LP5 MANUAL QUESTIONS :

1.What is BFS?

BFS (Breadth-First Search) is an algorithm for traversing or searching tree or graph data structures. It explores all nodes at the present depth level before moving on to nodes at the next depth level.

The algorithm uses a QUEUE data structure .

BFS is commonly used in many applications, such as finding the shortest path between two nodes, solving puzzles, and searching through a tree or graph.

2.Steps involved in traversing a graph by using Breadth-First Search:

Step 1: Take an Empty Queue.

Step 2: Select a starting node (visiting a node) and insert it into the Queue.

Step 3: Provided that the Queue is not empty, extract the node from the Queue and insert its child nodes (exploring a node) into the Queue.

Step 4: Print the extracted node.

3.DFS :

DFS (Depth-First Search) is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It uses a stack (or recursion) and is useful for tasks like pathfinding, cycle detection, and topological sorting.

The algorithm uses a STACK data structure.

DFS can be implemented using either a recursive or an iterative approach. The recursive approach is simpler to implement but can lead to a stack overflow error for very large graphs. The iterative approach uses a stack to keep track of nodes to be explored and is preferred for larger graphs.

4.Concept of OpenMP :

OpenMP (Open Multi-Processing) is an application programming interface (API) that supports shared-memory parallel programming in C, C++, and Fortran. It is used to write parallel programs that can run on multicore processors, multiprocessor systems, and parallel computing clusters.

OpenMP is widely used in scientific computing, engineering, and other fields that require

high-performance computing. It is supported by most modern compilers and is available on a wide range of platforms, including desktops, servers, and supercomputers.

5.Parallel BFS :

Parallel BFS is a version of Breadth-First Search where multiple nodes at the same level are processed simultaneously using multiple threads or processors, speeding up traversal on large graphs. It’s often used in high-performance computing.

6.What is Bubble Sort?

Bubble Sort is a simple sorting algorithm that works by repeatedly swapping adjacent elements if they are in the wrong order. It is called "bubble" sort because the algorithm moves the larger elements towards the end of the array in a manner that resembles the rising of bubbles in a liquid.

: The time complexity of Bubble Sort is O(n^2), which makes it inefficient for large lists.

: it is useful for educational purposes and for sorting small datasets.

7.Parallel Bubble Sort ?

Parallel Bubble Sort is a modification of the classic Bubble Sort algorithm that takes advantage of parallel processing to speed up the sorting process.

: In parallel Bubble Sort, the list of elements is divided into multiple sublists that are sorted concurrently by multiple threads.

8.Parallel Reduction Operation :

Parallel reduction is a common technique used in parallel computing to perform a reduction operation on a large dataset. A reduction operation combines a set of values into a single value, such as computing the sum, maximum, minimum, or average of the values.

: The parallel reduction algorithm works by dividing the input data into smaller chunks that can be processed independently in parallel.

:Parallel reduction has many applications in scientific computing, machine learning, data analytics, and computer graphics. It can be used to compute the sum, maximum, minimum, or average of large datasets, to perform data filtering, feature extraction, or image processing, to solve optimization problems, or to accelerate numerical simulations. Parallel reduction can also be combined with other parallel algorithms, such as parallel sorting, searching, or matrix operations, to achieve higher performance and scalability.

9.CUDA architecture:

CUDA is a parallel computing platform and programming model developed by NVIDIA. It allows developers to use the power of GPU (Graphics Processing Unit) to accelerate computations. CUDA architecture consists of host and device components, where the host is the CPU and the device is the GPU.

10.CUDA programming model:

CUDA programming model consists of host and device codes. The host code runs on the CPU and is responsible for managing the GPU memory and launching the kernel functions on the device. The device code runs on the GPU and performs the computations.

11.CUDA kernel function:

A CUDA kernel function is a function that is executed on the GPU. It is defined with the global keyword and is called from the host code using a launch configuration. Each kernel function runs in parallel on multiple threads, where each thread performs the same operation on different data.

12.Memory management in CUDA:

In CUDA, there are three types of memory: global, shared, and local. Global memory is allocated on the device and can be accessed by all threads. Shared memory is allocated on the device and can be accessed by threads within a block. Local memory is allocated on each thread and is used for temporary storage.

13.CUDA thread organization:

In CUDA, threads are organized into blocks, and blocks are organized into a grid. Each thread is identified by a unique thread index, and each block is identified by a unique block index.

14.Matrix multiplication:

Matrix multiplication is a fundamental operation in linear algebra. It involves multiplying two matrices and producing a third matrix. The resulting matrix has dimensions equal to the number of rows of the first matrix and the number of columns of the second matrix.

15. CUDA ?

CUDA stands for Compute Unified Device Architecture. It is a parallel computing platform and programming model developed by NVIDIA.CUDA allows developers to use the power of the GPU to accelerate computations. It is designed to be used with C, C++, and Fortran programming languages.CUDA architecture consists of host and device components. The host is the CPU, and the device is the GPU. The CPU is responsible for managing the GPU memory and launching the kernel functions on the device.

:CUDA devices have a hierarchical memory architecture consisting of multiple memory levels, including registers, shared memory, L1 cache, L2 cache, and global memory.

:CUDA supports various libraries, including cuBLAS for linear algebra, cuFFT for Fast Fourier Transform, and cuDNN for deep learning.

:CUDA programming requires a compatible NVIDIA GPU and an installation of the CUDA Toolkit, which includes the CUDA compiler, libraries, and tools.

16.CUDA Program for Matrix Multiplication (explaination):

The program multiplies two matrices of size n using CUDA. It first allocates host memory for the matrices and initializes them. Then it allocates device memory and copies the matrices to the device. It sets the kernel launch configuration and launches the kernel function matrix_multiply. The kernel function performs the matrix multiplication and stores the result in matrix c. Finally, it copies the result back to the host and frees the device and host memory.

17.HPC ?

HPC (High-Performance Computing) uses powerful computers and parallel processing to solve complex, large-scale problems quickly—common in science, engineering, and data analysis.

18.General approach for implementing an HPC application for AI/ML domain :

1. Data Parallelism – Split data across nodes/GPUs.
2. Model Parallelism – Split model across devices (for large models).
3. Use HPC Libraries/Frameworks – e.g., MPI, Horovod, NCCL.
4. Optimize I/O & Memory – Fast data loading, minimize bottlenecks.
5. Leverage Accelerators – GPUs, TPUs for faster computation.
6. Scale & Monitor – Use clusters, monitor resource usage.