HPC PRACTICALS

Installation commands of g++ and openmp-

```
sudo apt-get install g++
sudo apt-get install libomp.dev
```

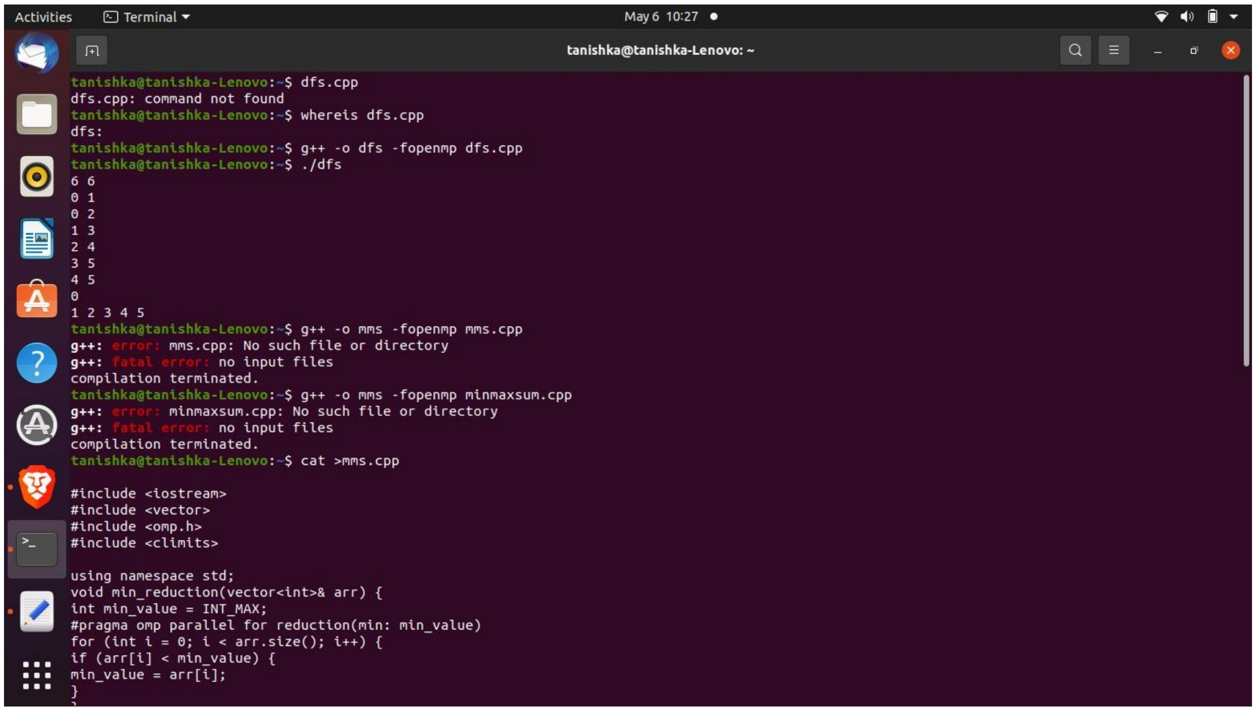
1) DFS and BFS using parallel programming and OpenMP

Steps-

- Create file using command-
  Cat > filename.cpp
- Write code in terminal and press ctrl D to save the code
- To compile file use command-
  g++ -o filename –fopenmp filename.cpp
- Run file using command-
  ./filename

DFS Code-

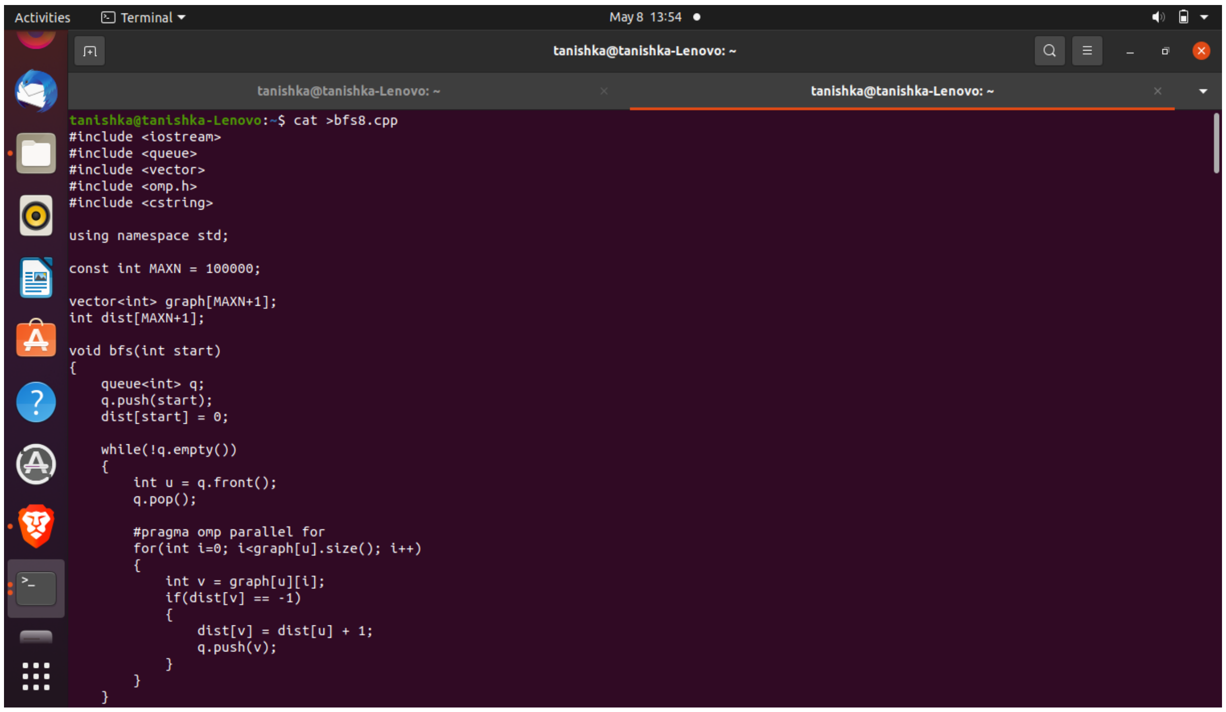
```
#include <iostream>
#include <vector>
#include <omp.h>

using namespace std;

const int MAXN = 1e5;

vector<int> adj[MAXN+5]; // adjacency list
bool visited[MAXN+5]; // mark visited nodes

void dfs(int node) {
    visited[node] = true;
    #pragma omp parallel for
    for (int i = 0; i < adj[node].size(); i++) {
        int next_node = adj[node][i];
        if (!visited[next_node]) {
            dfs(next_node);
        }
    }
}

int main() {
    cout << "Please enter nodes and edges";
    int n, m; // number of nodes and edges
    cin >> n >> m;

    for (int i = 1; i <= m; i++) {
        int u, v; // edge between u and v
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    int start_node; // start node of DFS
    cin >> start_node;

    dfs(start_node);

    // Print visited nodes
    for (int i = 1; i <= n; i++) {
        if (visited[i]) {
            cout << i << " ";
        }
    }
    cout << endl;

    return 0;
}
```



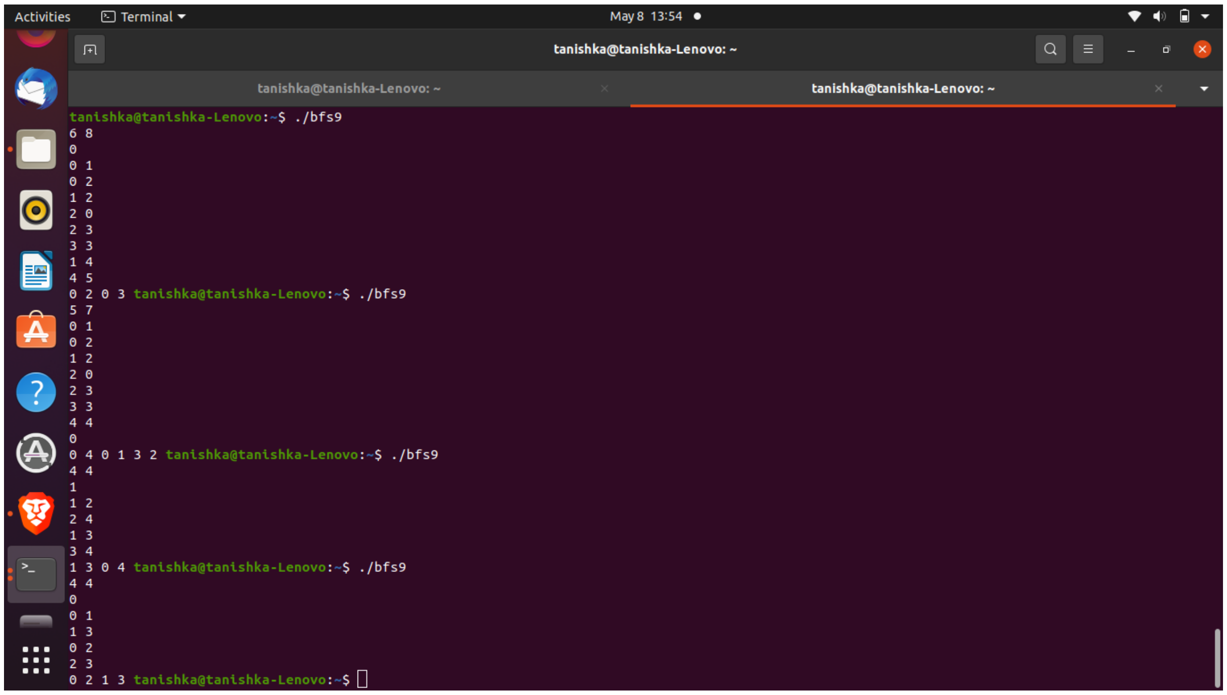
```
tanishka@tanishka-Lenovo:~$ dfs.cpp
dfs.cpp: command not found
tanishka@tanishka-Lenovo:~$ whereis dfs.cpp
dfs:
tanishka@tanishka-Lenovo:~$ g++ -o dfs -fopenmp dfs.cpp
tanishka@tanishka-Lenovo:~$ ./dfs
6 6
0 1
0 2
1 3
2 4
3 5
4 5
0
1 2 3 4 5
tanishka@tanishka-Lenovo:~$ g++ -o mms -fopenmp mms.cpp
g++: error: mms.cpp: No such file or directory
g++: fatal error: no input files
compilation terminated.
tanishka@tanishka-Lenovo:~$ g++ -o mms -fopenmp minmaxsum.cpp
g++: error: minmaxsum.cpp: No such file or directory
g++: fatal error: no input files
compilation terminated.
tanishka@tanishka-Lenovo:~$ cat >mms.cpp

#include <iostream>
#include <vector>
#include <omp.h>
#include <climits>

using namespace std;
void min_reduction(vector<int>& arr) {
int min_value = INT_MAX;
#pragma omp parallel for reduction(min: min_value)
for (int i = 0; i < arr.size(); i++) {
if (arr[i] < min_value) {
min_value = arr[i];
}
```

BFS Code

```
#include <iostream>
#include <queue>
#include <vector>
#include <omp.h>

using namespace std;

int main() {
    int num_vertices, num_edges, source;
    cin >> num_vertices >> num_edges >> source;

    vector<vector<int>> adj_list(num_vertices + 1);
    for (int i = 0; i < num_edges; i++) {
        int u, v;
        cin >> u >> v;
        adj_list[u].push_back(v);
        adj_list[v].push_back(u);
    }
    queue<int> q;
    vector<bool> visited(num_vertices + 1, false);
    q.push(source);
    visited[source] = true;

    while (!q.empty()) {
        int curr_vertex = q.front();
        q.pop();

        cout << curr_vertex << " ";
        #pragma omp parallel for shared(adj_list, visited, q) schedule(dynamic)
        for (int i = 0; i < adj_list[curr_vertex].size(); i++) {
            int neighbour = adj_list[curr_vertex][i];
            if (!visited[neighbour]) {
                visited[neighbour] = true;
                q.push(neighbour);
            }
        }    }

    return 0;
}
```


Activities
Terminal
May 8 13:54
tanishka@tanishka-Lenovo: ~
tanishka@tanishka-Lenovo: ~
tanishka@tanishka-Lenovo: ~

```
tanishka@tanishka-Lenovo:~$ cat >bfs8.cpp
#include <iostream>
#include <queue>
#include <vector>
#include <omp.h>
#include <cstring>

using namespace std;

const int MAXN = 100000;

vector<int> graph[MAXN+1];
int dist[MAXN+1];

void bfs(int start)
{
    queue<int> q;
    q.push(start);
    dist[start] = 0;

    while(!q.empty())
    {
        int u = q.front();
        q.pop();

        #pragma omp parallel for
        for(int i=0; i<graph[u].size(); i++)
        {
            int v = graph[u][i];
            if(dist[v] == -1)
            {
                dist[v] = dist[u] + 1;
                q.push(v);
            }
        }
    }
```


Activities
Terminal
May 8 13:54
tanishka@tanishka-Lenovo: ~
tanishka@tanishka-Lenovo: ~
tanishka@tanishka-Lenovo: ~

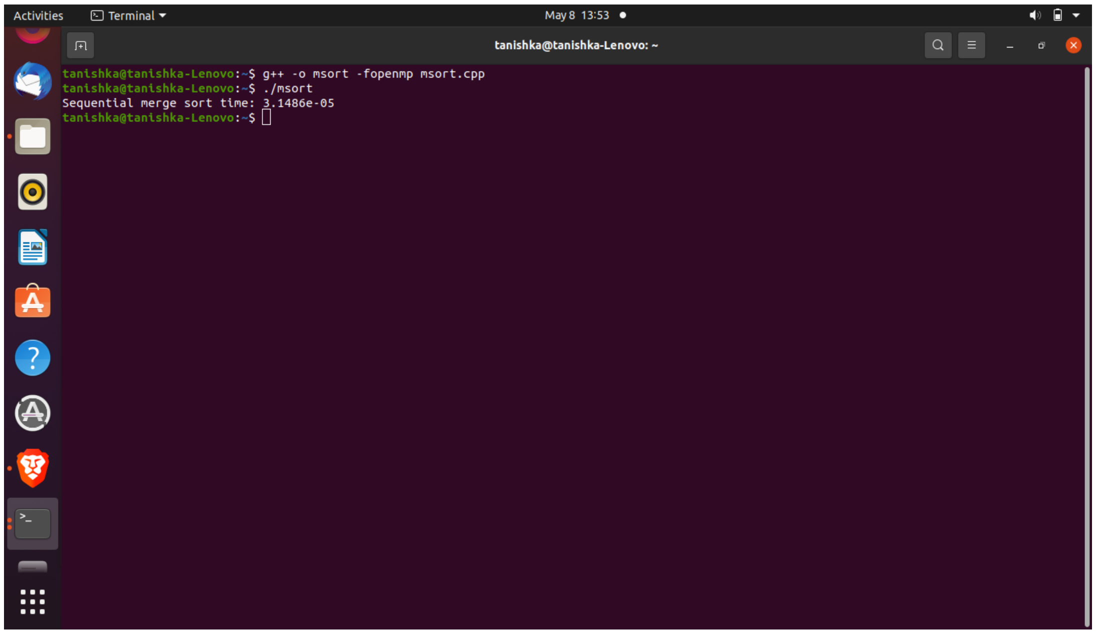
```
tanishka@tanishka-Lenovo:~$ ./bfs9
6 8
0
0 1
0 2
1 2
2 0
2 3
3 3
1 4
4 5
0 2 0 3 tanishka@tanishka-Lenovo:~$ ./bfs9
5 7
0 1
0 2
1 2
2 0
2 3
3 3
4 4
0
0 4 0 1 3 2 tanishka@tanishka-Lenovo:~$ ./bfs9
4 4
1
1 2
2 4
1 3
3 4
1 3 0 4 tanishka@tanishka-Lenovo:~$ ./bfs9
4 4
0
0 1
1 3
0 2
2 3
0 2 1 3 tanishka@tanishka-Lenovo:~$ 
```

2) Merge sort and bubble sort using parallel programming and OpenMP (use steps given in practical 1)

MergeSort Code-

```
#include <iostream>
#include <vector>
#include <omp.h>
using namespace std;
void merge(vector<int>& arr, int l, int m, int r) {
        int i, j, k;
        int n1 = m - l + 1;
        int n2 = r - m;
        vector<int> L(n1), R(n2);
        for (i = 0; i < n1; i++) {
                L[i] = arr[l + i];
        }
        for (j = 0; j < n2; j++) {
                R[j] = arr[m + 1 + j];
        }
        i = 0;
        j = 0;
        k = l;
        while (i < n1 && j < n2) {
                if (L[i] <= R[j]) {
                        arr[k++] = L[i++];
                } else {
                        arr[k++] = R[j++];
                }
        }
}
void merge_sort(vector<int>& arr, int l, int r) {
        if (l < r) {
                int m = l + (r - l) / 2;
#pragma omp task
                merge_sort(arr, l, m);
#pragma omp task
                merge_sort(arr, m + 1, r);
                merge(arr, l, m, r);
        }
}
```

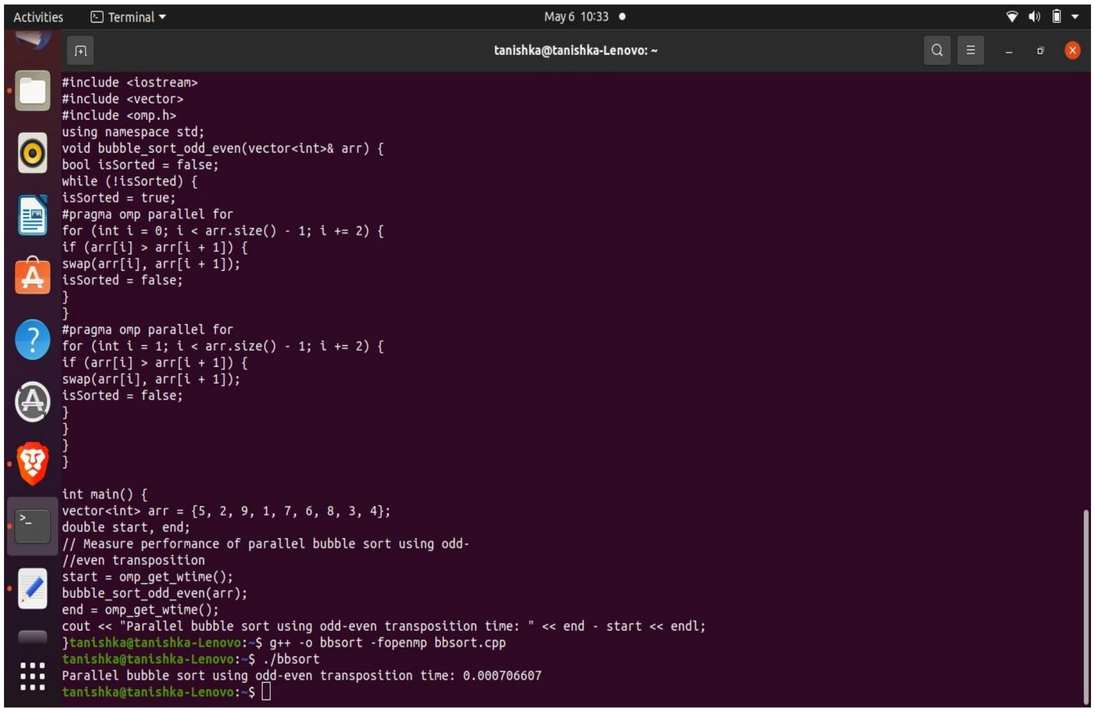
```
void parallel_merge_sort(vector<int>& arr) {
#pragma omp parallel
        {
#pragma omp single
                merge_sort(arr, 0, arr.size() - 1);
        }
}
int main() {
        vector<int> arr = {5, 2, 9, 1, 7, 6, 8, 3, 4};
        double start, end;
        // Measure performance of sequential merge sort
        start = omp_get_wtime();
        merge_sort(arr, 0, arr.size() - 1);
        end = omp_get_wtime();
        cout << "Sequential merge sort time: " << end - start <<endl;
        // Measure performance of parallel merge sort
        arr = {5, 2, 9, 1, 7, 6, 8, 3, 4};
        start = omp_get_wtime();
        parallel_merge_sort(arr);
        end = omp_get_wtime();
        return 0;
        }
```



```
tanishka@tanishka-Lenovo:~$ g++ -o msort -fopenmp msort.cpp
tanishka@tanishka-Lenovo:~$ ./msort
Sequential merge sort time: 3.1486e-05
tanishka@tanishka-Lenovo:~$ 
```

Bubble Sort Code-

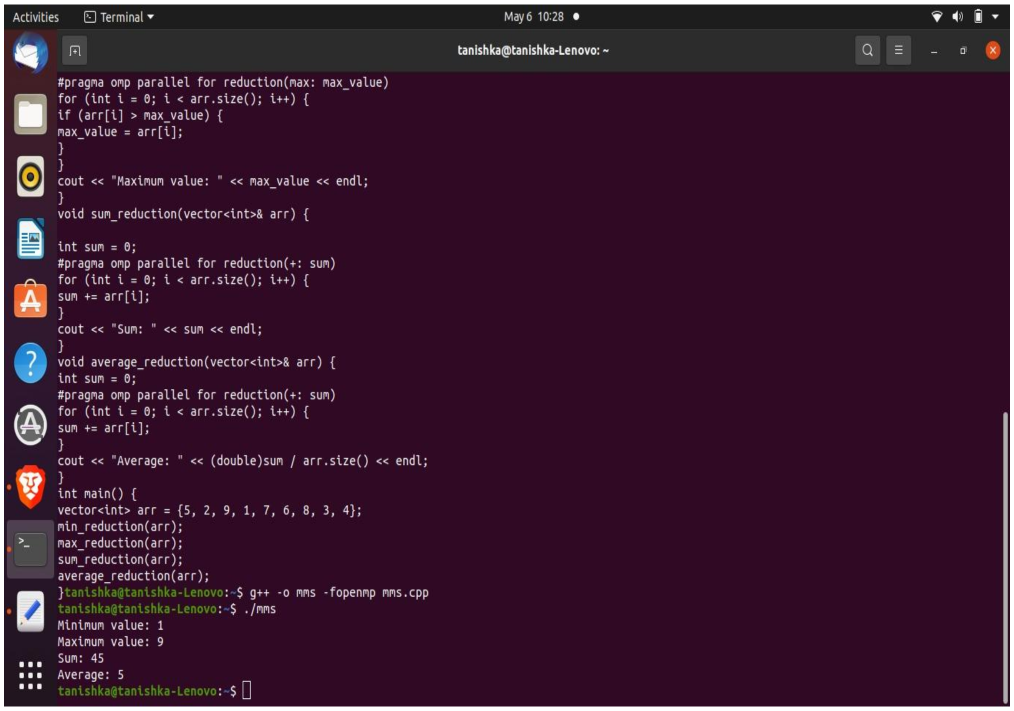
```
#include <iostream>
#include <vector>
#include <omp.h>
using namespace std;
void bubble_sort_odd_even(vector<int>& arr) {
bool isSorted = false;
while (!isSorted) {
isSorted = true;
#pragma omp parallel for
for (int i = 0; i < arr.size() - 1; i += 2) {
if (arr[i] > arr[i + 1]) {
swap(arr[i], arr[i + 1]);
isSorted = false;
}
}
#pragma omp parallel for
for (int i = 1; i < arr.size() - 1; i += 2) {
if (arr[i] > arr[i + 1]) {
swap(arr[i], arr[i + 1]);
isSorted = false;
}
}
}
}

int main() {
vector<int> arr = {5, 2, 9, 1, 7, 6, 8, 3, 4};
double start, end;
// Measure performance of parallel bubble sort using odd-
//even transposition
start = omp_get_wtime();
bubble_sort_odd_even(arr);
end = omp_get_wtime();
cout << "Parallel bubble sort using odd-even transposition time: " << end - start << endl;
}
```



```
#include <iostream>
#include <vector>
#include <omp.h>
using namespace std;
void bubble_sort_odd_even(vector<int>& arr) {
bool isSorted = false;
while (!isSorted) {
isSorted = true;
#pragma omp parallel for
for (int i = 0; i < arr.size() - 1; i += 2) {
if (arr[i] > arr[i + 1]) {
swap(arr[i], arr[i + 1]);
isSorted = false;
}
}
#pragma omp parallel for
for (int i = 1; i < arr.size() - 1; i += 2) {
if (arr[i] > arr[i + 1]) {
swap(arr[i], arr[i + 1]);
isSorted = false;
}
}
}
}

int main() {
vector<int> arr = {5, 2, 9, 1, 7, 6, 8, 3, 4};
double start, end;
// Measure performance of parallel bubble sort using odd-
//even transposition
start = omp_get_wtime();
bubble_sort_odd_even(arr);
end = omp_get_wtime();
cout << "Parallel bubble sort using odd-even transposition time: " << end - start << endl;
}tanishka@tanishka-Lenovo:~$ g++ -o bbsort -fopenmp bbsort.cpp
tanishka@tanishka-Lenovo:~$ ./bbsort
Parallel bubble sort using odd-even transposition time: 0.000706607
tanishka@tanishka-Lenovo:~$
```

3) Min, Max, Sum and Avg using parallel reduction
Code-

```
#include <iostream>
#include <vector>
#include <omp.h>
#include <climits>

using namespace std;
void min_reduction(vector<int>& arr) {
int min_value = INT_MAX;
#pragma omp parallel for reduction(min: min_value)
for (int i = 0; i < arr.size(); i++) {
if (arr[i] < min_value) {
min_value = arr[i];
}
}
cout << "Minimum value: " << min_value << endl;
}
void max_reduction(vector<int>& arr) {
int max_value = INT_MIN;
#pragma omp parallel for reduction(max: max_value)
for (int i = 0; i < arr.size(); i++) {
if (arr[i] > max_value) {
max_value = arr[i];
}
}
cout << "Maximum value: " << max_value << endl;
}
void sum_reduction(vector<int>& arr) {

int sum = 0;
#pragma omp parallel for reduction(+: sum)
for (int i = 0; i < arr.size(); i++) {
sum += arr[i];
}
cout << "Sum: " << sum << endl;
}
void average_reduction(vector<int>& arr) {
int sum = 0;
#pragma omp parallel for reduction(+: sum)
for (int i = 0; i < arr.size(); i++) {
sum += arr[i];
}
cout << "Average: " << (double)sum / arr.size() << endl;
}
int main() {
vector<int> arr = {5, 2, 9, 1, 7, 6, 8, 3, 4};
min_reduction(arr);
max_reduction(arr);
sum_reduction(arr);
average_reduction(arr);
}
```


Activities   Terminal   May 6 10:28

tanishka@tanishka-Lenovo: ~

```
#pragma omp parallel for reduction(max: max_value)
for (int i = 0; i < arr.size(); i++) {
if (arr[i] > max_value) {
max_value = arr[i];
}
}
cout << "Maximum value: " << max_value << endl;
}
void sum_reduction(vector<int>& arr) {

int sum = 0;
#pragma omp parallel for reduction(+: sum)
for (int i = 0; i < arr.size(); i++) {
sum += arr[i];
}
cout << "Sum: " << sum << endl;
}
void average_reduction(vector<int>& arr) {
int sum = 0;
#pragma omp parallel for reduction(+: sum)
for (int i = 0; i < arr.size(); i++) {
sum += arr[i];
}
cout << "Average: " << (double)sum / arr.size() << endl;
}
int main() {
vector<int> arr = {5, 2, 9, 1, 7, 6, 8, 3, 4};
min_reduction(arr);
max_reduction(arr);
sum_reduction(arr);
average_reduction(arr);
}tanishka@tanishka-Lenovo:~$ g++ -o mms -fopenmp mms.cpp
tanishka@tanishka-Lenovo:~$ ./mms
Minimum value: 1
Maximum value: 9
Sum: 45
Average: 5
tanishka@tanishka-Lenovo:~$
```