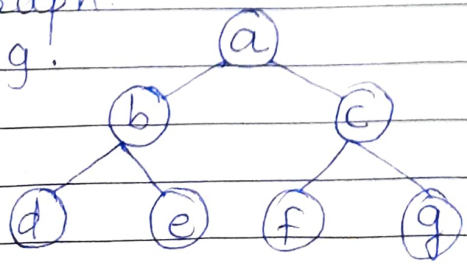
Assignment No. 1(A)

Title: Design and implement Parallel Breadth First search based on existing algorithms using OpenMP. Use a tree or an undirected graph for BFS.

Objective: To perform Parallel Breadth First search based on existing algorithms using OpenMP.

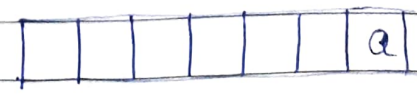
Prerequisite:
1. Basic of programming language
2. Concepts of BFS
3. Concept of Parallelism.

Theory:
- BFS is a graph traversal algorithm used to explore all the nodes of a graph or tree systematically starting from the root node & visiting all the neighboring nodes at the current depth level before moving on to the next depth level.
- It uses queue data structure to keep track of the nodes that need to be visited and marks each visited node to avoid processing it again.
- Applications of BFS are finding shortest path, solving puzzles, searching through a tree or graph
- e.g.



Queue:

Print a:



Print b: | | | | c | d | c |

Print c: | | | g | f | e | d |

Print d: | | | | g | f | e |

Print e: | | | | | g | f |

Print f: | | | | | | g |

Print g: | | | | | | |

- OpenMP (Open Multi-Processing) is an API that supports shared-memory parallel programming in C, C++, Fortran.
- OpenMP provides a set of directives & functions that can be inserted into the source code of a program to parallelize its execution.
- OpenMp is used in scientific computing, engineering and other fields that require high performance computing.

Conclusion: Hence we achieved parallelism while implementing BFS.

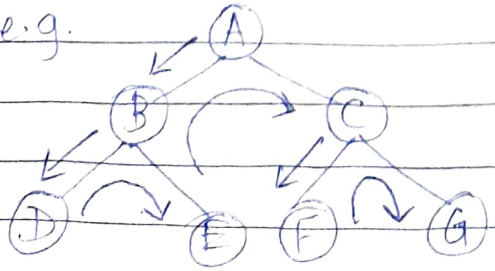
Assignment No. 1(B)

Title: Design and implement Parallel Depth First Search on existing algorithms using OpenMP. Use a Tree or an undirected graph for DFS.

Objective: To perform Parallel Depth First Search based on existing algorithms using OpenMP.

Prerequisite:
1. Basic of programming language
2. Concepts of DFS.
3. Concept of Parallelism.

Theory:
- DFS stands for Depth-First Search. It is a graph traversal algorithm that explores as far as possible along each before backtracking.
- DFS can be implemented using either a recursive or an iterative approach.
- Recursive approach is simpler to implement but can lead to a stack overflow error for very large graphs.
- Iterative approach uses a stack to keep track of nodes to be explored.
- A standard DFS implementation puts each vertex of the graph into one of two categories:
  1. Visited
  2. Not Visited.
- e.g.



- Parallel DFS works by dividing the graph into smaller subgraphs that are explored simultaneously.
- To explore subgraph, the processors maintain a stack data structure that store the nodes in the order of exploration.
- Parallel DFS can be implemented using several parallel computing models such as OpenMP, MPI & CUDA.
- In OpenMP, # pragma omp parallel for directive is used to distribute the work among multiple threads. By using this directive, each thread operates on a different part of the graph, which increases the performance of DFS algorithm.

Conclusion: In this way we achieved parallelism while implementing DFS.

Assignment No: 2

Title: Write a program to implement parallel Bubble sort & Merge sort using OpenMP. Use existing algorithm & measure the performance of sequential & parallel algorithm.

Objectives:
1. To understand parallel Bubble sort
2. To understand parallel Merge sort.

Theory:

① Bubble sort

→ • There are two phases in this algorithm called as odd even phases. In this algorithm, n elements are sorted in n phases where n is even.

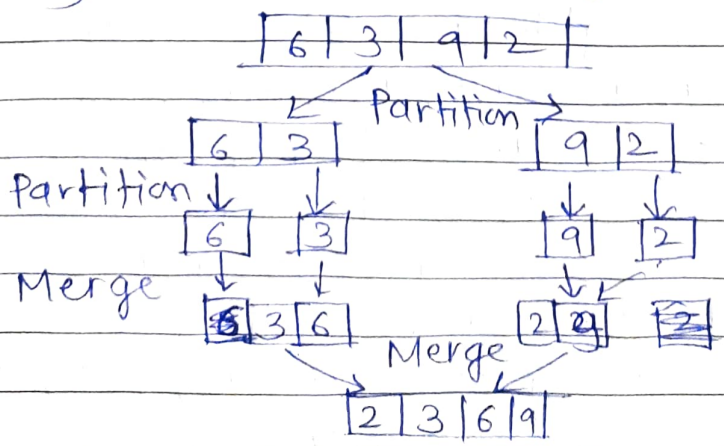
- Consider a sequence to be sorted is $\langle a_1, a_2 \cdots a_n \rangle$ The odd phase works on this basis that the elements with odd indices are compared with their neighbors are and found as out of sequence they are exchanged.
- This means the pairs with odd indices & their neighbours are compared.
  e.g. $(a_1, a_2), (a_3, a_4), \ldots (a_{n-1}, a_n)$ are compared and exchanged if not in proper sequence assume n as even here.
- After comparison among a pair & if found as out of sequence they are exchanged. This means the pairs $(a_2, a_3), (a_4, a_5) \ldots (a_n, a_{n+1})$ are compared exchanged.
- Algorithm:

```
Algorithm Even_odd(n)
{
  for(i=1;i<=n;i++)
    {
      if(i%2!=0)
        {
          for(j=0; j<=(n/2-1)
            { perform_exchange (a2j+1, a2j+2);
            }
        }
      else
      {
        for(j=0; j<=(n/2-1))
          { perform_exchange (a2j, a2j+1)
          }
      }
    }
}
```

② Merge sort:

→ • It first divides the unsorted listed into smallest possible sub-lists compares it with the adjacent list.

• It implements parallelism every nicely by following the divide & conquer algorithm.



Conclusion: Hence we implemented parallel sorting algorithm i.e. Bubble sort & Merge sort.

Assignment No: 3

Title: Implement Min, Max, Sum and Average operations using parallel reduction

Objective:
1. To understand parallel reduction operations.
2. To understand vectors operations.

Theory:

- CUDA

a) CUDA (Compute Unified Service Architecture) is a parallel computing platform and application programming interface model created by NVIDIA.

b) It allows software developers & software engineers to use CUDA enabled graphics processing unit for general purpose processing an approach termed GPU.

c) It is also a software layer that gives direct access to the GPU's virtual instruction set & parallel computational element for the execution of compute kernels.

d) This accessibility makes it easier for the specialist in parallel programming to use GPU resources in contact to prior API's like direct 3D and openGL.

- Min - Max operations:

① Max method

→ It returns the larger element a and b compare function can be emitted.
→ Syntax:
max(object_type a, object_type b, compare function)

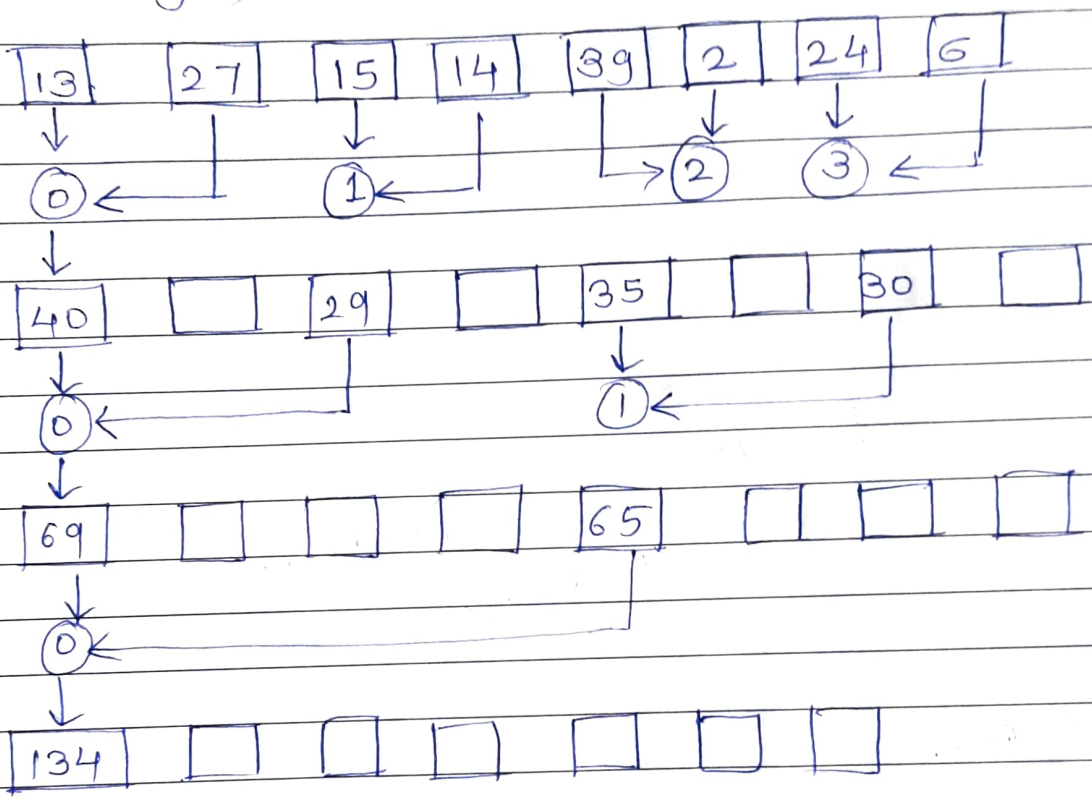
② Min Method
→ It returns the smaller element of a and b compare function can be omitted. if there is no compare function used in min() then elements are compared function used to determine which one of the object is smaller than the objects a and b are non-numeric type.

③ Arithmetic mean operations:
→ Basic arithmetic operation are addition, subtraction, multiplication and division although this subject also includes more advanced operations.

④ standard deviation:
→ It is represented by sigma (σ) which measures the amount of variation or dispersion of a set of data values.

• Parallel Reduction
→ Reduction operation reduce the collection of values to a single value.
→ Some operations like addition, manipulation, bitwise AND/OR/XOR, finding maximum and

minimum amongst a given set of numbers sequential computation complexity can be O(log n).



- Vector:

→ A vector is a quality that has both magnitude & direction are parallel if they have the same direction.

→ Two vectors are parallel if they are scalar multiplies of one another when you calculate the dot product & your answer is non zero it just means the two vectors are not perpendicular. Two vectors are parallel when they are scalar multiples of each.

Conclusion: Hence we studied parallel reduction using min, max, sum, avg and CUDA program. that

given n element vectors to find min, max, arithmetic values in the vector.

Assignment No: 4

Title: Write a CUDA program for:
1. Addition of two large vector.
2. Matrix multiplication using CUDA.

Objectives:
1. To understand vector and matrix operation.
2. To implement parallel algorithm to perform matrix and vector operations.

Theory:
- While executing the parallel algorithm of matrix vector multiplication, it is necessary to distribute not only the matrix A. but also the vector b and the result vector C.
- If the processor holds the matrix row & all the elements of vector b and C.
- Matrices are widely used in mathematical modelling of various processes, phenomenon and system.
- The efficiency of matrix computation is highly important many standard libraries contain procedure for various matrix multiplication.
- Add two large vectors:

⇒ When added together in this different order, these same three vector still produce a resultant with the same magnitude and direction as before. The order in which vectors are added using the head to tail method insignificant.

⇒ Two vectors adding, subtracting:

e.g. $\begin{bmatrix} A_x & A_y & A_z \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = A_x B_x + A_y B_y + A_z B_z$

- Multiply vector & matrix:

→ e.g.

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 9 \\ 8 \\ 7 \end{bmatrix}$$

$$X = \begin{bmatrix} 1\times9+2\times8+3\times7 \\ 4\times9+5\times9+6\times7 \end{bmatrix} \qquad X = \begin{bmatrix} 46 \\ 118 \end{bmatrix}$$

→ The basic usefulness of matrics is to represent line or transformation of vectors or linear mapping between vector space.

⇒

Conclusion: Hence we have implemented vector & matrix operations to design parallel algorithm.