1. Parallel Efficiency Analysis:

Tested on the 1000 nodes, 3000 edges sample test case given.
For 1 process :

```
[cs3401.13@rce problem3_2]$ sbatch run_2node_mpi.sbatch large_file.txt 1
Submitted batch job 56431
[cs3401.13@rce problem3_2]$ ls
compile.sh  large_file.txt  mpi_56431.err  mpi_56431.out  mpi_graph  mpi_gr
[cs3401.13@rce problem3_2]$ cat mpi_56431.out
========================================
SLURM_JOB_ID = 56431
SLURM_NODELIST = node[06-07]
SLURM_JOB_GPUS =
========================================
Process 0 running on node06.local
Total Execution Time: 8.0799e-05 seconds
0 0
```

For 2 processes :

```
[cs3401.13@rce problem3_2]$ sbatch run_2node_mpi.sbatch large_file.txt 2
Submitted batch job 56432
[cs3401.13@rce problem3_2]$ ls
compile.sh  large_file.txt  mpi_56431.err  mpi_56431.out  mpi_56432.err  mp
[cs3401.13@rce problem3_2]$ cat mpi_56432.out
========================================
SLURM_JOB_ID = 56432
SLURM_NODELIST = node[06-07]
SLURM_JOB_GPUS =
========================================
Process 0 running on node06.local
Process 1 running on node06.local
Total Execution Time: 0.000188627 seconds
0 0
1 0
2 0
3 0
4 0
```

For 4 processes :

```
[cs3401.13@rce problem3_2]$ sbatch run_2node_mpi.sbatch large_file.txt 4
Submitted batch job 56433
[cs3401.13@rce problem3_2]$ ls
compile.sh  large_file.txt  mpi_56431.err  mpi_56431.out  mpi_56432.err  mp
[cs3401.13@rce problem3_2]$ cat mpi_56433.out
========================================
SLURM_JOB_ID = 56433
SLURM_NODELIST = node[06-07]
SLURM_JOB_GPUS =
========================================
Process 1 running on node06.local
Process 0 running on node06.local
Process 3 running on node07.local
Process 2 running on node07.local
Total Execution Time: 0.0018343 seconds
0 0
1 0
2 0
```

Here, we observe that the execution time is increasing instead of decreasing.
There are three major reasons:
1. The Problem Size Is Extremely Small :
For 1 process, it was just 80 microseconds.
MPI startup + synchronization overhead alone is often:

- 50–500 microseconds

So what we are measuring is mostly : MPI initialization, MPI_Allreduce latency, Synchronization overhead. But not the actual computation.

When the workload is too small:

$T_{communication} >>> T_{computation}$

So adding more processes increases overhead.

2. MPI_Allreduce on Full n Array Every Iteration :

This is expensive:

MPI_Allreduce(MPI_IN_PLACE, new_comp.data(), n, MPI_INT, MPI_MIN, MPI_COMM_WORLD);

Cost per iteration: O(nlogP)

When n is small => latency dominates
When P increases => communication cost increases

Speedup and Efficiency :

S(2)=0.00008/0.000188 ≈ 0.42
S(4)=0.00008/0.0018343 ≈ 0.043

Parallel efficiency:

E(2)=0.42/2=0.21=21%
E(4)=0.043/4 ≈ 1%

Estimating maximum graph size processable :

Theoritically, since each iteration scans all edges: T is I(m+nlogP)

If time roughly scales linearly with m: T is proportional to m

For 3000 edges, it took about 0.00008 seconds => in 600 seconds, we can do – (600/0.00008)*3000 = **22500000000 edges = 22.5 B edges**

2. Communication Profiling:

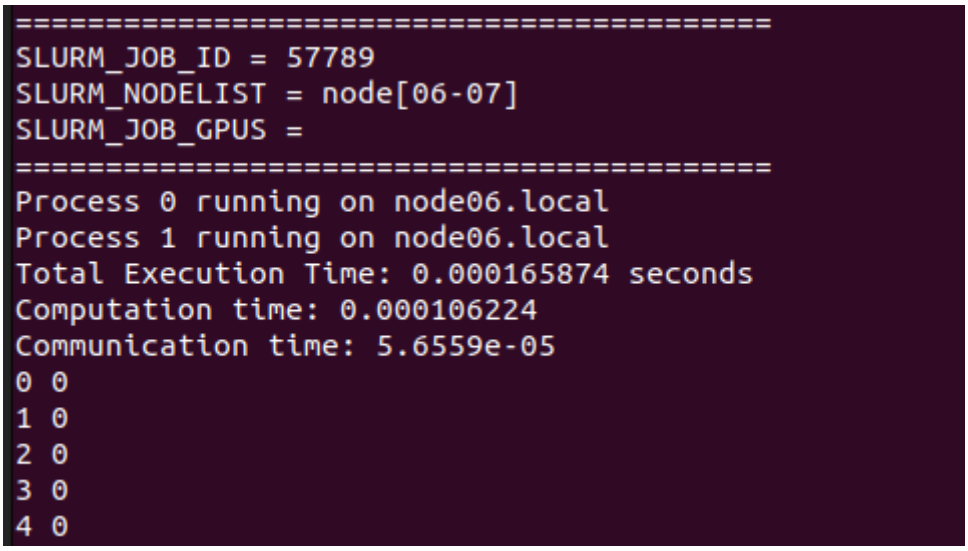
```
cat: mpi_56436.er: No such file or directory
[cs3401.13@rce problem3_2]$ cat mpi_56436.err
[cs3401.13@rce problem3_2]$ cat mpi_56436.out
========================================
SLURM_JOB_ID = 56436
SLURM_NODELIST = node[06-07]
SLURM_JOB_GPUS =
========================================
Process 0 running on node06.local
Process 1 running on node06.local
Process 2 running on node07.local
Process 3 running on node07.local
Total Execution Time: 0.0020443 seconds
Computation time: 0.000109826
Communication time: 0.00313823
0 0
1 0
2 0
```

Communication-to-Computation Ratio = 0.00313823/0.000109826 = **2.85745634**

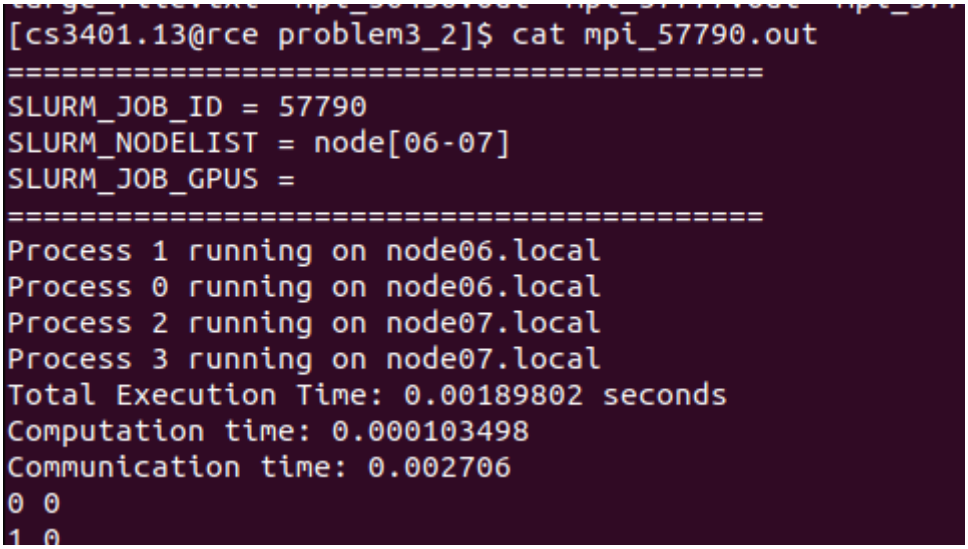
For 1 process :

```
[cs3401.13@rce problem3_2]$ cat mpi_57777.out
========================================
SLURM_JOB_ID = 57777
SLURM_NODELIST = node[06-07]
SLURM_JOB_GPUS =
========================================
Process 0 running on node06.local
Total Execution Time: 8.312e-05 seconds
Computation time: 7.561e-05
Communication time: 5.876e-06
0 0
1 0
2 0
3 0
4 0
```

For 2 processes :



For 4 processes :



As expected in parallel effeciency analysis, the communication time communication cost increases significantly.
From 1 process to 2 processes -> increased by 0.000056559/0.00000587 = 9.635264055x = 10x times
From 2 processes to 4 processes -> increased by 0.002706/0.000056559 = 47.843844481x = 48x times
=> From 1 process to 4 processes -> increased by 480x times.

If we look at the actual computation time,
From 1 to 2 processes -> S(2) = 0.00007561/0.000106224 = 0.71179771

From 2 to 4 processes -> 0.000103498/0.000106224 = 0.97433725

The main message is MPI_Allreduce(... n integers ...)

Message size per iteration: n×sizeof(int)
From logs, message size per Allreduce: 0.0038147 MB

3. Computational Complexity:

FLOPs Analysis:
Per edge:
1 min() operation
2 comparison operations
2 assignment operations

~5 operations per edge.

Per iteration: operations = 5m operations

If I iterations: TotalFLOPs = 5mI

Plus: n comparisons for local_changed = nI

So total: O(I(m+n))

Communication Complexity:

Each iteration:O(nlogP)

So total:O(I(m+nlogP))

Theoretical Scaling :

If:

- m >> n
- I constant

Then: T = O(m)