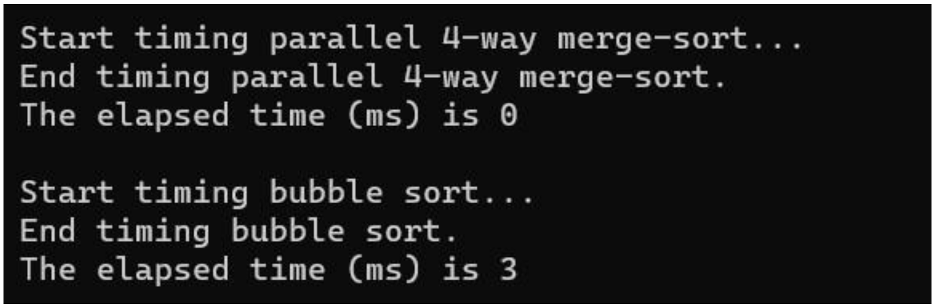
Benedictus Alvian Sofjan

COMP3230 Assignment 1

Q2.3 Performance Evaluation

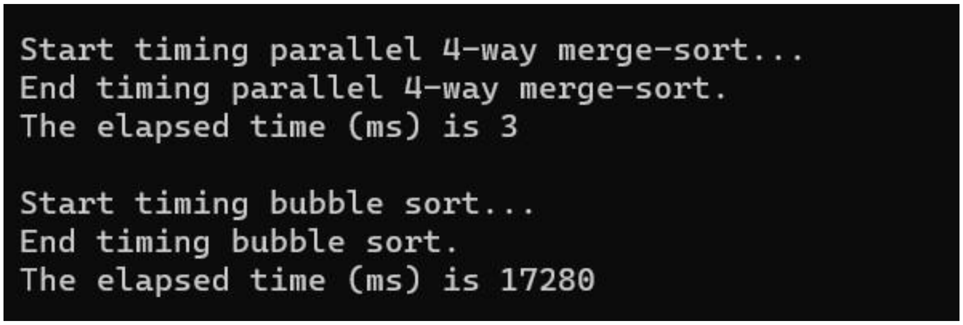
n=1, max_num= 256



```
Start timing parallel 4-way merge-sort...
End timing parallel 4-way merge-sort.
The elapsed time (ms) is 0

Start timing bubble sort...
End timing bubble sort.
The elapsed time (ms) is 3
```

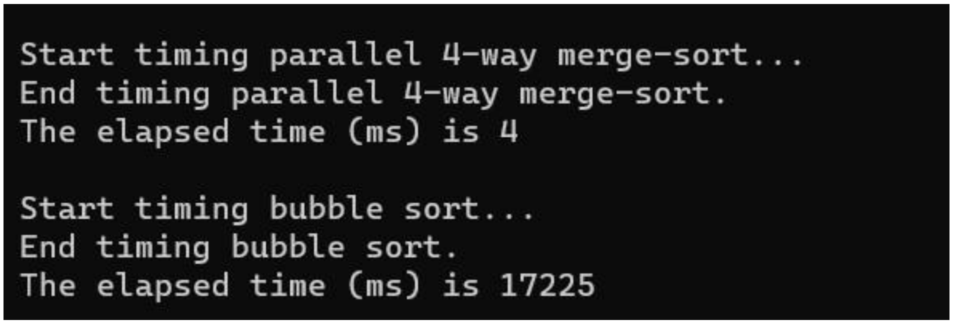
n=2, max_num=4096



```
Start timing parallel 4-way merge-sort...
End timing parallel 4-way merge-sort.
The elapsed time (ms) is 3

Start timing bubble sort...
End timing bubble sort.
The elapsed time (ms) is 17280
```

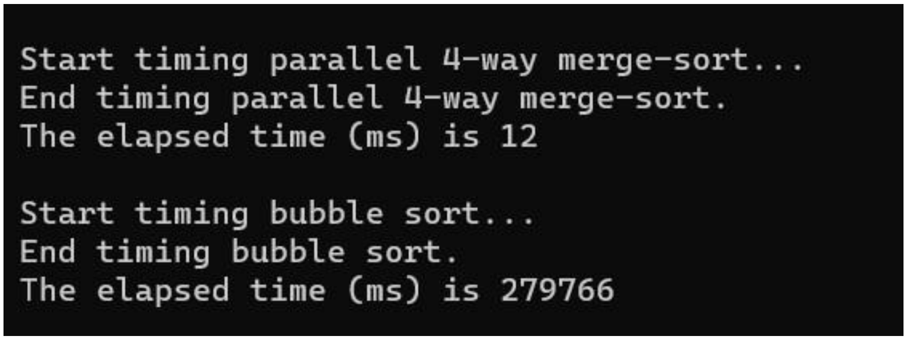
n=3, max_num=1024



```
Start timing parallel 4-way merge-sort...
End timing parallel 4-way merge-sort.
The elapsed time (ms) is 4

Start timing bubble sort...
End timing bubble sort.
The elapsed time (ms) is 17225
```

Proper size chosen: n=4, max_num=1024



```
Start timing parallel 4-way merge-sort...
End timing parallel 4-way merge-sort.
The elapsed time (ms) is 12

Start timing bubble sort...
End timing bubble sort.
The elapsed time (ms) is 279766
```

Speedup: 279766/12 = 16842.125 times ~ 23000x faster

Discussion and Analyses

Parallel 4-way merge-sort is always faster than bubble sort given its logarithm design O(n log n) compared to bubble sort's $O(n^2)$ and its ability to run on multiple threads concurrently.

n=2, max_num=4096 and n=3, max_num=1024 sorts the same amount of integer, but parallel 4-way merge-sort elapsed time is similar due to the limit of the number of cores. Likewise, bubble sort elapsed time is similar because it runs on a single core.