# INF236
# Assignment 2
Alina Artemiuk

The problem of the assignment is to develop two parallel Breadth-First Search (BFS) algorithms, pbfs and abfs, and experiment with different strategies to optimize their performance. In both algorithms, the exploration from a particular layer in the BFS should be done in parallel. In pbfs, the vertices discovered by each thread at a given distance should be stored locally before being placed in a common array S. In abfs, the algorithm should start by executing the BFS algorithm sequentially and then switch to a parallel scheme at some point. The vertices discovered by each thread in abfs should be stored locally, and only copied back to S every $k^{th}$ iteration.

---------- SBFS ----------

sbfs results

| road_usa | delaunay_n24 | hugebubbles-00020 | rgg_n_2_22_s0 |
|---|---|---|---|
| 1.832295 | 1.433672 | 3.837764 | 0.982941 |

---------- PBFS ----------

The algorithm starts by initializing the parent pointers (p) and distance from the starting vertex (dist) to -1 for all vertices except the starting vertex (vertex 1) which has its parent and distance initialized to itself and 0 respectively. The queue S is initialized with vertex 1. Since all parameters passed are shared, S[0] stores the number of vertices in S (num_r in the sequential version) for communication between threads.

The while loop continues until there are no more vertices in the queue. The work is divided among the threads using OpenMP's parallel for directive, where each thread processes a subset of the vertices in the queue. For each vertex v, the algorithm checks its neighbors (w) and if the neighbor has not been visited before (its parent pointer is still -1), it sets its parent pointer to v and updates its distance from the starting vertex as the distance of its parent plus one. The newly discovered vertices are added to a local temporary array loc_T.

After each iteration, the discovered vertices in each thread's temporary array are gathered. The barrier statements ensure that all threads have finished their work before proceeding to the next step. The total number of vertices discovered is computed and the indices for the discovered vertices in the queue are updated using a prefix sum operation (T array).

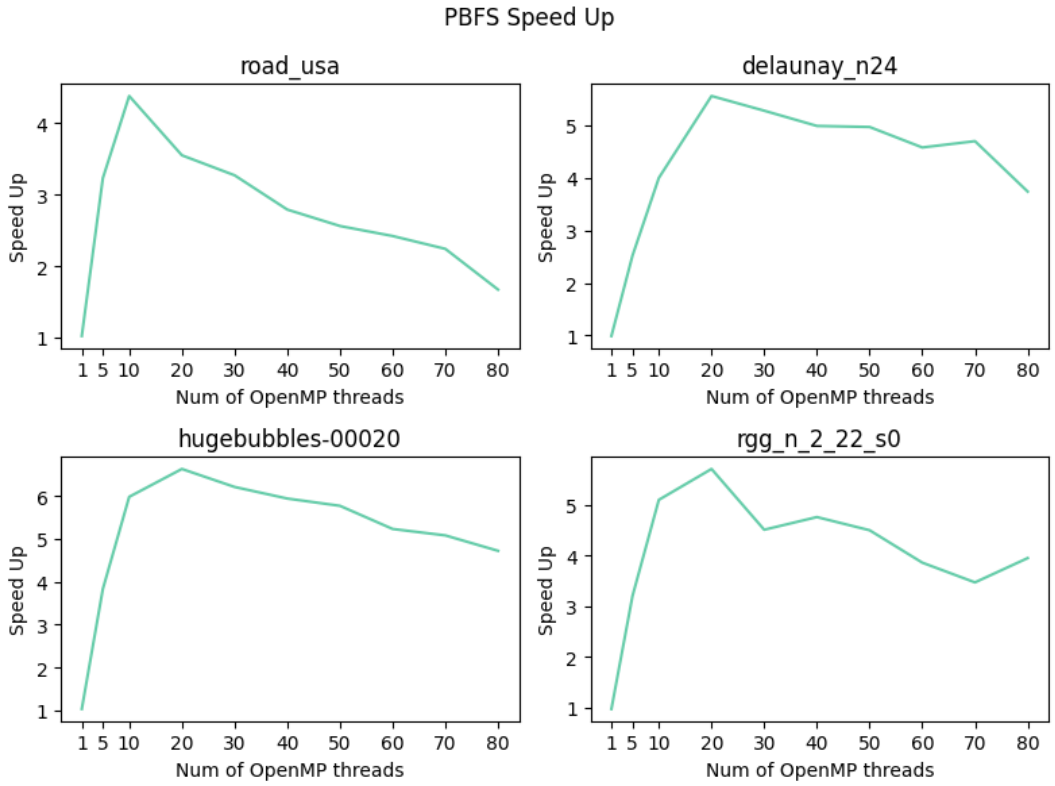
Finally, the discovered vertices are added to queue S in the correct order using the prefix sum indices computed in the previous step.

pbfs results

| threads | road_usa | delaunay_n24 | hugebubbles-00020 | rgg_n_2_22_s0 |
|---|---|---|---|---|
| 1 | 1.789829 | 1.445662 | 3.709301 | 1.008198 |
| 5 | 0.567278 | 0.56867 | 1.001591 | 0.307518 |
| 10 | 0.418127 | 0.358583 | 0.641881 | 0.192715 |
| 20 | 0.516541 | 0.257947 | 0.578872 | 0.172161 |
| 30 | 0.560186 | 0.271333 | 0.618444 | 0.217917 |
| 40 | 0.65735 | 0.287496 | 0.646044 | 0.206651 |
| 50 | 0.715254 | 0.288555 | 0.66544 | 0.218292 |
| 60 | 0.757124 | 0.312857 | 0.73426 | 0.254864 |
| 70 | 0.816838 | 0.305162 | 0.755268 | 0.283034 |
| 80 | 1.100287 | 0.383678 | 0.813799 | 0.248629 |

pbfs speed up

| threads | road_usa | delaunay_n24 | hugebubbles-00020 | rgg_n_2_22_s0 |
|---|---|---|---|---|
| 1 | 1.02 | 0.99 | 1.03 | 0.97 |
| 5 | 3.23 | 2.52 | 3.83 | 3.20 |
| 10 | 4.38 | 4.00 | 5.98 | 5.10 |
| 20 | 3.55 | 5.56 | 6.63 | 5.71 |
| 30 | 3.27 | 5.28 | 6.21 | 4.51 |
| 40 | 2.79 | 4.99 | 5.94 | 4.76 |
| 50 | 2.56 | 4.97 | 5.77 | 4.50 |
| 60 | 2.42 | 4.58 | 5.23 | 3.86 |
| 70 | 2.24 | 4.70 | 5.08 | 3.47 |
| 80 | 1.67 | 3.74 | 4.72 | 3.95 |




---------- AFBS ----------

ABFS algorithm starts with the sequential execution of the BFS algorithm. Then, at layer 200, it switches to a parallel version of the BFS algorithm. But now the found vertices are copied back to the shared array S at every $k^{th}$ iteration.

In the parallel part of the algorithm, the threads are allocated a part of the vertices from the last round of the sequential algorithm. Each thread maintains its own local queue of vertices to be processed using an array called loc_S. Any discovered vertices in the parallel part should remain with the thread that discovered them. Each thread then performs a BFS starting from the vertices in its local queue, enqueuing the discovered vertices in its local queue. Once a thread has processed all the vertices in its local queue, it reports the number of vertices discovered to a shared array called T.

The algorithm then synchronizes the threads using a barrier and aggregates the number of vertices discovered by each thread in the shared array T. Then, the algorithm determines the size of the new queue S for the next iteration by summing up the elements in the shared array T. It also

computes the prefix sum of the elements in the shared array T to determine the starting position of each thread's vertices in the new queue S.

Finally, the algorithm updates the new queue S with the vertices discovered by each thread in its local queue, using the starting position computed earlier. If the current iteration count is a multiple of k, the algorithm updates the global queue S with the vertices in the local queue. The algorithm then repeats the parallel BFS until all vertices have been explored.

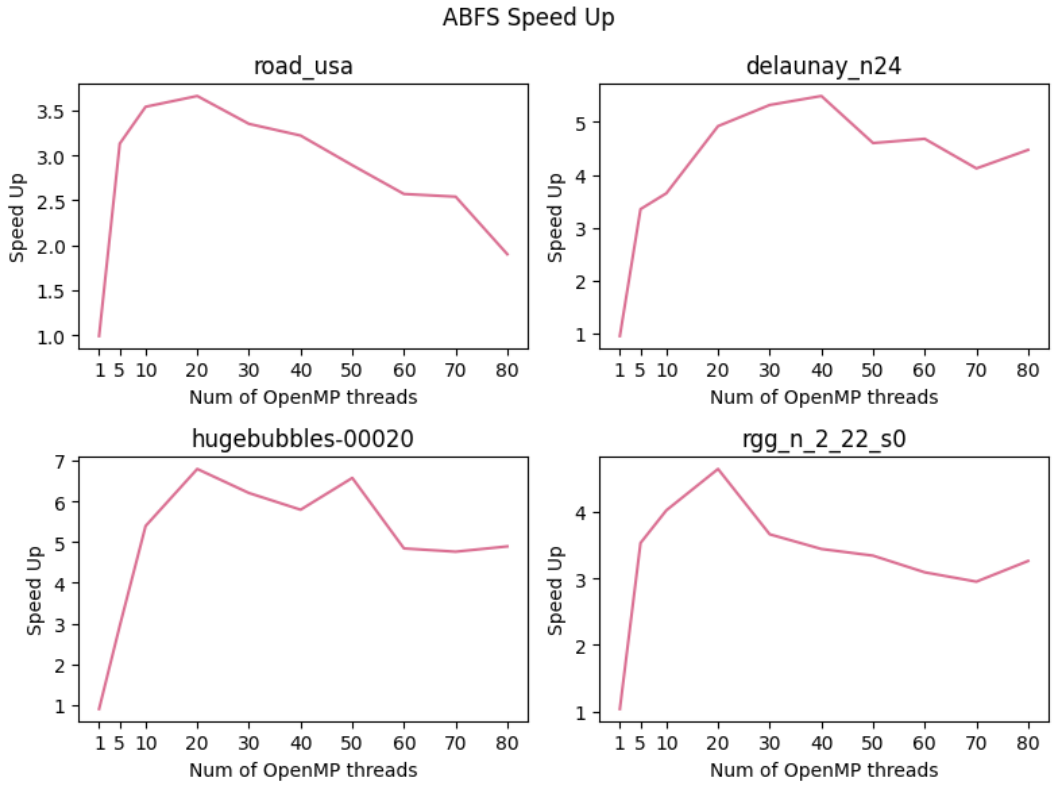
Unfortunately, I haven't fully figured out how to switch between the loc_S and S arrays so that everything works as it should. The commented code demonstrates my attempts:( However, I leave the working code, which switches from the sequential version to the parallel version at layer 200. This is not what I needed to show in the report, but still..

abfs results

| threads | road_usa | delaunay_n24 | hugebubbles-00020 | rgg_n_2_22_s0 |
|---|---|---|---|---|
| 1 | 1.852659 | 1.504761 | 4.262383 | 0.943271 |
| 5 | 0.585652 | 0.427975 | 1.306336 | 0.278452 |
| 10 | 0.517114 | 0.393186 | 0.712335 | 0.244775 |
| 20 | 0.500424 | 0.291465 | 0.565574 | 0.212068 |
| 30 | 0.547044 | 0.269556 | 0.618553 | 0.268836 |
| 40 | 0.568952 | 0.261019 | 0.663033 | 0.285801 |
| 50 | 0.633908 | 0.311635 | 0.584268 | 0.294588 |
| 60 | 0.714053 | 0.3061 | 0.792147 | 0.318073 |
| 70 | 0.721406 | 0.34758 | 0.805597 | 0.333501 |
| 80 | 0.963584 | 0.320468 | 0.785321 | 0.301796 |

abfs speed up

| threads | road_usa | delaunay_n24 | hugebubbles-00020 | rgg_n_2_22_s0 |
|---|---|---|---|---|
| 1 | 0.99 | 0.95 | 0.90 | 1.04 |
| 5 | 3.13 | 3.35 | 2.94 | 3.53 |
| 10 | 3.54 | 3.65 | 5.39 | 4.02 |
| 20 | 3.66 | 4.92 | 6.79 | 4.64 |
| 30 | 3.35 | 5.32 | 6.20 | 3.66 |
| 40 | 3.22 | 5.49 | 5.79 | 3.44 |
| 50 | 2.89 | 4.60 | 6.57 | 3.34 |
| 60 | 2.57 | 4.68 | 4.84 | 3.09 |
| 70 | 2.54 | 4.12 | 4.76 | 2.95 |
| 80 | 1.90 | 4.47 | 4.89 | 3.26 |


ABFS Speed Up
road_usa
delaunay_n24
hugebubbles-00020
rgg_n_2_22_s0
Speed Up
Num of OpenMP threads
1 5 10 20 30 40 50 60 70 80