DORUK DOGANAY – DSD20

PERCOLATION ANALYSIS:

1. Doubling N increased the runtime for all cases. Doubling N causes runtimes to be multiplied with the corresponding numbers (approximately these numbers) found below:
   PercolationDFS: x 16.
   PercolationDFSFast: x 4.
   PercolationUF (QuickFind) x 16.
   PercolationUF (QuickUWPC): x 4.

2. Doubling T increased the runtime for all cases. Doubling T caused the runtimes to be multiplied with the following numbers (approximately) for each case:
   PercolationDFS: x 2.
   PercolationDFSFast: x 2.
   PercolationUF (QuickFind) x 2.
   PercolationUF (QuickUWPC): x 2.

3. PercolationDFS: $O(N^4T)$
   PercolationDFSFast: $O(N^2T)$
   PercolationUF (QuickFind): $O(N^4T)$
   PercolationUF(QuickUWPC): $O(N^2T)$

4. Seconds in a day: 86400. For all the following runtimes, I ran the code with N=200 and T=100. Thus, make calculations dividing this number by each runtime:
   PercolationDFS: 1100
   PercolationDFSFast: 67000
   PercolationUF (QuickFind): 1220
   PercolationUF (QuickUWPC): 123000

5. Both PercolationDFS and PercolationDFSFast uses approximately 4N^2 bytes of memory given that they store an integer grid of size N^2 and an integer takes up 4 bytes of space.
   Because a Boolean uses a single byte of space, the Boolean grid created in PercolationUF with QuickFind or QuickUWPC should use about N^2 bytes. These terms are going to dominate the calculations.