CMPE 300 ANALYSIS OF ALGORITHMS

PROJECT 2 REPORT

a) We have completed the project. Our code is running successfully. We used checkered-split approach to implement the code.

b) Our project consists of two parts which is separated by a conditional statement. In the if part, the code checks whether the processors rank is 0. This part belongs to the manager processor. It first collects arguments. Then reads the given input file line by line. Then it initiates an 2D array called "table". This array carries all the significant information during the process. It is first filled via given input. Here, we decided to keep the info via an array of strings. Because we couldn't send an array of objects to other processors using MPI methods. Also, we first tried to use NumPy arrays. It had many facilities in terms of dividing the array as we please. However, we also needed to recombine the arrays and it was easier to the with normal arrays. So, we prefer it to be simpler and switched to the normal arrays. We even gave up keeping the info as an 2D array. Instead, we used one dimensional organized array. We then easily divided the array and distributed the parts to the other processors.

Then, we started to write Worker Processor's part. It was very hard to decide how to distribute the information of the edges. Our first method was distributing the information in four stages. First odd rows send, then even rows send, then columns send in a similar way. However, we realized that it was getting more and more complex as we tried to add diagonal transmissions. We tried to debug it, but it was quiet a mess. So, we found a more efficient way and started writing this part from the scratch. This time, we send and received the information one by one. First every processor sends its right part, then left part, then upper part and then lower part. After that the processors also sent their corners each dimension. We completed the transmission in 8 steps this way. But it got much easier.

After receiving all the needed information, we decided to wrap up the original array with the received information. We initiated arrays with ".0" elements. Which indicates that there is no tower in that location.

Then, we applied the war simulation to the important part of the array. During this, we kept the health points in an array and renewed the health points of the tower after each round. We deleted the towers which had health points equal or lower than 0 in the end of each round. Since in each round, edge information needed to be updated, we wrap up the code with an outer for loop. For waves, we have another outer for loop.

c) Since the communication between processors are much more time costly with respect to operations that are done in a processor, the time complexity of the algorithm mostly depends on that. The algorithm that we used in our code starts with all processors sending data to 8 sides one by one in a cyclic manner. But this operation takes place exactly 8 times independent of the processor number. Because all processors do this transfer concurrently time complexity is not affected. The time spend here solely depends on the number of waves. Total time spend in a wave, if time spend between two processors while communicating is t, is $8 * 8 * \mathrm{t}$. If we have w number of waves, we spend $64 * \mathrm{w} * \mathrm{t}$ amount of time. But during the processor zeros distribution of the input data to other processors we spend considerable amount of time. And, while receiving the output from the processors we spend similar amount of time. The time spent depends on number of processors used while sending and receiving. This operation is also repeated in every wave. If we have p number of processors (including processor zero), w number of waves and time spend between two processors while communicating is t; total time spend in sending and receiving data from zero to other processors is $2 * \mathrm{w} * (\mathrm{p} - 1) * \mathrm{t}$. Apart from these the time spend in a processor can be considered. Since we check every cell of the data in one processor, the time we spend is $8 * w * \left(\frac{N}{\sqrt{P-1}}\right)^2 * T$ (N = one side of the whole board, T time spend for one operation). Overall, usually communication between processors is more costly than operations in a processor number of basic operation is $2 * w * (p - 1) + 64 * w$. Time complexity can be expressed as $O(w * p)$.

d) Test outputs are provided in the zip file.

e) One of the hardest parts we have encountered was debugging. Since we have failed to download the MPI compiler to the windows, we used Virtual Box and It has no debug tools. Thus, we had to debug by using print function. And It was nearly impossible to do it after first input.

Also, it was hard to decide how to approach the problem to implement the checkered-split version. We first try to use more complex method and we quickly got lost in out own code. So, we reimplemented from the scratch and used much simpler but a little bit slower approach which we have explained in part b.