202211032 김수호 – HW2 Report

1. Implementation Results

Initially, I modified the given code and ran it 100 times to calculate the average result. When running the original code, I observed a wide range of results from 0.8 to 1.6. Therefore, I adopted the following approach:

Bmp file size	Speed up
128	0.937877
256	0.952160
512	0.947195
768	0.949156
1024	0.968949

The table above shows the speedup values obtained by running the original code.

Bmp file size	Speed up
128	0.963364
256	0.968527
512	0.987207
768	1.017271
1024	1.018471

The table above shows the speedup values obtained by swapping the x and y axes.

Bmp file size	Speed up
128	1.758436
256	2.411985
512	3.002736
768	2.622776
1024	2.282822

The table above shows the speedup values obtained by running 4 threads simultaneously.

Bmp file size	Speed up
128	1.842538
256	3.498382
512	3.623909
768	2.946589
1024	2.443838

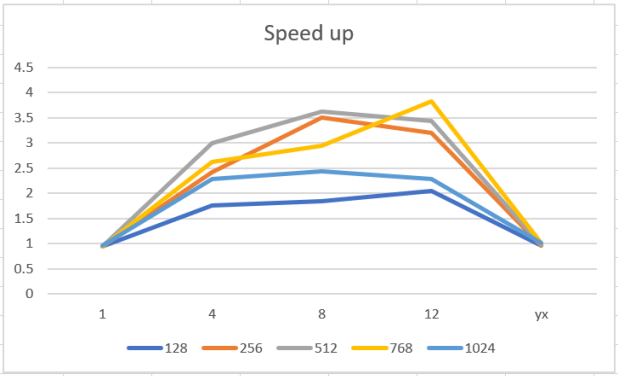
The table above shows the speedup values obtained by running 8 threads simultaneously.

Bmp file size	Speed up
128	2.042761
256	3.192077

512	3.439881
768	3.819564
1024	2.289215

The table above shows the speedup values obtained by running 12 threads simultaneously.

	1	4	8	12	yx
128	0.937877	1.758436	1.842538	2.042761	0.963364
256	0.95216	2.411985	3.498382	3.192077	0.968527
512	0.947195	3.002736	3.623909	3.439881	0.987207
768	0.949156	2.622776	2.946589	3.819564	1.017271
1024	0.968949	2.282822	2.443838	2.289125	1.018471



The graph above includes all cases and represents the results graphically. The x-axis represents the number of threads and the xy transformation, while the y-axis represents the speedup.

2. Analysis of Results and Optimization Approach

First, I considered the efficiency related to the x and y axes, as discussed in class. Assuming that viewing based on the x-axis is 0.25 and viewing based on the y-axis is 1, I thought the original code was less efficient because it views based on the y-axis. Therefore, I swapped the x and y axes, and the results are shown in the table above. Although there was some improvement, swapping the x and y axes did not significantly enhance efficiency.

Next, I attempted to improve efficiency by using threads, as discussed in class. Assuming the original code runs on a single core, I hypothesized that increasing the number of threads would proportionally increase efficiency. The table above shows that running multiple processes simultaneously with threads improved efficiency by 1.7 to 3.8 times compared to the original code. However, while efficiency significantly increased with 4 or 8 threads, it did not increase as much, or even decreased, with 12 threads.

In such cases, I considered that the overhead of creating and managing too many threads, as well as contention errors caused by multiple threads, might be the reasons for the decreased efficiency. Thus, I thought it is important to consider the optimal number of threads based on the size of the BMP file.