Distributed Operating System Project 3 Report

PA3_Team 14

Anirudh Atrey, Mudit Arya, Prateek Abbi, Syed Faizan Ali

(59919994, 32141614, 89387132, 26592828)

(aniruddh.atrey@ufl.edu, arya.mudit@ufl.edu, prateekabbi@ufl.edu, syed.ali1@ufl.edu)

How to Run the Program

To run the program, you will need the following frameworks in your system:

- .NET SDK: To run the project, your system must have .NET SDK installed.
- AKKA.NET: Since our project is using AKKA framework and each node is having its own actors, you need to act AKKA.NET framework in your system.

To add these packages run the following commands in your terminal:

- dotnet add package Akka
- dotnet add package Akka.FSharp

With these SDKs and framework mentioned above, you need only one command to run the code: "**dotnet run <nodes> <topology> <algorithm>".**

Replace <nodes> with actual number of nodes,
<topology> with either of the following four:

- full
- 2D
- line
- imp3D

<algorithm> with either of the following two:

- gossip
- push-sum

What is working

The Project is supposed to simulate the Chord Protocol with Actor model i.e., each node in the chord network should have an actor. For this we have used various functions (as described in the given paper). Below mentioned is the description of each function:

- construct_full_network: This function is used to create a complete graph, excluding self-connections. This method systematically connects every unique pair of nodes, creating a fully interconnected network. It iterates through a specified number of nodes, linking each pair. The connections for each node are stored in a mutable array, which is subsequently added to the main network array, passed by reference. As a result, a symmetric and fully connected network is formed.

- construct_2D_network: The `construct_2D_network` function is designed to create a two-dimensional grid network comprising various nodes. It begins by determining the size of the grid, accomplished by calculating the square root of the total number of nodes ({num_nodes}). To ensure the integrity of the network, it avoids connecting the edge nodes to external points. Instead, it establishes connections for each node with its immediate neighbors—those located above, below, to the left, and to the right. This network is represented through an array, where each node is associated with an array of its connected neighbors. This type of network structure is particularly useful in scenarios that require the analysis of spatial interactions, such as in certain graph-based algorithms and simulations.

- construct_line_network: In F#, this function creates a sequential network structure. It takes two inputs: an array of arrays representing an existing network, and a total node count, {num_nodes}. The function establishes connections based on each node's index within the range 0 to {num_nodes - 1}. Every intermediate node is linked to its adjacent nodes - the one before and the one after. Specifically, the first node (index 0) is connected only to its subsequent node, while the final node (index num_nodes - 1) is linked to its preceding node. This forms a network akin to a straight line or chain, where each node is linked to its immediate neighbors.

- construct_imperfect_2D_network: The `construct_imperfect_3D_network` function is designed to create a network structure within a three-dimensional (3D) space, tailored for a specified number of nodes. Initially, it calculates the length of the sides of a cube that will encompass all the nodes, aiming to distribute them evenly across the three dimensions. The function assigns 3D coordinates (x, y, z) to each node based on its index.
  Subsequently, the program identifies the nearby neighbors for each node within this 3D grid. It considers all adjacent nodes (dx, dy, dz), ensuring that each neighbor falls within the grid's limits. These neighboring relationships are documented in an array, outlining the connections between the nodes.

- node_function: This function is designed to execute two distinct distributed algorithms: one centered around a gossip-based communication method, and the other, a push-sum algorithm, which is employed for collective computation purposes. This function specifically outlines the operational conduct of a node within the system. Let's delve into a breakdown of its critical elements and operational mechanisms.
  - **Main Loop (loop Function):** This recursive function serves as the ongoing operational core of the node. It actively monitors and reacts to incoming messages:
    - Init: Establishes the node's adjacent nodes, its own index, the mode of operation (either gossip or push-sum), and the criteria for failure.
    - Gossip: Manages the reception of gossip messages. When active, the node disseminates the gossip message to a randomly chosen neighbor and informs the boss_actor upon initial receipt.
    - Triggering_Gossip: Similar to the gossip process, this function is utilized to stimulate the spread of gossip.
    - Push: Engages with the push-sum algorithm by addressing incoming push messages. It recalculates the new ratio after updating the overall sum and the weight.
    - Triggering_Push: This routine initiates the push-sum procedure. It verifies for convergence and, in its absence, splits the total and weight, sending one part to a random neighbor while retaining the remainder.

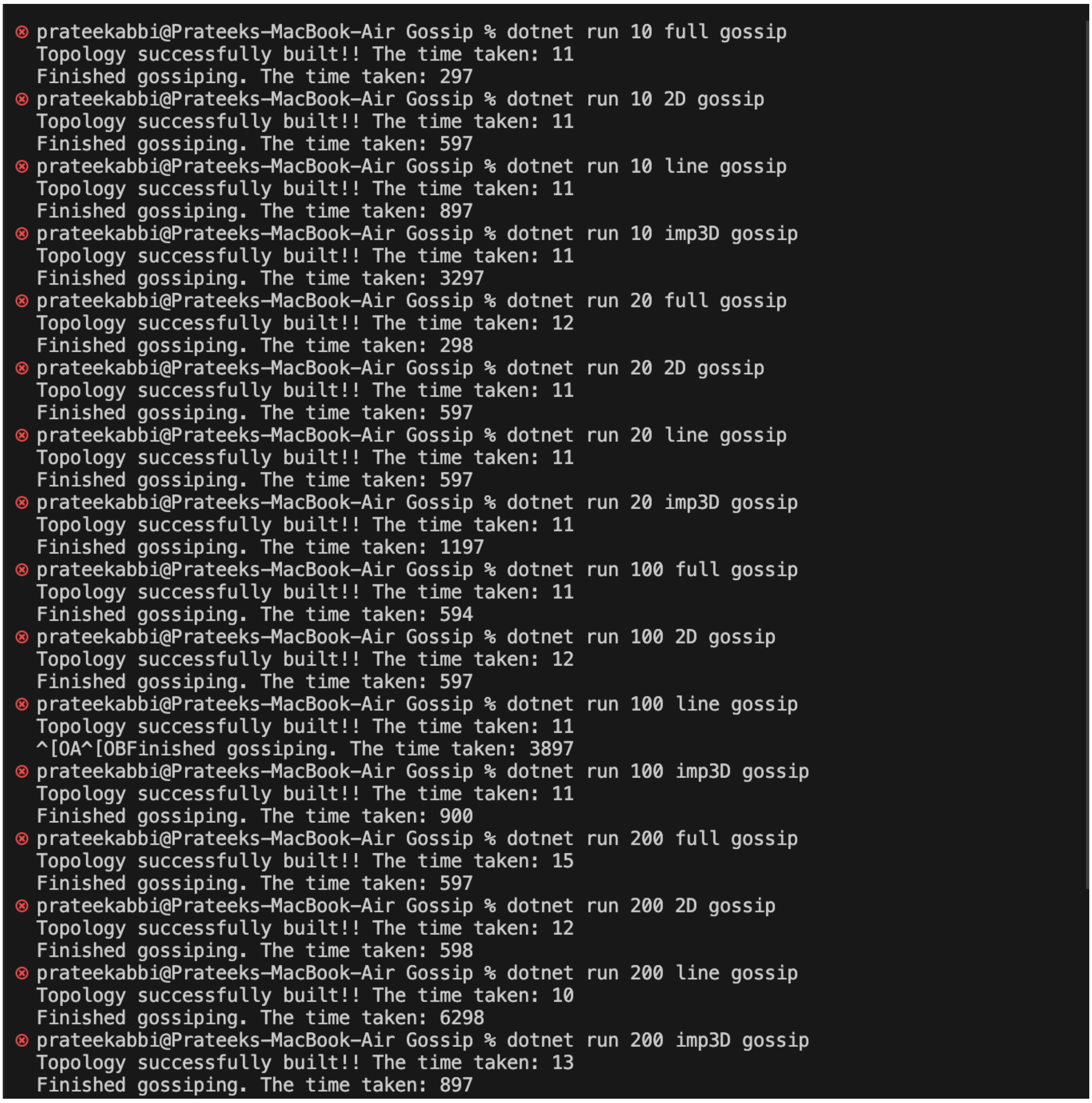
- boss_gossip: This function is designed to set up a communication network based on gossip within a distributed system. This is particularly relevant when utilizing the Akka.NET framework, which is grounded in the actor model. In the realm of concurrent computing, the actor model serves as a foundational concept. It treats actors as the fundamental elements of concurrency. Within this framework, each actor has the capability to receive and process messages, send them to other actors, modify its own internal state, and even create new actors. This method is widely employed in .NET applications, facilitated by the Akka.NET framework. The `boss_gossip` function is a practical implementation of these concepts, enabling efficient and scalable communication in distributed systems.
  - **Initialization:** The script sets up changeable variables to keep an eye on the state, including node_array_actor, num_nodes, and count. These variables store details about the nodes in the system, such as the total number of nodes and the number of messages or actions handled.
  - **Boss Timer:** A timing mechanism (timer_of_boss) is utilized to track the duration of the gossip procedure.
  - **Recursive Loop Function:** The core operation of the actor is maintained through a recursive function named Loop. This function outlines how the boss actor reacts to different kinds of messages (boss_message).
  - **Message Recognition:** To manage various message types, the function utilizes pattern recognition:

    - Start_Gossip: This step begins the gossiping process by dispatching a gossip message to a single node.
    - Received_Gossip: Upon receiving gossip, the node_array_actor gets an update.
    - Gossiping: This initiates further communication with nodes in the node_array_actor.
    - Finished_Gossip: This stage deals with the end of gossiping, updating the node_array_actor and count, and records the duration taken if all nodes have been reached.
    - Start_Push: This initiates push-based communication (like calculating averages or sums across nodes) and begins by sending initial messages to nodes.
    - Pushing: This action allows nodes to receive updates based on the push method.
    - Finished_Push: Similar to Complete_Gossip, this step manages the completion of the push-sum algorithm.
  - **Control Flow:** This technique employs F#'s actor computation expression for managing both asynchronous message exchanges and the state of the actor. The let! binding serves to asynchronously acquire messages from the actor's mailbox.
  - **Termination:** Upon completion of the gossip or push-sum process (either all nodes are contacted or the appropriate ratio is achieved), the function logs the time elapsed and terminates the program.

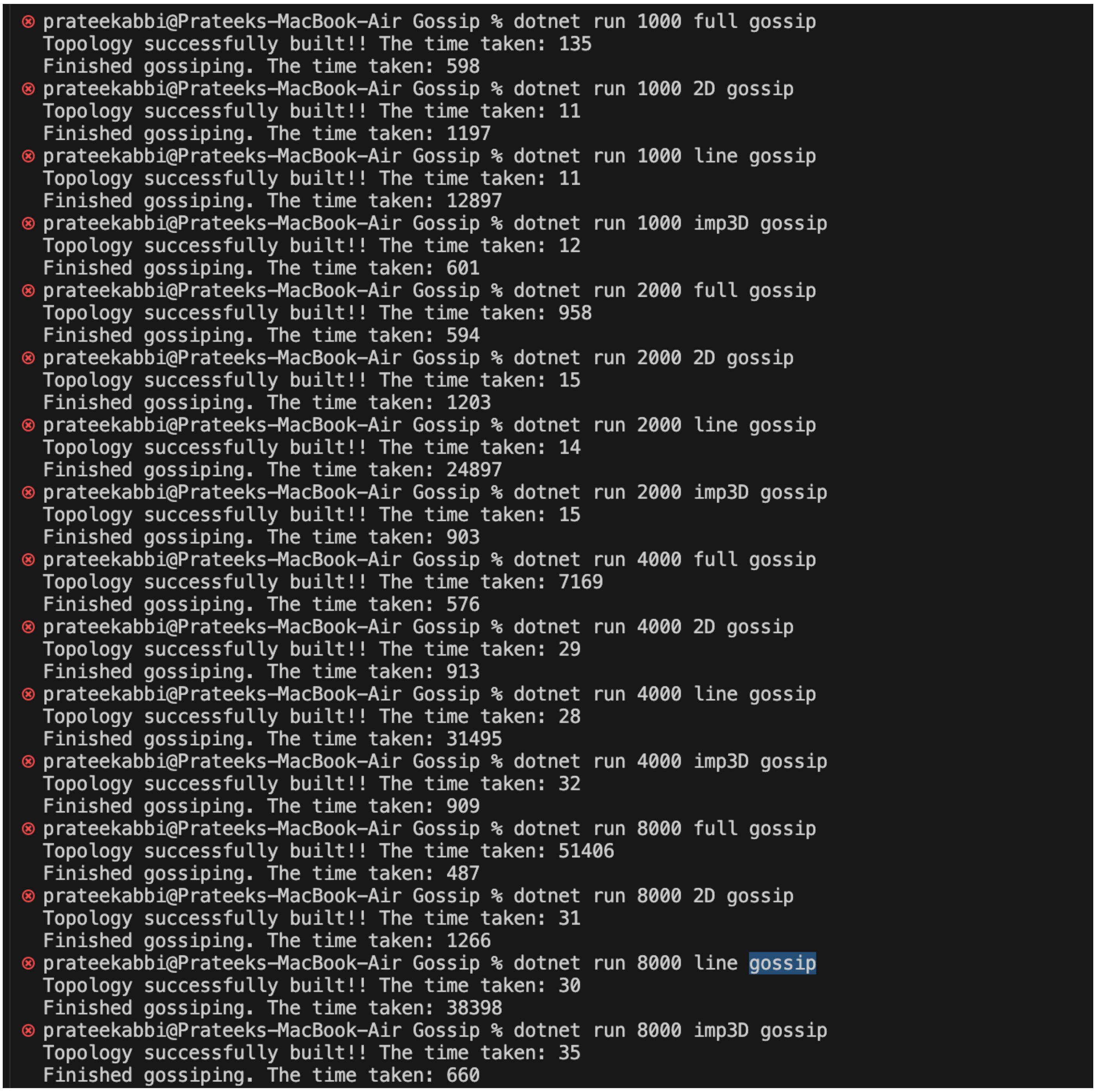
- main function: The main function is a component of a simulation in F# that simulates distributed computing. It establishes a network of various nodes and implements specific algorithms. These algorithms demonstrate the processes of exchanging information and performing computations across a distributed network.
  - **Command Line Argument Handling:** This method requires three inputs: the total count of nodes (nodes), the structural design of the network (topology), and the chosen method for operation (algorithm). It doesn't directly process command line inputs, rather it anticipates these inputs as parameters.
  - **Initialization:** The method initiates by activating a timer to track duration. It establishes a blank array to depict the interconnections between nodes. Additionally, it constructs the actor 'boss_gossip.' While the specific function of this actor isn't detailed in the provided segment, it is presumed that this actor plays a pivotal role in managing activities throughout the network.
  - **Node Creation:** The array_node_actor generates a series of node actors, with each actor symbolizing a different network node.
  - **Building the Network Structure:** Depending on the specified topology (full, 2D, line, imp2D, or a fallback default that exits the application if the topology is unidentified), the network's layout is determined. This influences how nodes are interconnected in various configurations.
  - **Node Setup:** An Init message is dispatched to each node actor, detailing their connections, identification number, and relevant information, to prepare them for operation.
  - **Algorithm Execution:** The function initiates a verification of the provided algorithm, either gossip or push-sum. For gossip, it launches a communication protocol where nodes in the network circulate a specific message ("Fire!"). In the case of push-sum, it starts a distributed numerical calculation among the nodes. Additionally, the function consistently monitors to determine if a set period (gossip_interval) has passed, which is essential for the continuation of the respective algorithm.
  - **Error Managing:** To effectively manage errors, it's crucial to address exceptions such as 'index out of range' and issues related to network or node configurations.
  - **Exit Conditions:** The continuous operation suggested by the while loops in both the gossip and push-sum algorithms is a point of concern. The current implementation does not provide clear exit criteria, potentially leading to indefinite execution or requiring manual termination. This approach needs to be restructured to ensure controlled and predictable algorithm termination.

Screenshots of the output

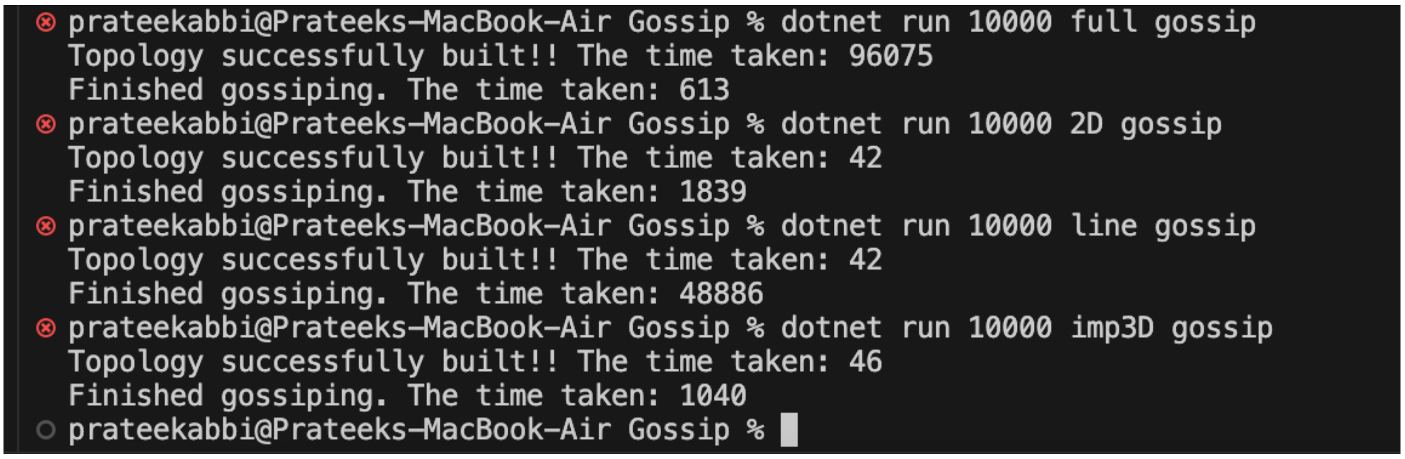
Gossip



```
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 full gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 297
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 2D gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 597
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 line gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 897
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 imp3D gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 3297
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 full gossip
  Topology successfully built!! The time taken: 12
  Finished gossiping. The time taken: 298
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 2D gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 597
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 line gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 597
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 imp3D gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 1197
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 full gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 594
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 2D gossip
  Topology successfully built!! The time taken: 12
  Finished gossiping. The time taken: 597
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 line gossip
  Topology successfully built!! The time taken: 11
  ^[OA^[OBFinished gossiping. The time taken: 3897
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 imp3D gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 900
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 full gossip
  Topology successfully built!! The time taken: 15
  Finished gossiping. The time taken: 597
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 2D gossip
  Topology successfully built!! The time taken: 12
  Finished gossiping. The time taken: 598
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 line gossip
  Topology successfully built!! The time taken: 10
  Finished gossiping. The time taken: 6298
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 imp3D gossip
  Topology successfully built!! The time taken: 13
  Finished gossiping. The time taken: 897
```

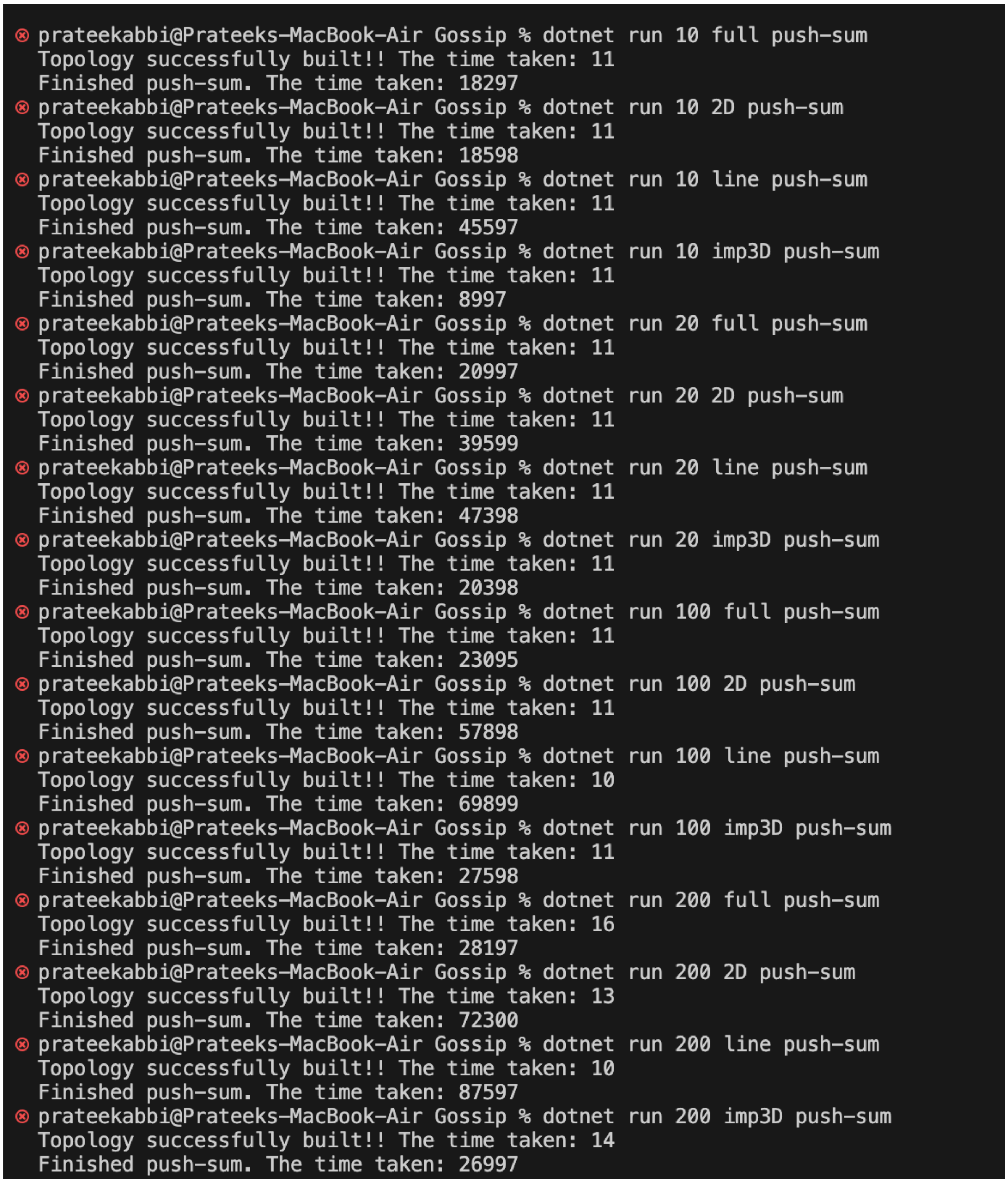


```
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 full gossip
  Topology successfully built!! The time taken: 135
  Finished gossiping. The time taken: 598
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 2D gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 1197
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 line gossip
  Topology successfully built!! The time taken: 11
  Finished gossiping. The time taken: 12897
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 imp3D gossip
  Topology successfully built!! The time taken: 12
  Finished gossiping. The time taken: 601
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 full gossip
  Topology successfully built!! The time taken: 958
  Finished gossiping. The time taken: 594
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 2D gossip
  Topology successfully built!! The time taken: 15
  Finished gossiping. The time taken: 1203
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 line gossip
  Topology successfully built!! The time taken: 14
  Finished gossiping. The time taken: 24897
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 imp3D gossip
  Topology successfully built!! The time taken: 15
  Finished gossiping. The time taken: 903
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 full gossip
  Topology successfully built!! The time taken: 7169
  Finished gossiping. The time taken: 576
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 2D gossip
  Topology successfully built!! The time taken: 29
  Finished gossiping. The time taken: 913
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 line gossip
  Topology successfully built!! The time taken: 28
  Finished gossiping. The time taken: 31495
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 imp3D gossip
  Topology successfully built!! The time taken: 32
  Finished gossiping. The time taken: 909
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 full gossip
  Topology successfully built!! The time taken: 51406
  Finished gossiping. The time taken: 487
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 2D gossip
  Topology successfully built!! The time taken: 31
  Finished gossiping. The time taken: 1266
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 line gossip
  Topology successfully built!! The time taken: 30
  Finished gossiping. The time taken: 38398
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 imp3D gossip
  Topology successfully built!! The time taken: 35
  Finished gossiping. The time taken: 660
```

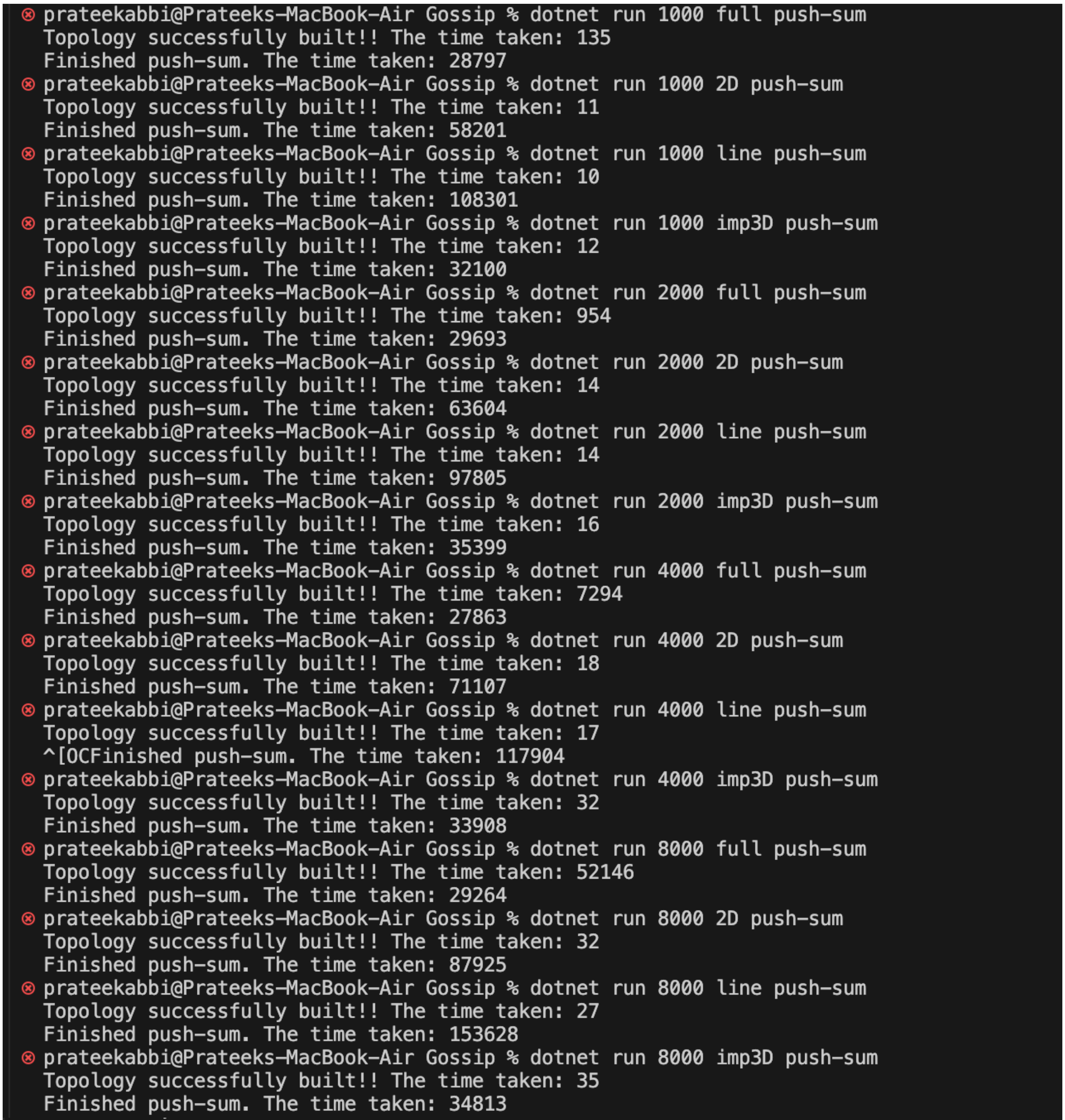


```
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 full gossip
  Topology successfully built!! The time taken: 96075
  Finished gossiping. The time taken: 613
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 2D gossip
  Topology successfully built!! The time taken: 42
  Finished gossiping. The time taken: 1839
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 line gossip
  Topology successfully built!! The time taken: 42
  Finished gossiping. The time taken: 48886
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 imp3D gossip
  Topology successfully built!! The time taken: 46
  Finished gossiping. The time taken: 1040
○ prateekabbi@Prateeks-MacBook-Air Gossip %
```

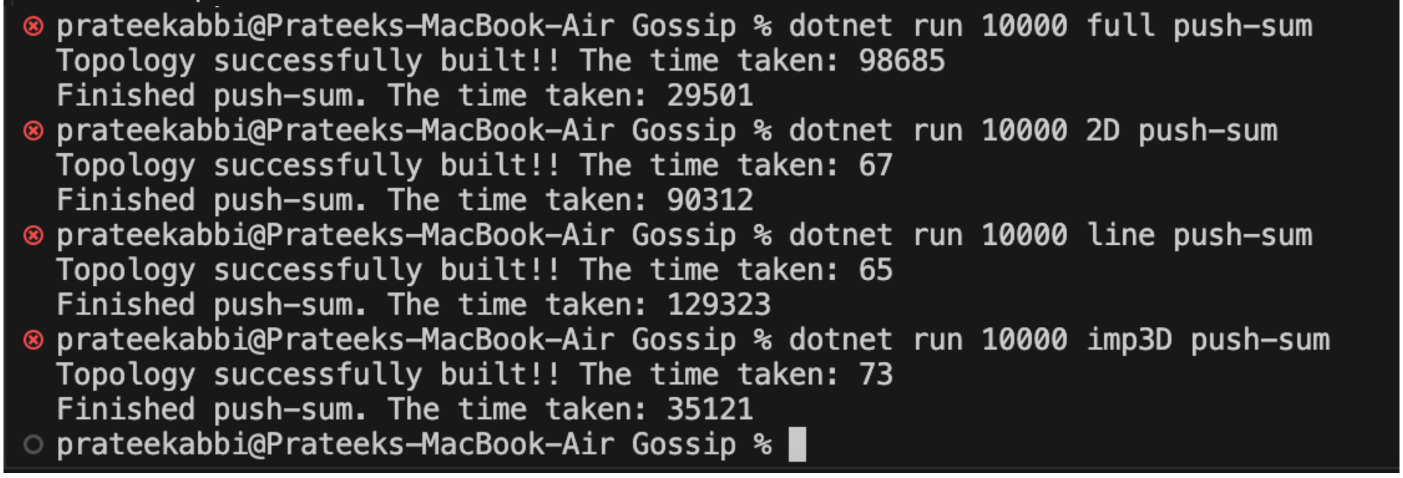
Push Sum



```
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 full push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 18297
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 2D push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 18598
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 line push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 45597
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10 imp3D push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 8997
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 full push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 20997
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 2D push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 39599
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 line push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 47398
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 20 imp3D push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 20398
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 full push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 23095
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 2D push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 57898
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 line push-sum
  Topology successfully built!! The time taken: 10
  Finished push-sum. The time taken: 69899
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 100 imp3D push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 27598
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 full push-sum
  Topology successfully built!! The time taken: 16
  Finished push-sum. The time taken: 28197
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 2D push-sum
  Topology successfully built!! The time taken: 13
  Finished push-sum. The time taken: 72300
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 line push-sum
  Topology successfully built!! The time taken: 10
  Finished push-sum. The time taken: 87597
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 200 imp3D push-sum
  Topology successfully built!! The time taken: 14
  Finished push-sum. The time taken: 26997
```

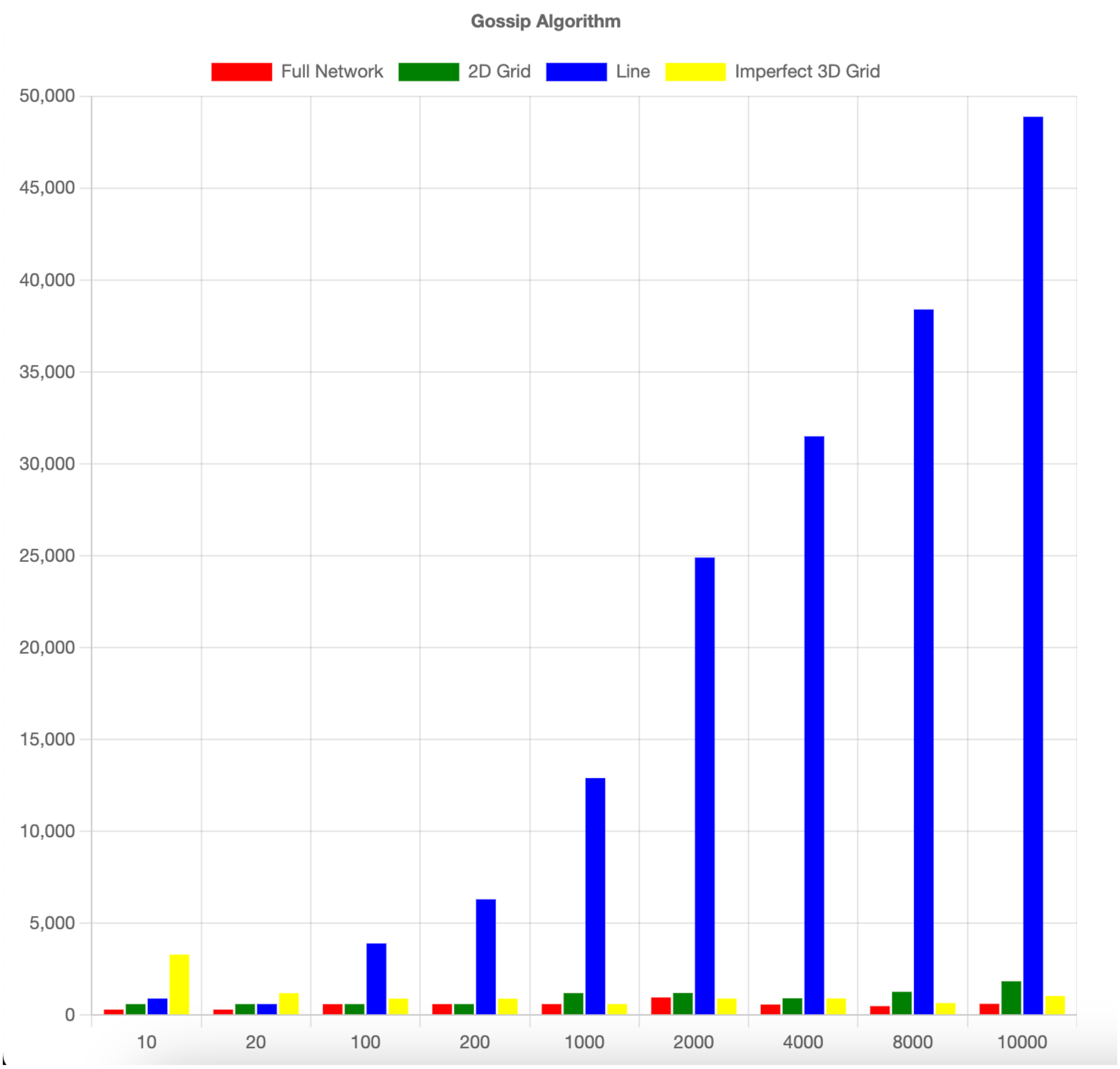


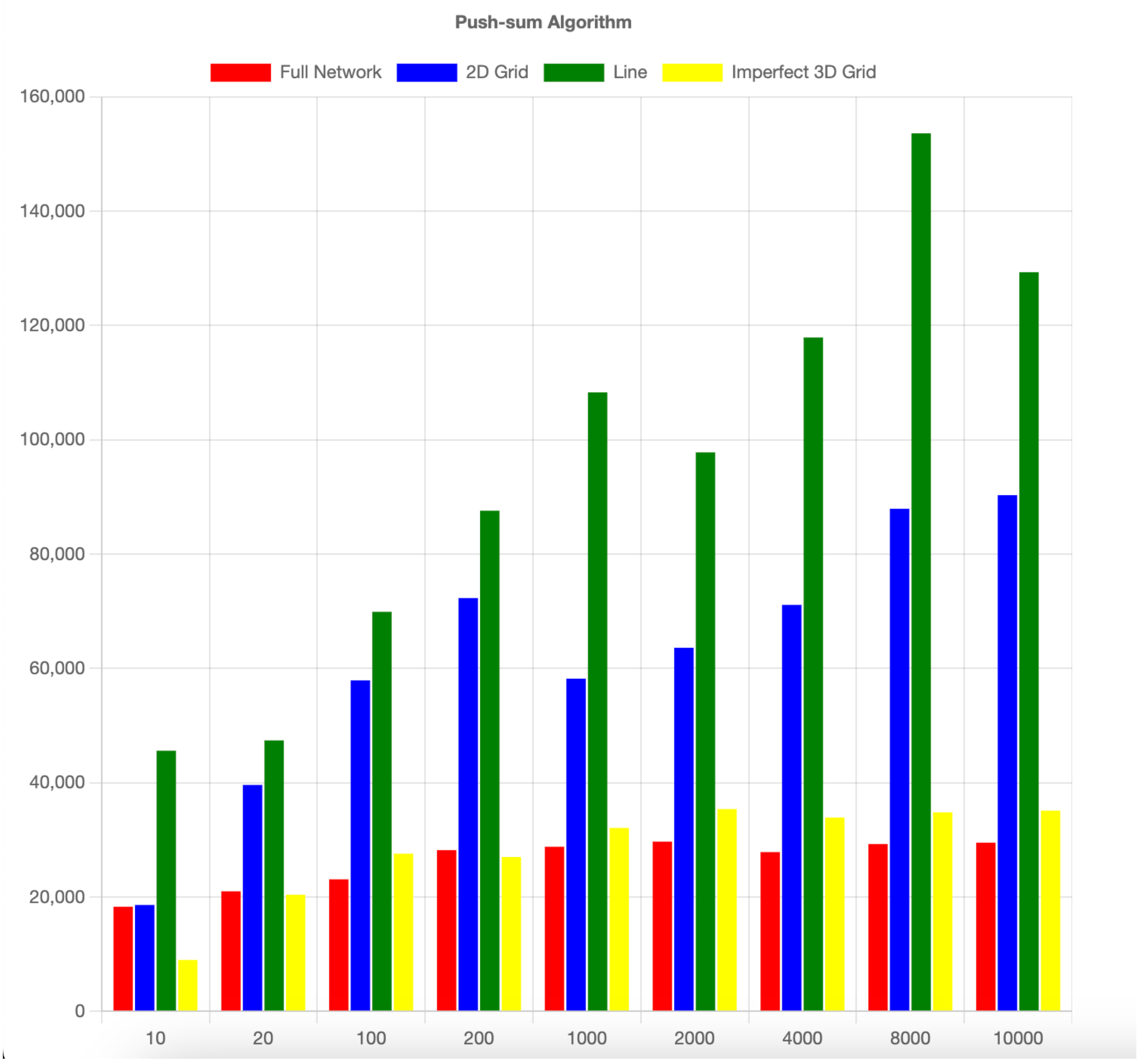
```
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 full push-sum
  Topology successfully built!! The time taken: 135
  Finished push-sum. The time taken: 28797
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 2D push-sum
  Topology successfully built!! The time taken: 11
  Finished push-sum. The time taken: 58201
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 line push-sum
  Topology successfully built!! The time taken: 10
  Finished push-sum. The time taken: 108301
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 1000 imp3D push-sum
  Topology successfully built!! The time taken: 12
  Finished push-sum. The time taken: 32100
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 full push-sum
  Topology successfully built!! The time taken: 954
  Finished push-sum. The time taken: 29693
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 2D push-sum
  Topology successfully built!! The time taken: 14
  Finished push-sum. The time taken: 63604
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 line push-sum
  Topology successfully built!! The time taken: 14
  Finished push-sum. The time taken: 97805
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 2000 imp3D push-sum
  Topology successfully built!! The time taken: 16
  Finished push-sum. The time taken: 35399
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 full push-sum
  Topology successfully built!! The time taken: 7294
  Finished push-sum. The time taken: 27863
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 2D push-sum
  Topology successfully built!! The time taken: 18
  Finished push-sum. The time taken: 71107
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 line push-sum
  Topology successfully built!! The time taken: 17
  ^[OCFinished push-sum. The time taken: 117904
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 4000 imp3D push-sum
  Topology successfully built!! The time taken: 32
  Finished push-sum. The time taken: 33908
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 full push-sum
  Topology successfully built!! The time taken: 52146
  Finished push-sum. The time taken: 29264
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 2D push-sum
  Topology successfully built!! The time taken: 32
  Finished push-sum. The time taken: 87925
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 line push-sum
  Topology successfully built!! The time taken: 27
  Finished push-sum. The time taken: 153628
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 8000 imp3D push-sum
  Topology successfully built!! The time taken: 35
  Finished push-sum. The time taken: 34813
```



```
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 full push-sum
  Topology successfully built!! The time taken: 98685
  Finished push-sum. The time taken: 29501
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 2D push-sum
  Topology successfully built!! The time taken: 67
  Finished push-sum. The time taken: 90312
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 line push-sum
  Topology successfully built!! The time taken: 65
  Finished push-sum. The time taken: 129323
⊗ prateekabbi@Prateeks-MacBook-Air Gossip % dotnet run 10000 imp3D push-sum
  Topology successfully built!! The time taken: 73
  Finished push-sum. The time taken: 35121
○ prateekabbi@Prateeks-MacBook-Air Gossip % 
```

Graph of “Number of nodes” vs “Avg. Hop Count”





Largest Network

We were able to make largest network with 10000 nodes We could have gone more but it was taking too much of time. So, we decided to stop at 10000 nodes only.