1. We have compleed the project. Our code runs successfully all example inputs.
2. We choose to split the processors checkered. checkered implementation runs faster and challenging that we enjoyed while coding.

-We create NxN 2D array that contains all board, and send parts of maps to the each processors.

-Coordinates of towers are read from input and pushed the array. The algorithm finds each tower’s processor and store it to the array and sends it to the corresponding processor at the start of a wave. Now everything needed is sends from manager to the processors.

-Each processor first has to take its row, column, wave round and part of the map that it is responsible for. After that for loop iterates number of wave times to simulate each wave. It receives placement of towers at this wave. If the coordinate is empty it replaces it to the tower.

- Each processor executes 8 rounds. Each tower should know how much damage it will take. To do this, a processors should know its surrounding processors maps to control towers’s around. To do this,

* even rows send bottom side to below odd and wait response from it.
* odd rows send bottom side to below even and wait response from it.
* odd columns send their right sides to right and wait response from it
* even columns send their right sides to right and wait response from it

But processors do not have cross neighbors. We get the cross neighbors’ cells from its right left top and bottom neighbors since they have that information from former send and receives.

After finding all neighbors of processors, we create a array that contains cell and its side neighbors to check how much damage a tower get, and decrease the health of tower. This procedure executes for each round.

At the end of the waves, we concatenate the part maps to create the whole map.

1. Complexity is approximately equals to the communication complexity. So, it is convenient to count send operations. At manager processor

for i in range(1, P+1):

row = (i-1) // sqrt\_p

column = (i-1) % sqrt\_p

part\_map = [[column for column in row[column \* sqrt\_cell:(

column+1)\*sqrt\_cell]] for row in game\_map[row \* sqrt\_cell:(row+1)\*sqrt\_cell]]

comm.send({"row": row, "column": column, "wave": wave,

"part\_map": part\_map}, dest=i, tag=0)

for i in range(wave):

wave\_count = i

process\_towers = [[] for x in range(size)]

o\_tower = o\_s[i]

plus\_towers = plus\_s[i]

for o in o\_tower:

process\_towers[find\_myself(

o["col"], o["row"], sqrt\_cell)-1].append(o)

for plus in plus\_towers:

process\_towers[find\_myself(

plus["col"], plus["row"], sqrt\_cell)-1].append(plus)

for ad1 in range(1, P+1):

comm.send(process\_towers[ad1-1], dest=ad1, tag=22)

this two send operations are count and found

(P)+(P)\*wave

For each worker processor at each wave 8 round is executed and 4 send operations perform so, wave\*8\*4 and in addition there is one more send. Operation to send final maps to the manager

F(P,Wave)=(P)+(P)\*wave+wave\*8\*4+1 ~ O(P+P\*wave+wave)