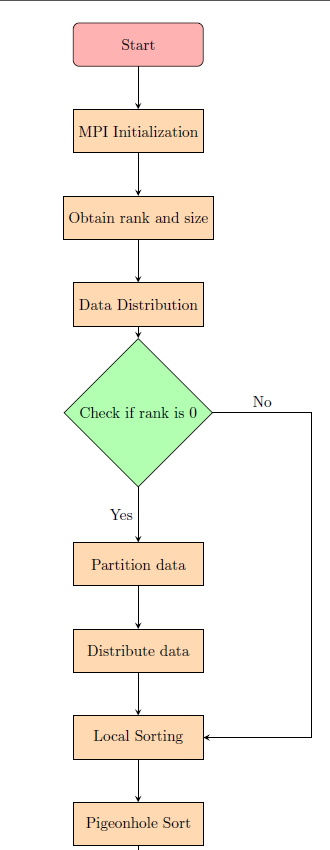
**3.2.4 MPI Flowchart**

****

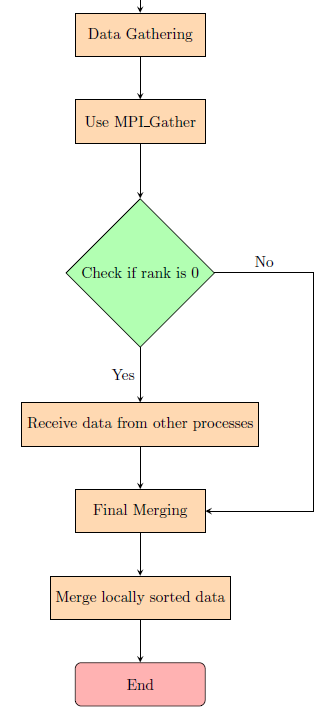
****

Fig 2

**Given figure tells about the following:**

Start:

The process begins with the "Start" node, indicating the starting point of the flowchart.

MPI Initialization:

This step involves initializing MPI for parallel processing.

Obtain rank and size:

MPI assigns a unique rank to each process and provides the total number of processes (size). This information is crucial for subsequent steps.

Data Distribution:

The process checks whether the rank is 0. If yes, it proceeds to partition and distribute the input data among different MPI processes.

Check if rank is 0:

A decision point. If the rank is 0, it indicates the master process, which performs additional tasks such as data partitioning.

Partition data:

If the rank is 0, this step involves partitioning the input data into chunks for distribution among MPI processes.

Distribute data:

Distribute the partitioned data to the corresponding MPI processes.

Local Sorting:

Each MPI process independently performs local sorting using the Pigeonhole Sort algorithm on its assigned subset of data.

Pigeonhole Sort:

This step represents the Pigeonhole Sort algorithm applied locally to each subset of data.

Data Gathering:

Collect the locally sorted data from all MPI processes.

Use MPI\_Gather:

MPI function to gather the locally sorted data. This step is particularly important for the master process.

Check if rank is 0 (again):

Another decision point. If the rank is 0, it indicates the master process, which proceeds to the next step.

Receive data from other processes:

The master process receives locally sorted data from all other processes.

Final Merging:

The master process merges the locally sorted subsets into the final globally sorted array.

Merge locally sorted data:

This step involves the actual merging of locally sorted subsets.

End:

The process concludes with the "End" node, indicating the end of the flowchart.

**3.3.3 Cuda Flowchart**

A diagram of a diagram

Description automatically generated

Fig 3

Start: The beginning of the flowchart.

CUDA Initialization: Initialize CUDA, which allows you to harness the parallel processing capabilities of GPUs.

Allocate Memory: Allocate memory on the GPU for storing data.

Transfer Data: Copy the input data from the host (CPU) to the allocated memory on the GPU.

CUDA Kernel: Design a CUDA kernel to perform the Pigeonhole Sorting algorithm on the GPU.

Launch Configuration: Specify the configuration for launching the CUDA kernel, determining the number of blocks and threads.

Sorting Operation: Implement the core Pigeonhole Sorting algorithm within the CUDA kernel.

Transfer Back: Copy the sorted data from the GPU back to the host.

Deallocate Memory: Free the memory allocated on the GPU.

End: The end of the flowchart.