Assignment No: - HPC-Group1-1

TITLE	Parallel Searching Algorithms
PROBLEM STATEMENT /DEFINITION	Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS .
OBJECTIVE	• To understand concept of Breadth First Search and Depth First Search based on sequential algorithm. • To understand concept of parallel algorithm. • To compare performance by varying number of processors used and also with sequential algorithm.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems 1. Open source Linux or its derivative 2. Master slave parallel computation model
REFERENCES	• Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, "Introduction to Parallel Computing", 2nd edition, Addison-Wesley, 2003, ISBN: 0-201-64865-2.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Related Mathematics 8. Algorithm. 9. Test Cases 10. Conclusion and applications

Assignment No: - HPC-Group1-1

Problem statement: Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS .

- **Aim**
  Write a program to design and implement parallel Breadth First Search and Depth First Search algorithm

- **Prerequisites**
  - Concept of existing sequential algorithms.
  - Concept of High Performance Computing.

- **Learning Objectives**
  - To understand concept of Breadth First Search and Depth First Search based on sequential algorithm.
  - To understand concept of parallel algorithm.
  - To compare performance by varying number of processors used and also with sequential algorithm.

- **Learning Outcomes**
  After successfully completing this assignment, you should be able to
  - Display result for parallel Breadth First Search and Depth First Search
  - Analyze performance by varying number of processors used and also with sequential algorithm.
  - Calculate speedup, efficiency, throughput

- **Concepts related Theory :-**

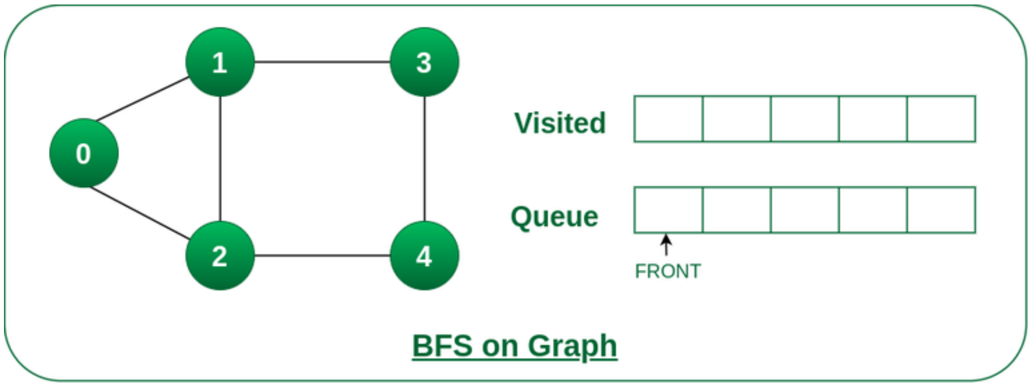
  Breadth First Search (BFS)
  The breadth-first search (BFS) algorithm is used to search a tree or graph data structure for a node that meets a set of criteria. It starts at the tree's root or graph and searches/visits all nodes at the current depth level before moving on to the nodes at the next depth level.

  Follow the below method to implement BFS traversal.

  - Declare a queue and insert the starting vertex.
  - Initialize a **visited** array and mark the starting vertex as visited.
  - Follow the below process till the queue becomes empty:
    - Remove the first vertex of the queue.
    - Mark that vertex as visited.
    - Insert all the unvisited neighbours of the vertex into the queue.

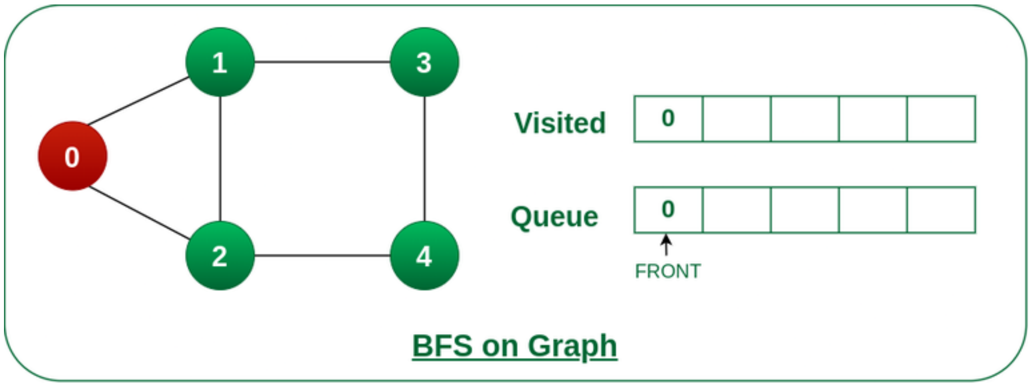
Illustration:

Step1: Initially queue and visited arrays are empty.



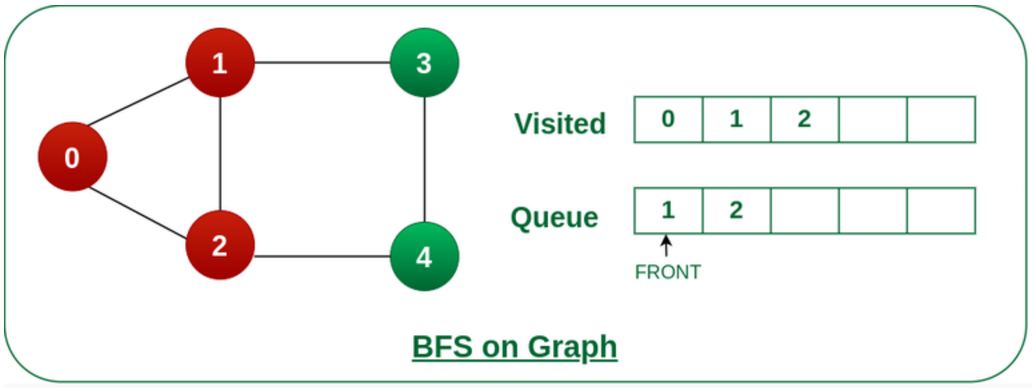
Queue and visited arrays are empty initially.

Step2: Push node 0 into queue and mark it visited.



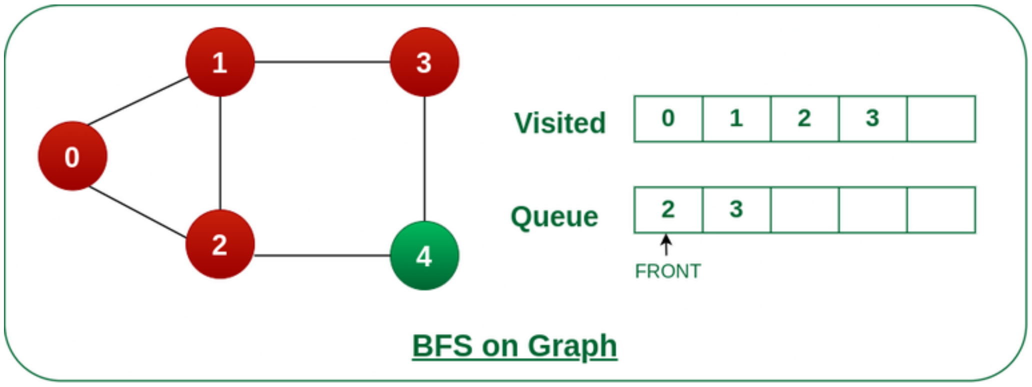
Push node 0 into queue and mark it visited.

Step 3: Remove node 0 from the front of queue and visit the unvisited neighbours and push them into queue.



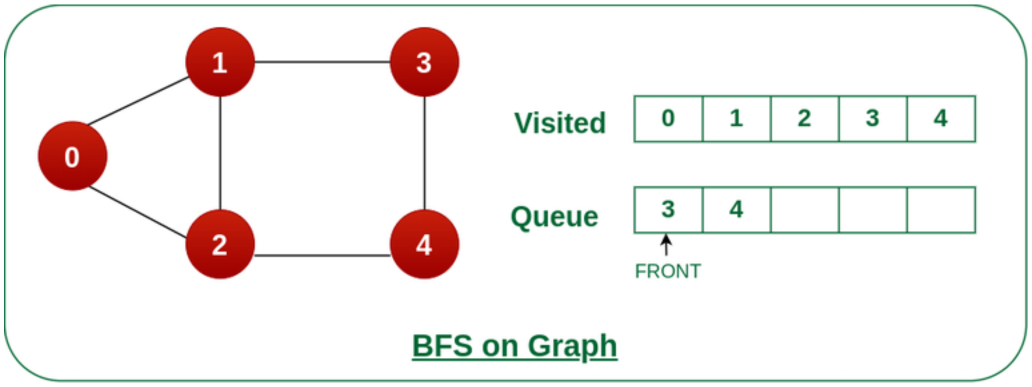
Remove node 0 from the front of queue and visited the unvisited neighbours and push into queue.

Step 4: Remove node 1 from the front of queue and visit the unvisited neighbours and push them into queue.



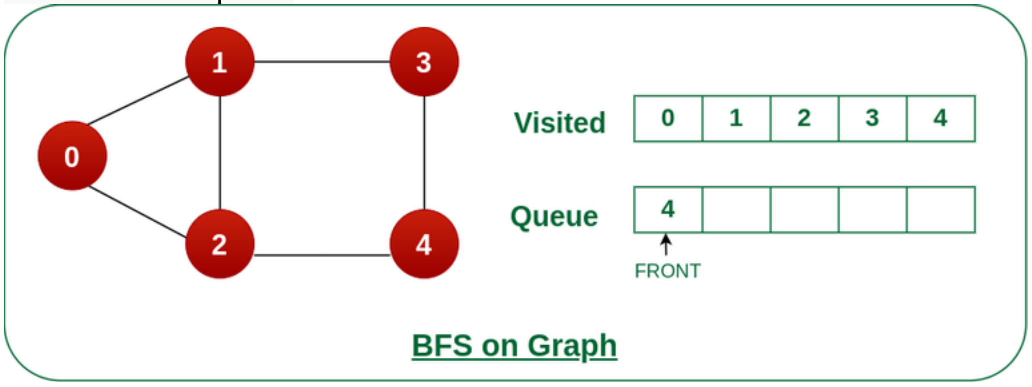
Remove node 1 from the front of queue and visited the unvisited neighbours and push

Step 5: Remove node 2 from the front of queue and visit the unvisited neighbours and push them into queue.



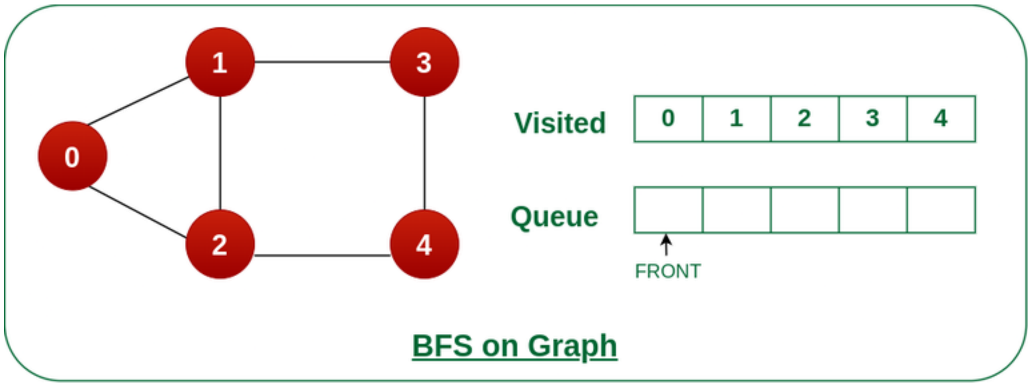
Remove node 2 from the front of queue and visit the unvisited neighbours and push them into queue.

Step 6: Remove node 3 from the front of queue and visit the unvisited neighbours and push them into queue.
As we can see that every neighbours of node 3 is visited, so move to the next node that are in the front of the queue.



Remove node 3 from the front of queue and visit the unvisited neighbours and push them into queue.

Steps 7: Remove node 4 from the front of queue and visit the unvisited neighbours and push them into queue.
As we can see that every neighbours of node 4 are visited, so move to the next node that is in the front of the queue.



Remove node 4 from the front of queue and visit the unvisited neighbours and push them into

queue. Now, Queue becomes empty, So, terminate these process of iteration.

Depth First Search Algorithm

A standard DFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

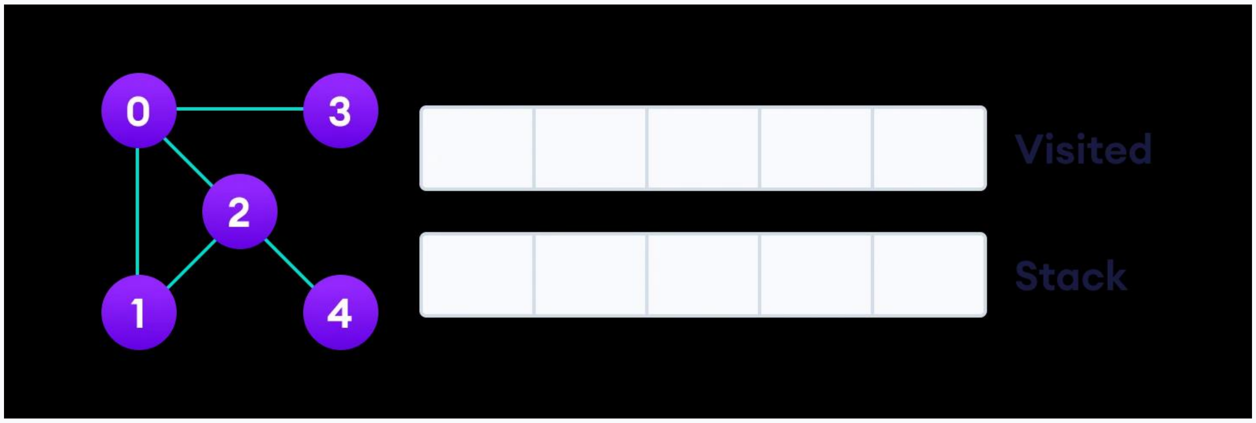
The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.

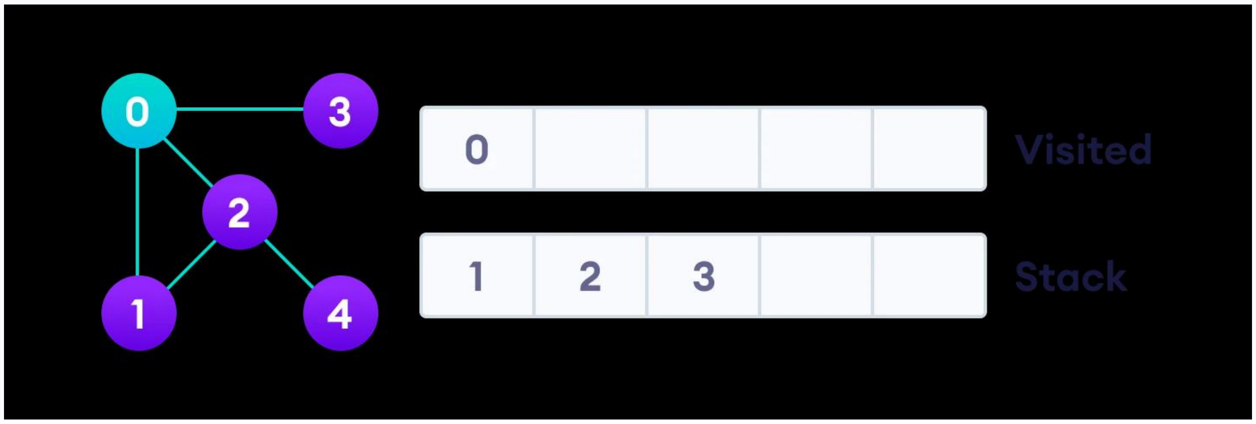
4. Keep repeating steps 2 and 3 until the stack is empty.

Depth First Search Example

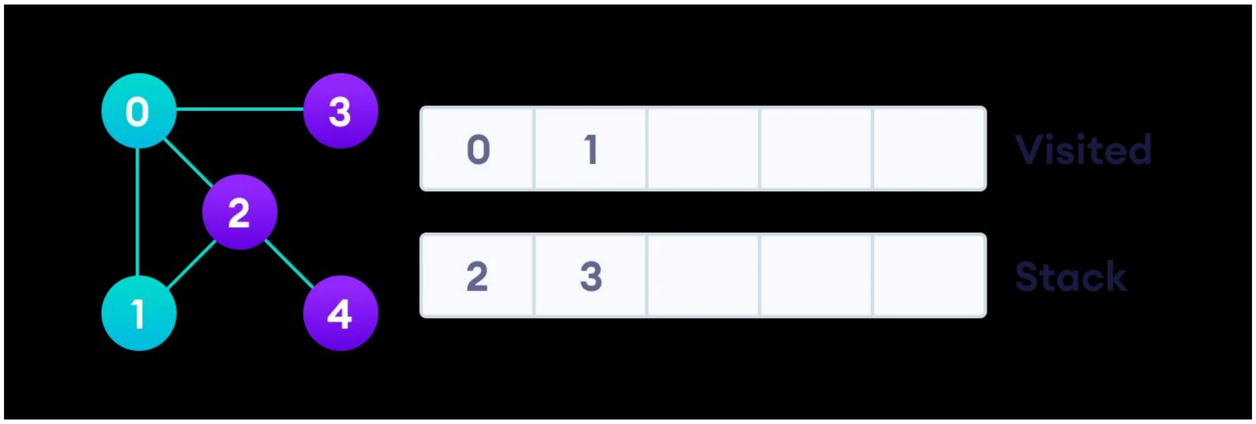
Let's see how the Depth First Search algorithm works with an example. We use an undirected graph with 5 vertices.



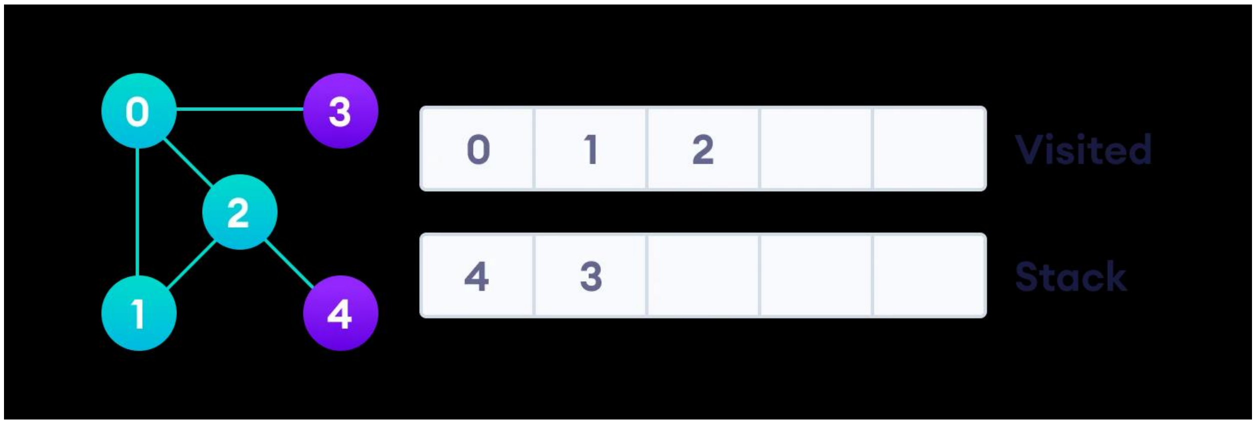
We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



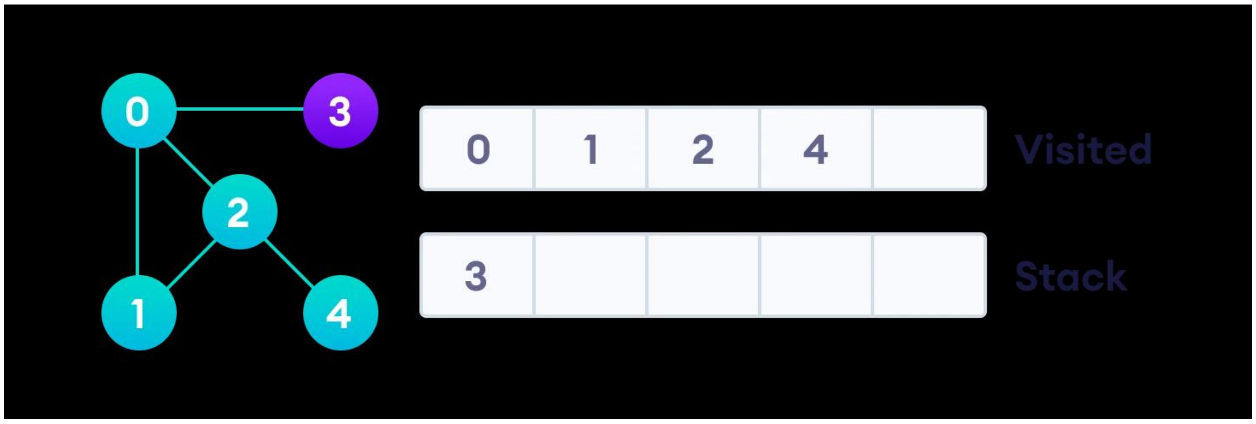
Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.



Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

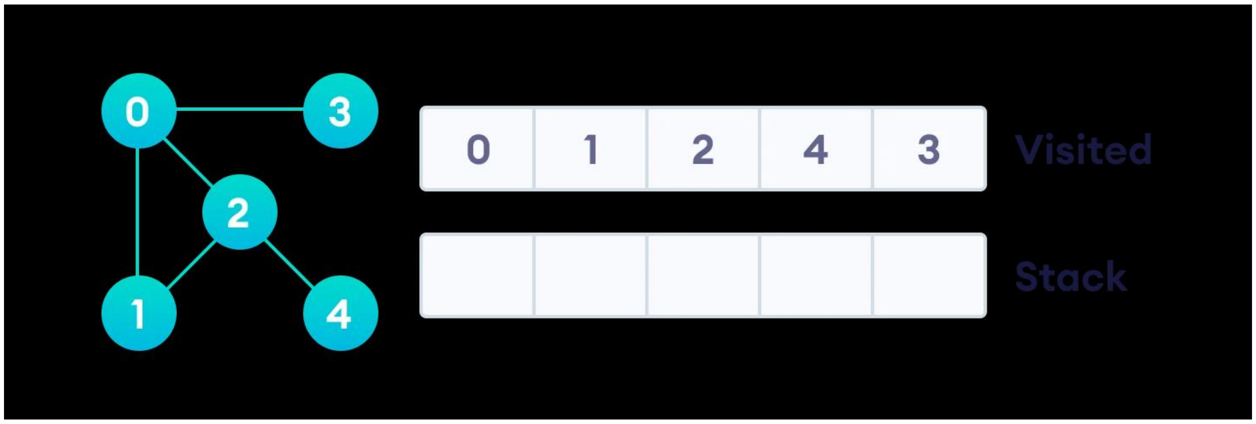


Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.



Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.



After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

Parallel DFS and BFS: Extract parallelism from existing sequential algorithms and implement using OPENMP

Conclusion :

After successfully completing this assignment, student should be able to understand and implement parallel BFS and DFS in OpenMP.