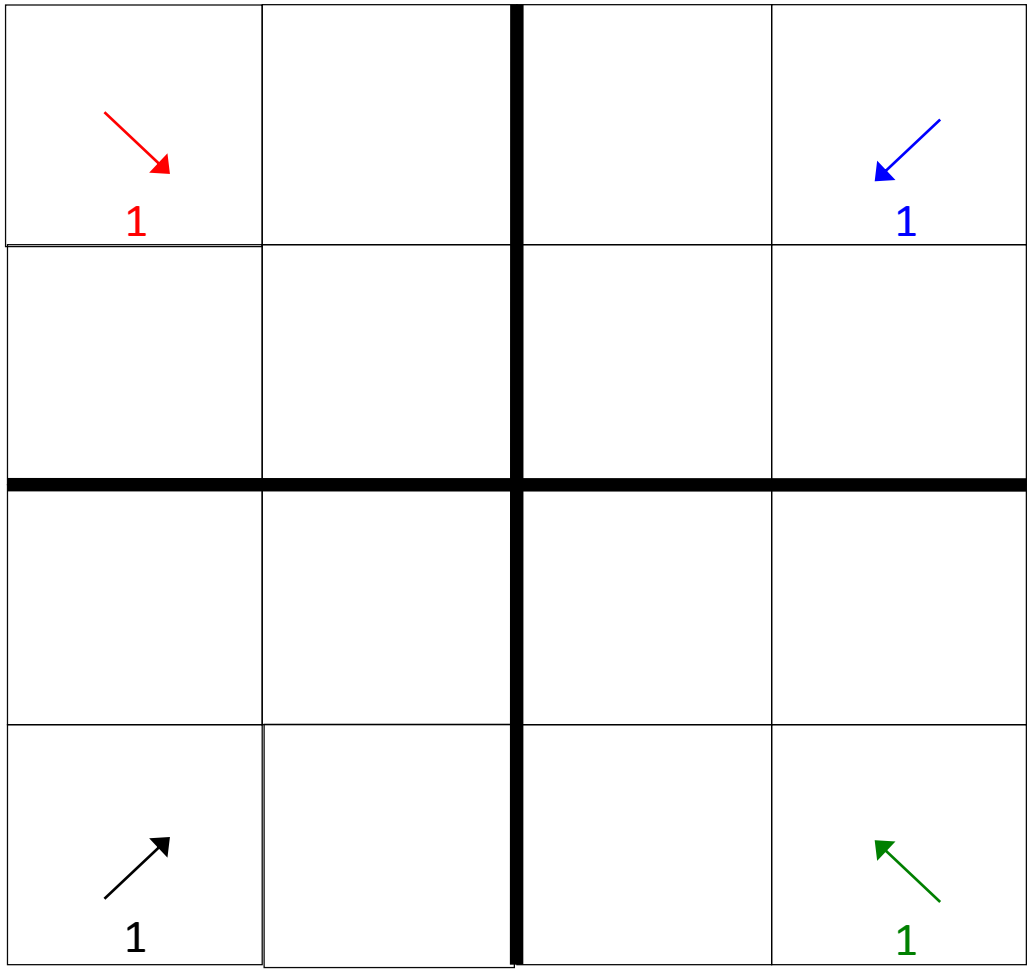
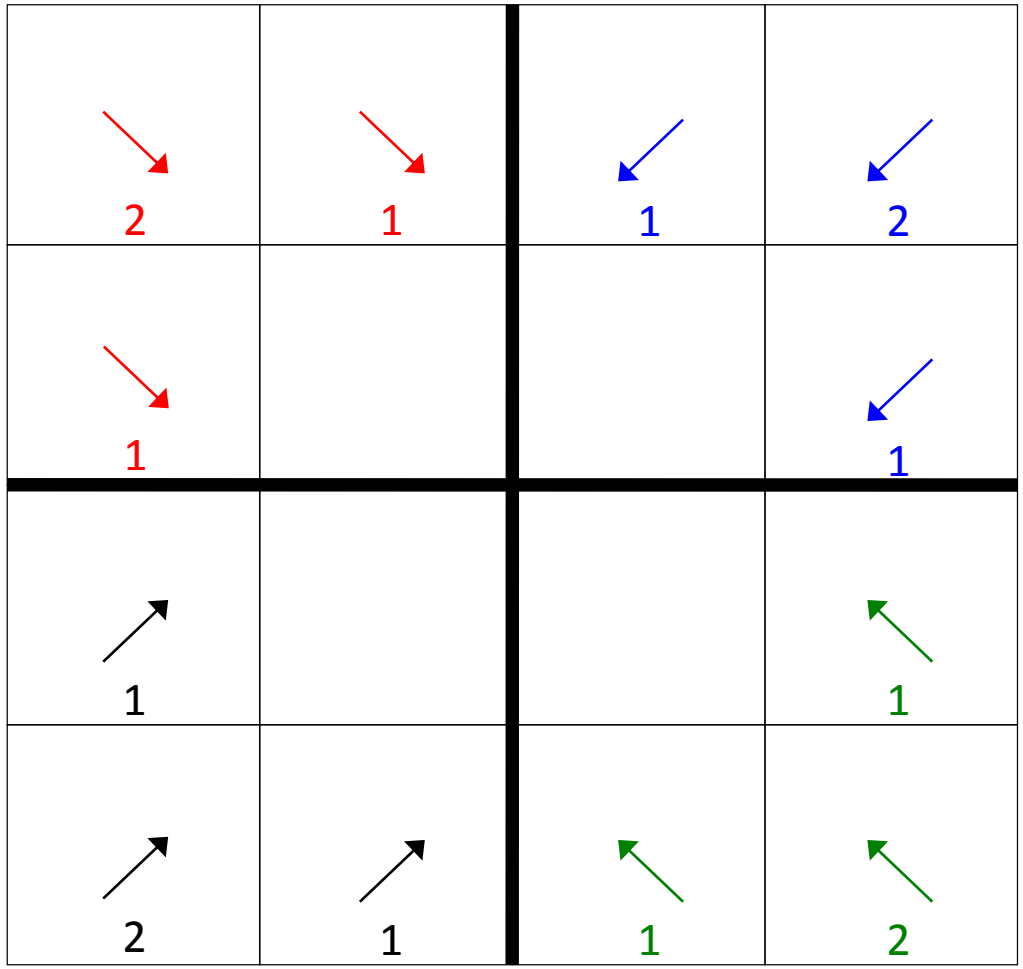


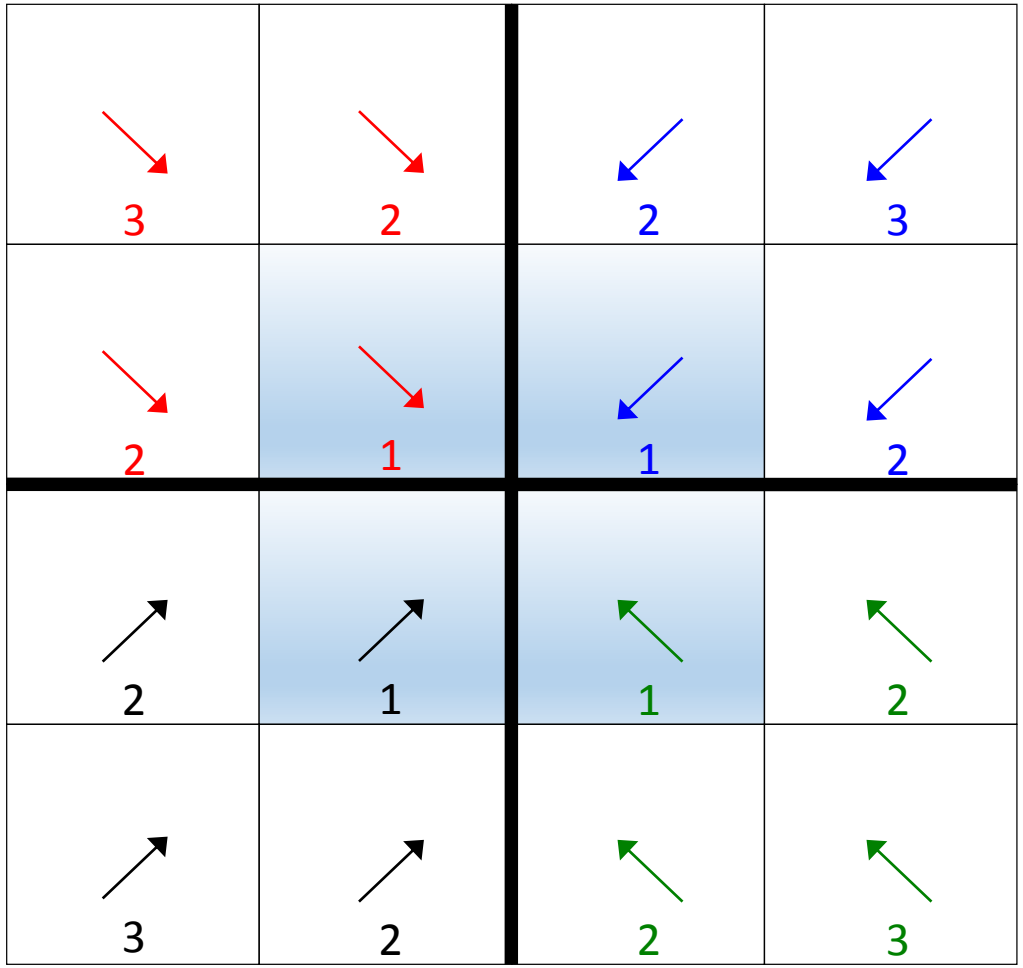
$P_x = 4$, $P_y = 4$
$N_g = 3$, $N_m = 1$

$N_{fill} = 2$

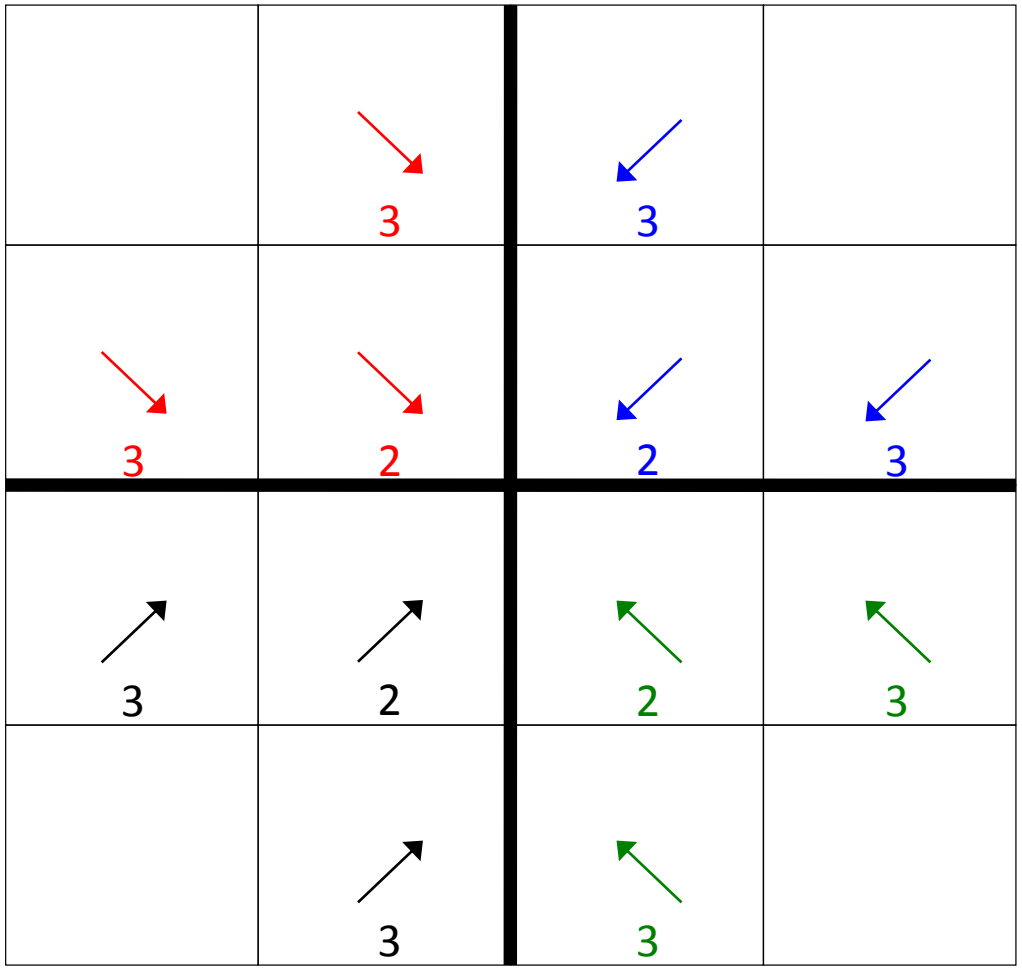
A 2D domain decomposed into 4 processors in x and 4 processors in y. The bold lines represent the quadrants we've split this into. We partition the problem into M = 1, G = 3, $A_x = 1$, $A_y = 1$, $A_m = 1$, $A_g = 1$. The number of tasks is N_{tasks} = 12.



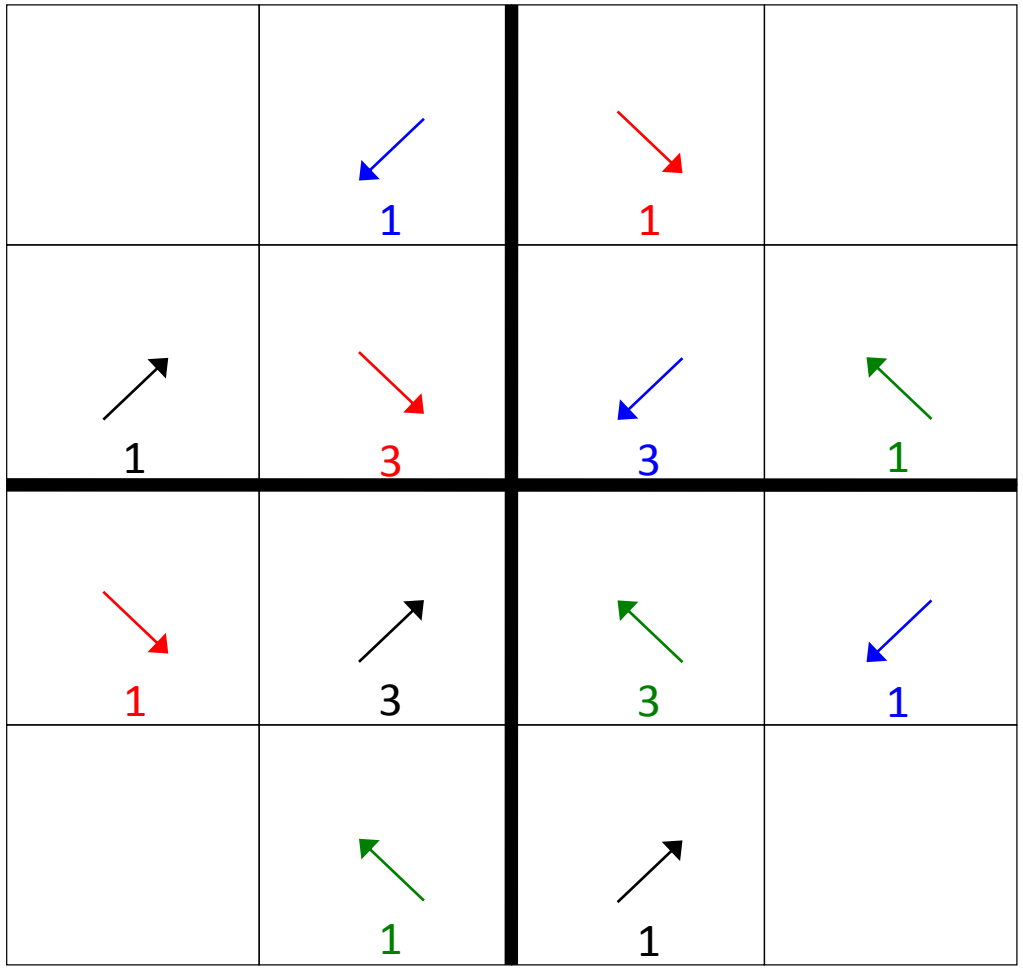
Stage 2: The corner processors have completed the first task and communicated to downstream dependents. The corner processors now compute the second task, and the dependents compute the first task.



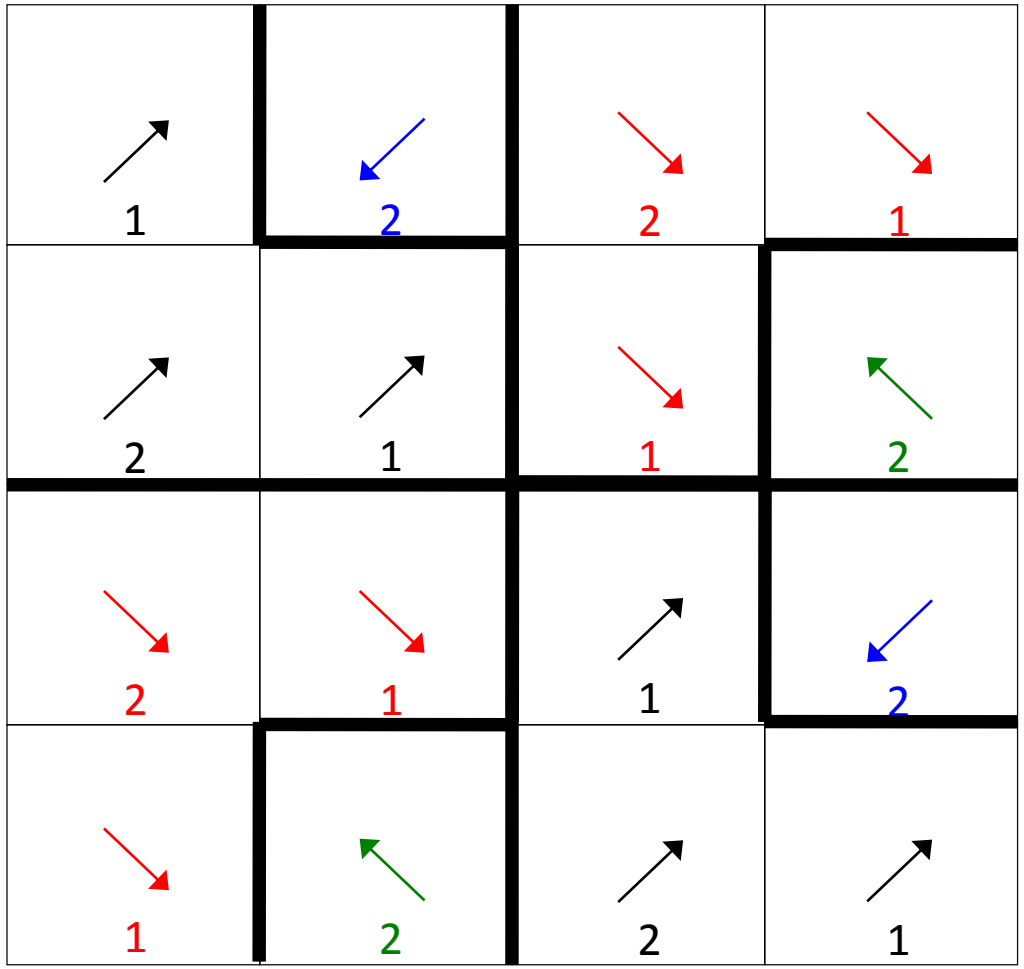
Stage 3: The central processors have begun computation, verifying that the predicted N_{fill} value of 2 was correct. These processors will no longer be idle until they are done computing all tasks, as this is the optimal scheduling algorithm. We can also see that tasks are being queued on processors. For instance in the right half the domain, Task Green 1 has been communicated from processor (4,2) to processor (4,3), and Task Blue 1 has been communicated from processor (4,3) to processor (4,2). However, processor (4,3) will solve tasks originating from quadrant (++) before it solves tasks originating from quadrant (+-) (according to depth of graph algorithm). Therefore in Stage 3 processor (4,3) solves Task Blue 2. Similar behavior occurs across the domain.



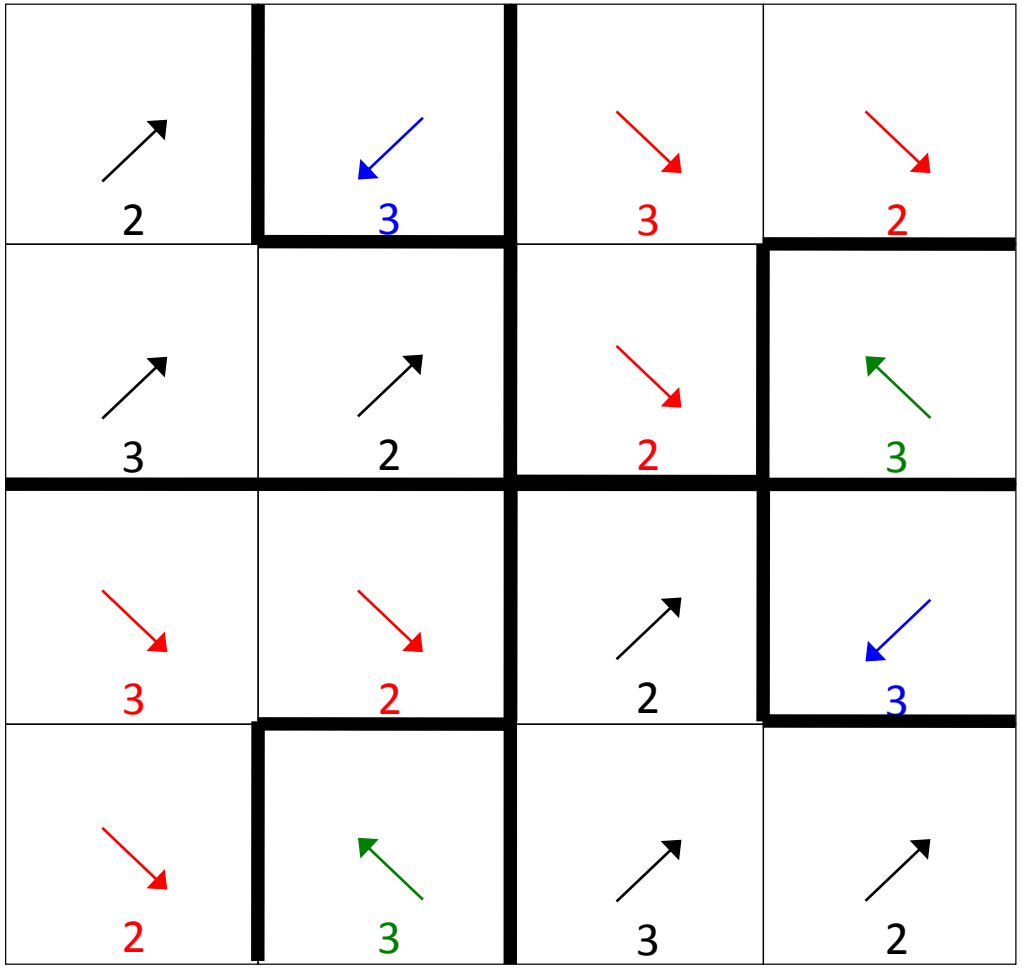
Stage 4



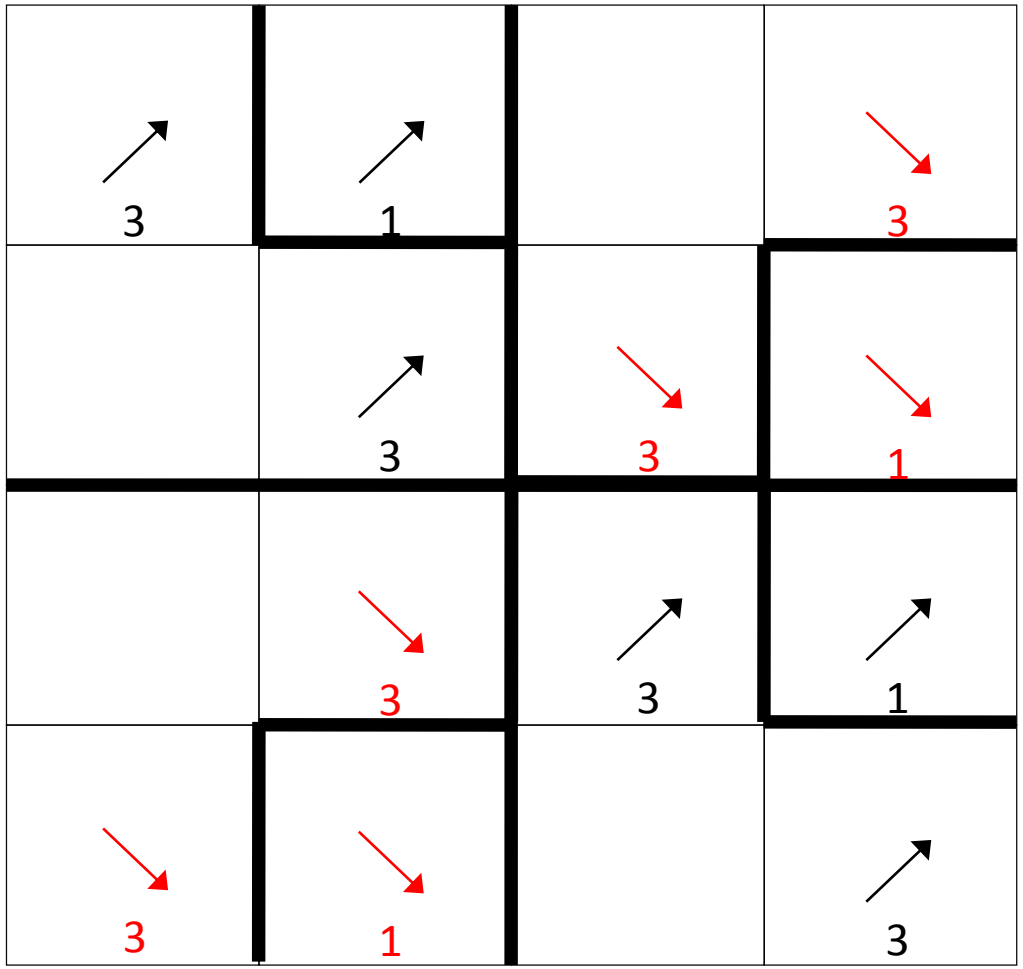
Stage 5



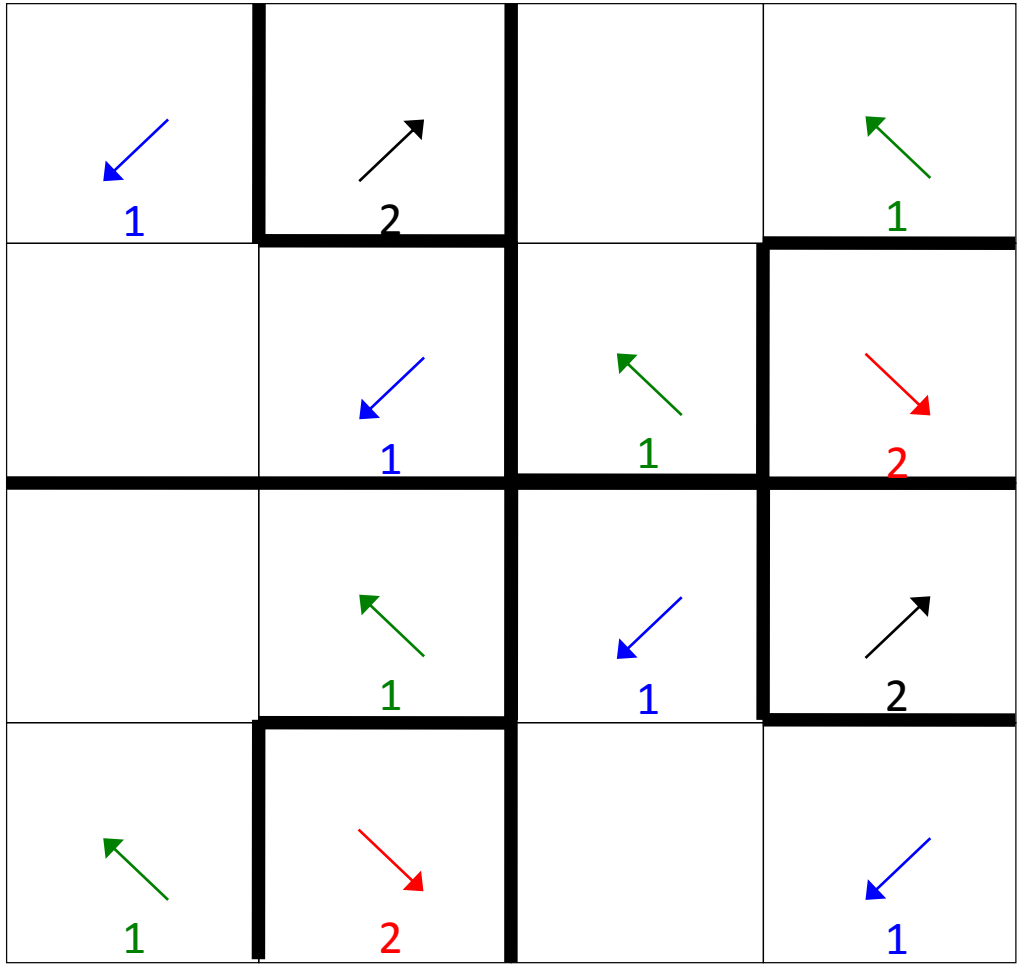
Stage 6: Some conflicting tasks have the same D (because they all exist in the same quadrant), so tiebreakers based on $Omega_x$ are used. In the lower quadrant, for example, we can see that after stage five, both the green and red directions were ready to communicate to their neighboring processors. However, since the red direction posses and $Omega_x > 0$, it has priority, so this task is performed first. The same happens in other quadrants experiencing similar conflicts.



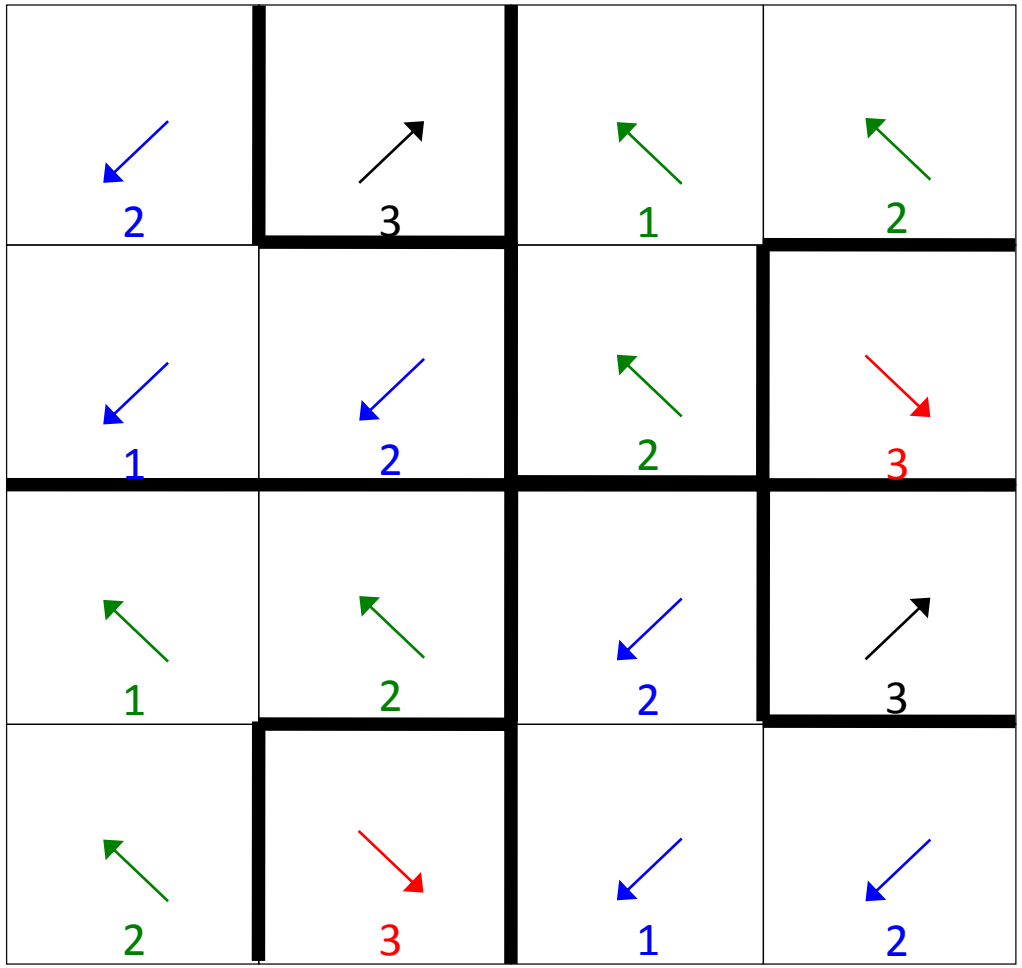
Stage 7



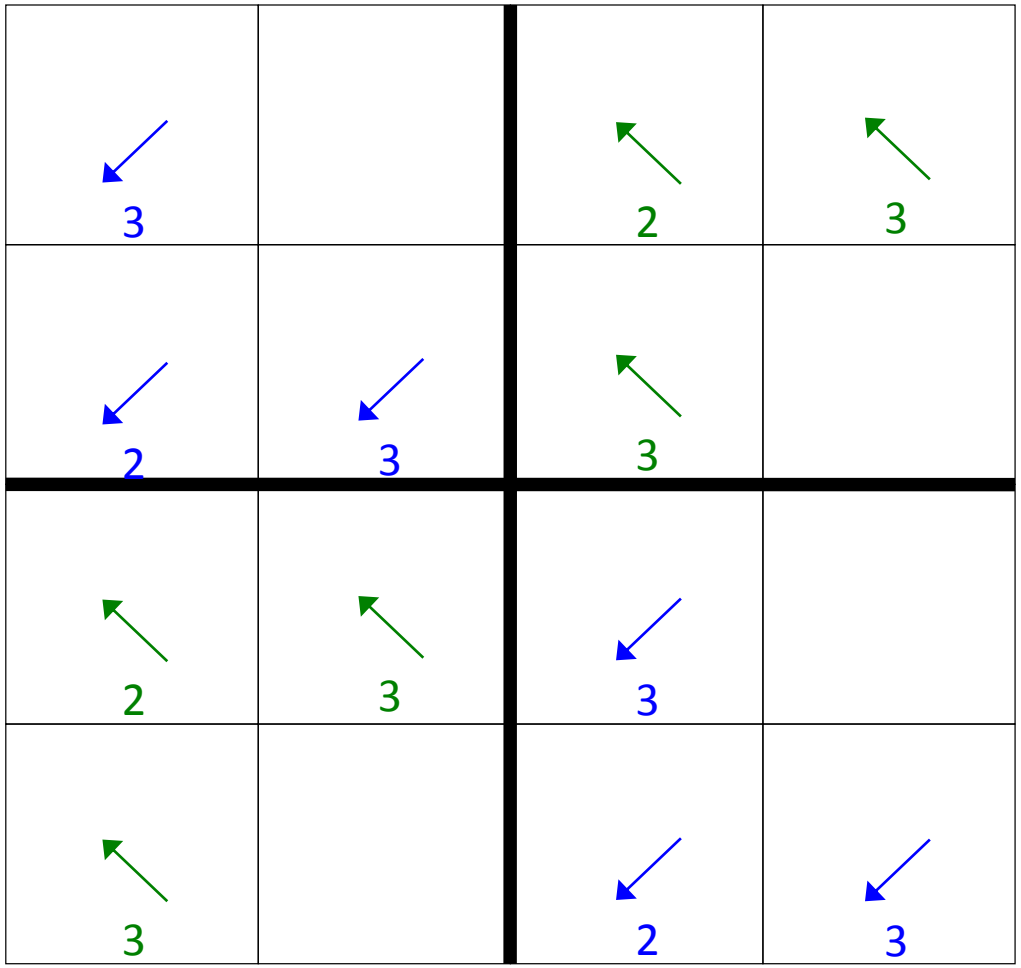
Stage 8



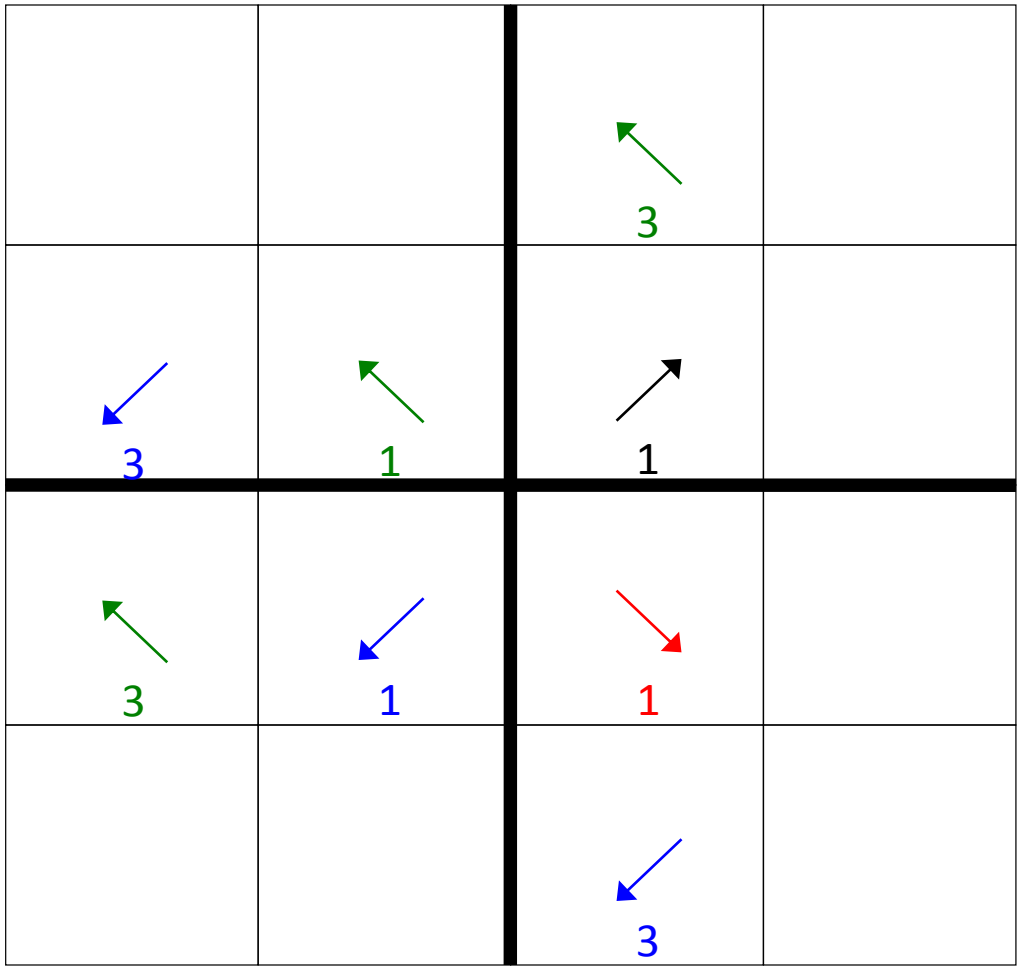
Stage 9: The secondary tasks that were queued on processors now begin executing.



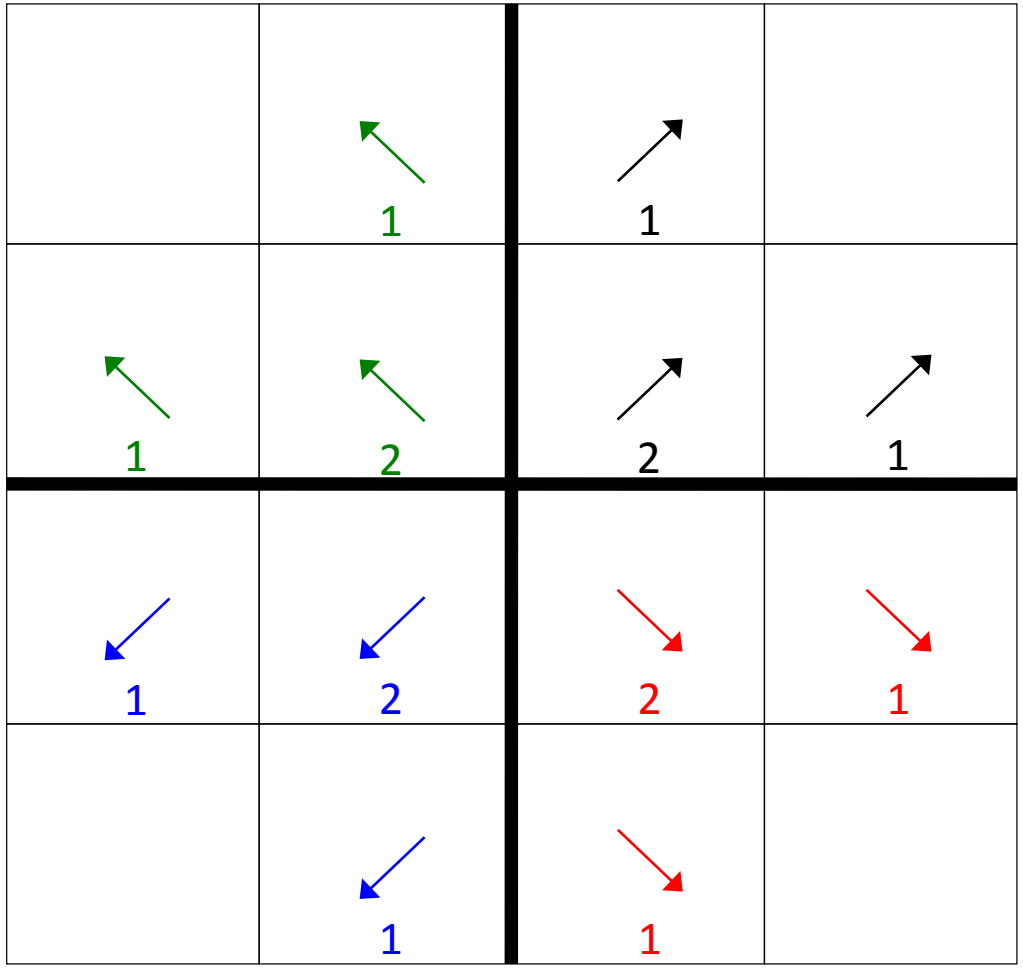
Stage 10: The secondary tasks that were queued on processors now begin executing.



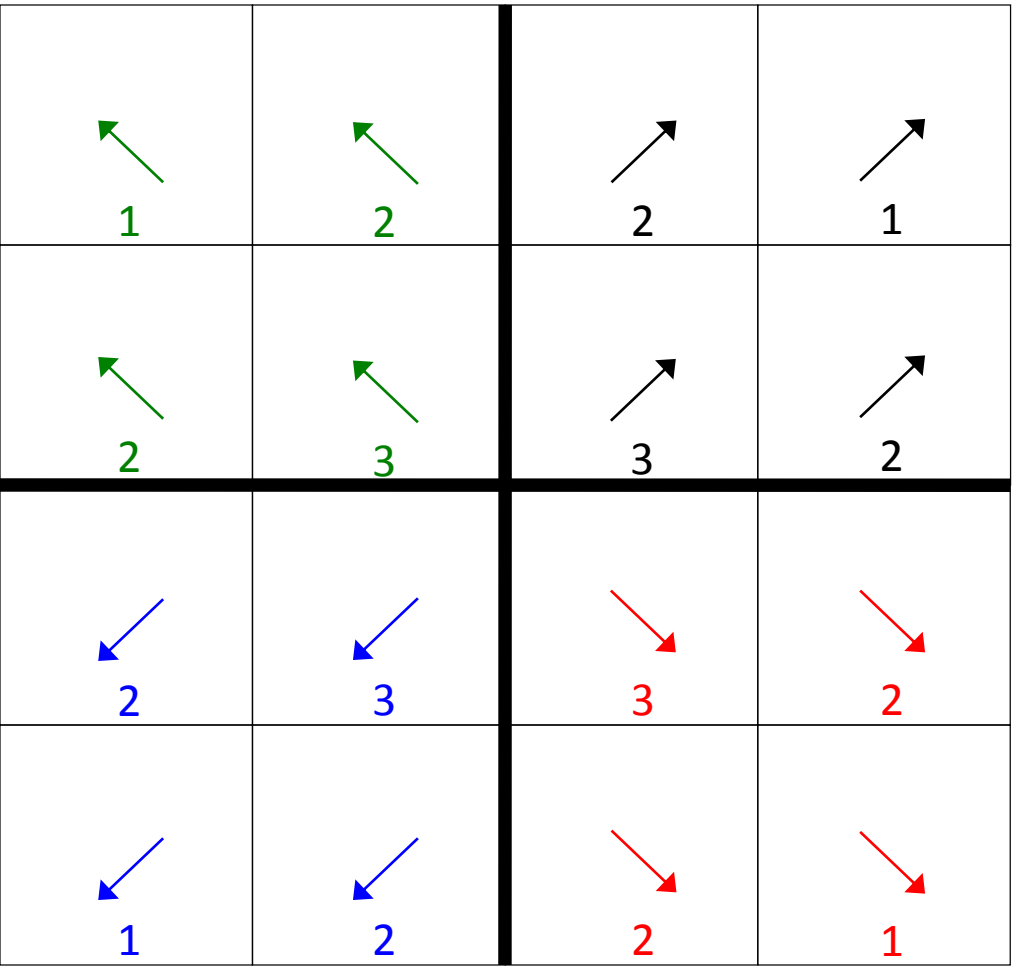
Stage 11



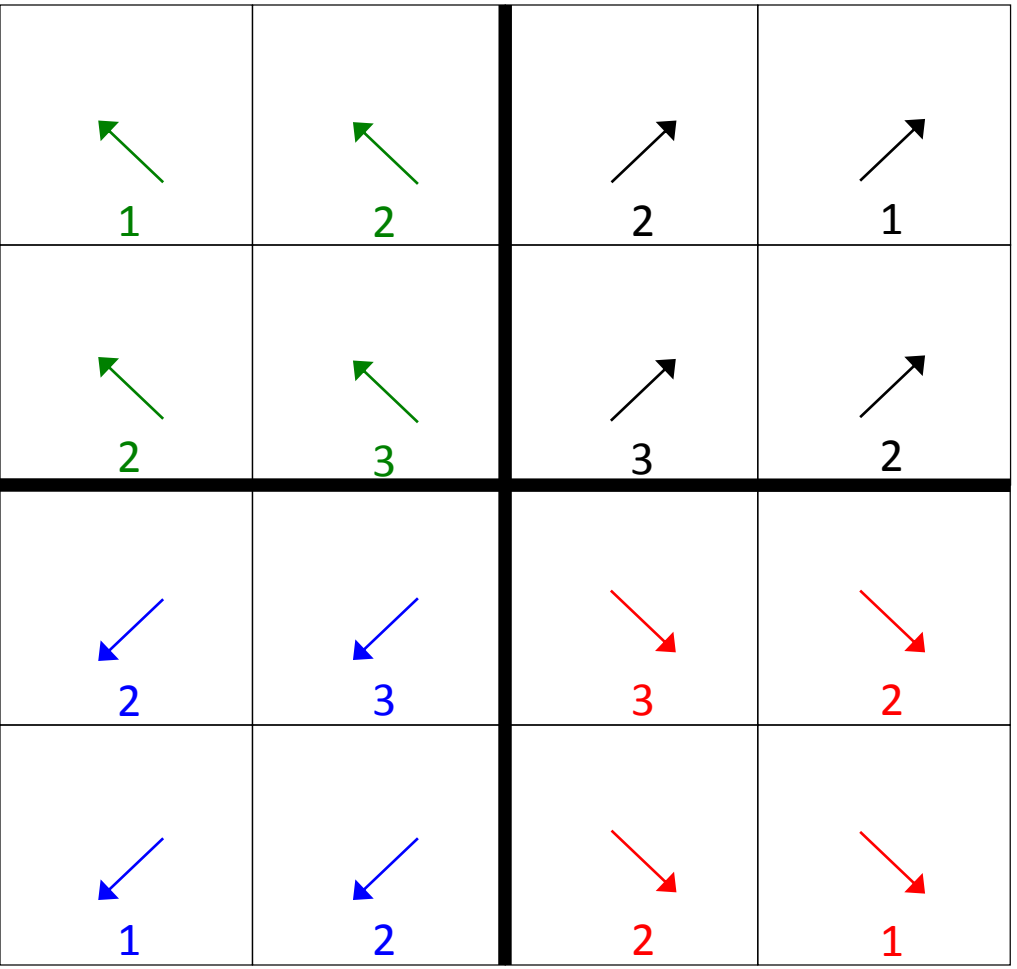
Stage 12



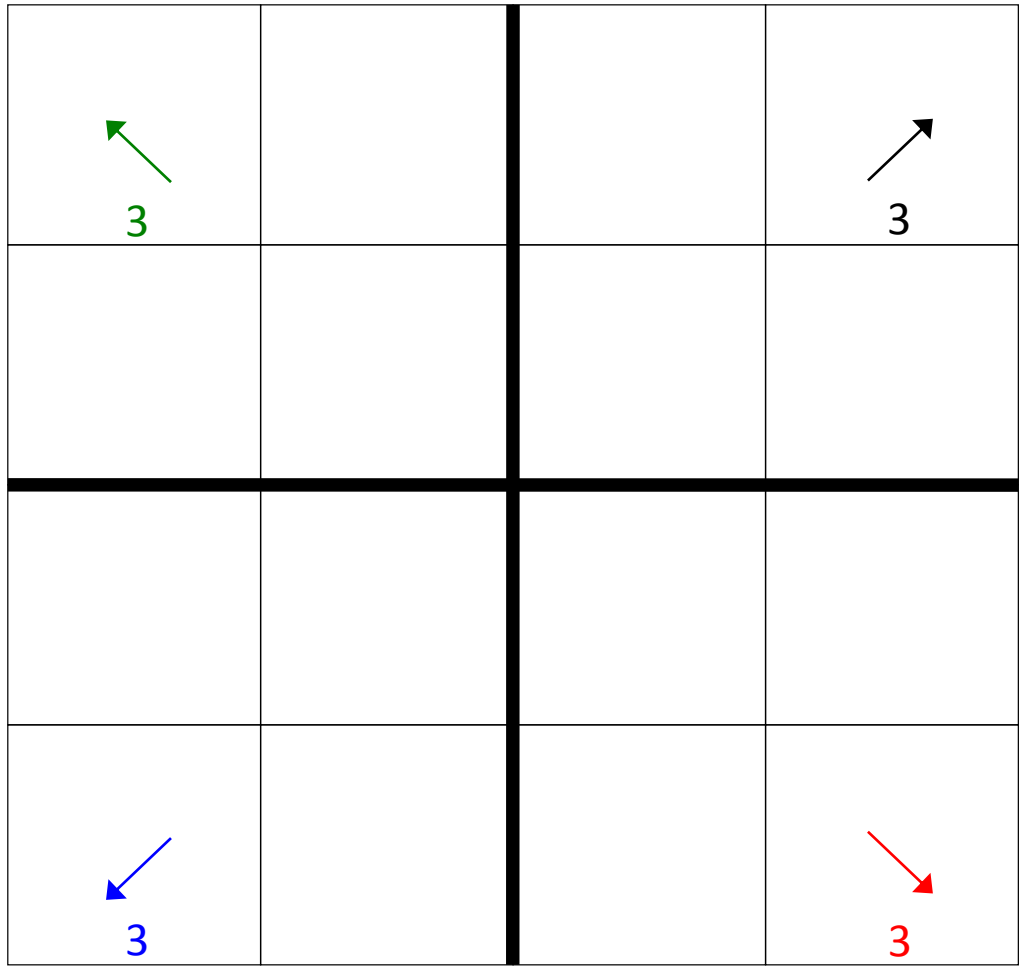
Stage 13



Stage 14



Stage 15



Stage 16: Verifies $N_{stages} = N_{fill} + N_{tasks} = 16$