bfs. The dataset for this home work is in a readable shared directory "/pdchome/hw2_graphs/". DO NOT copy it to your own directory on course machine to save disk space. You can use "cd <path_to_hw2>/bfs; ./bfs <path_to_graph>"

to test whether program "bfs" can execute correctly. We have implemented a serial version of top-down BFS for you, and you should see a similar result in the figure below.



```
Your Code: Timing Summary
Threads   Top Down          Bottom Up         Hybrid
   1:     0.13 (1.00x)      0.00 (1.00x)      0.00 (1.00x)
   2:     0.11 (1.20x)      0.00 (0.94x)      0.00 (1.08x)
   4:     0.11 (1.21x)      0.00 (0.86x)      0.00 (0.97x)
   8:     0.11 (1.22x)      0.00 (0.90x)      0.00 (1.04x)
  16:     0.11 (1.22x)      0.00 (0.93x)      0.00 (1.05x)
  32:     0.11 (1.20x)      0.00 (0.51x)      0.00 (0.20x)
  40:     0.14 (0.98x)      0.00 (0.48x)      0.00 (0.60x)
----------------------------------------------------------
Reference: Timing Summary
Threads   Top Down          Bottom Up         Hybrid
   1:     0.12 (1.00x)      0.30 (1.00x)      0.06 (1.00x)
   2:     0.07 (1.56x)      0.19 (1.63x)      0.04 (1.44x)
   4:     0.06 (1.91x)      0.09 (3.46x)      0.03 (2.10x)
   8:     0.04 (3.03x)      0.04 (6.92x)      0.02 (3.14x)
  16:     0.03 (3.93x)      0.03 (11.88x)      0.02 (3.00x)
  32:     0.05 (2.29x)      0.02 (17.21x)      0.02 (2.84x)
  40:     0.02 (5.30x)      0.02 (19.48x)      0.02 (3.32x)
----------------------------------------------------------
Correctness:
Bottom Up Search is not Correct
Hybrid Search is not Correct

Speedup vs. Reference:
Threads        Top Down         Bottom Up             Hybrid
   1:              0.87        1405111.18          562934.08
   2:              0.67         810078.13          422168.25
   4:              0.55         348810.21          260982.95
   8:              0.35         182780.85          186127.33
  16:              0.27         110110.84          197126.28
  32:              0.45          41616.36           40164.87
  40:              0.16          34908.29          101201.13
```

任务 2

并行加速自顶向下的 BFS。修改 hw2/bfs/bfs.cpp，利用 OpenMP 对函数 bfs_top_down 进行加速（在合适的地方插入合适的 Pragma）。你可以参考 hw2/common/graph.h 和 hw2/bfs/bfs.h 以熟悉程序中存储图的数据结构和接口。

Task2

Accelerate top-down BFS. Modify "hw2/bfs/bfs.cpp" and use OpenMP to accelerate the serial version. You can get familiar with used graph data structure and APIs in "hw2/common/graph.h" and "hw2/bfs/bfs.h".

任务 3

实现并加速自底向上的 BFS。修改 hw2/bfs/bfs.cpp，实现函数 bfs_bottom_up 并利用 OpenMP 对其进行加速，需要保证程序在所给数据集上的正确性。实现完成后利用 make 重新编译可执行文件 bfs 并测试，程序会反馈正确性。可以参考以下自底向上的 BFS 伪代码。

Task3

Implement and accelerate bottom-up BFS. Implement "bfs_bottom_up" in hw2/bfs/bfs.cpp and accelerate it with OpenMP. You can use command "make" to regenerate a new executable file "bfs" and test its accuracy and speedup. Basic pseudocode for the algorithm is as follows:

```
for each vertex v in graph:
    if v has not been visited AND
       v shares an incoming edge with a vertex u on the frontier:
          add vertex v to frontier;
```

任务 4（可选）

实现并加速混合的 BFS。自底向上和自顶向下的 BFS 每一步在不同图的性质（如 frontier 的大小）中有不同的表现，可以动态选择以提升性能。修改 hw2/bfs/bfs.cpp ，实现函数 bfs_hybrid 并利用 OpenMP 对其进行加速，需要保证程序在所给数据集上的正确性。实现完成后利用 make 重新编译可执行文件 bfs 并测试。

Task4 (Optional)

Implement and accelerate bottom-up BFS. In some steps of the BFS, the bottom-up BFS is significantly faster than the top-down version. In other steps, the top-down version is significantly faster. Implement function "bfs_hybrid" in "hw2/bfs/bfs.cpp" and accelerate it with OpenMP. You can use command "make" to regenerate a new executable file "bfs" and test its accuracy and speedup.

撰写书面报告：在完成上述任务后，撰写报告阐述你的编程思路、测试结果和分析，报告要求尽量**简洁**。**不要**粘贴代码截图。

Report: Along with your code, we would like you to hand in a clear but **concise** high-level description of how your implementation works as well as the test results and analysis. **DO NOT** paste your code in your report.