```
PS D:\C++> g++ -fopenmp parallel_dfs.cpp
PS D:\C++> ./a out
```

HPC 1 Final:-

```
#include<iostream>
#include<vector>
#include<omp.h>
#include<queue>
#include<bits/stdc++.h>
using namespace std;
queue<int>q;
void bfs(int start, int* arr, int n, int visit[])
{
    #pragma omp parallel for ordered
    for(int i=0; i<n; i++)
    {
        #pragma omp ordered
        if( ( *(arr + (n*start) + i)  == 1 ) && (visit[i] == 0) )
        {
            cout<<i<<" ";
            q.push(i);
            visit[i] = 1;
        }
    }
    q.pop();
    if(!q.empty()) bfs(q.front(), (int*)arr, n, visit);
}
void dfs(int start, int* arr, int n, int visited[]) {
    //#pragma omp parallel for ordered
    for(int i = 0; i < n; i++) {
        //#pragma omp ordered
        if( (*(arr + (start*n) + i) == 1) && (!visited[i]) )
        {
            visited[i] = 1;
            cout<<i<<" ";
            dfs(i, (int*)arr, n, visited);
        }
    }
}
int main()
{
    //freopen("input.txt","r",stdin);
    //freopen("output.txt","w",stdout);
    //cout<<"BFS 0 1 2 3 4 5 6"<<endl;
    cout<<"Enter the number of vertices: ";
    int n;
    cin>>n;
```

```
    int arr[n][n] = {0};
    cout<<"Enter the number of edges: ";
    int edges;
    cin>>edges;
    for(int j=0; j<edges; j++)
    {
        int a, b;
        cout<<"Enter the two edges:"<<endl;
        cin>>a>>b;
        arr[a][b] = 1;
        arr[b][a] = 1;
    }
    int visit[n] = {0};
    int visited[n] = {0};
    cout<<"Enter the start vertex: ";
    int start;
    cin>>start;
    clock_t strt = clock();
    visit[start] = 1;
    cout<<start<<" ";
    q.push(start);
    cout<<"\nBfs is:";
    bfs(start, (int*)arr, n, visit);
    cout<<"\nDfs is:";
    dfs(start, (int *)arr, n, visited);
    clock_t stop = clock();
    cout<<"\nTime required : "<<(double)(stop-strt)<<" ms"<<endl;
    return 0;
}
```

HPC 2 Final:-

```
#include<iostream>
#include<cstdlib>
#include<omp.h>
#include<time.h>
using namespace std;
void merge(int array[],int low1, int high1,int low2,int high2, int n)
{
      int temp[n];
      int i=low1,j=low2,k=0;
      while(i<=high1 && j<=high2)
      {
            if(array[i]<array[j])
                  temp[k++]=array[i++];
            else
                  temp[k++]=array[j++];
      }
      while(i<=high1)
            temp[k++]=array[i++];
      while(j<=high2)
            temp[k++]=array[j++];
      for(i=low1,j=0;i<=high2;i++,j++)
            array[i]=temp[j];
}
void mergesort(int array[], int low, int high, int n)
{
      if(low<high)
      {
            int mid=(low+high)/2;
            #pragma omp parallel sections
            {
                  #pragma omp section
                  {
                        mergesort(array,low,mid,n);
                  }
                  #pragma omp section
                  {
                        mergesort(array,mid+1,high,n);
                  }
            }
         merge(array,low,mid,mid+1,high,n);
      }
}
void display(int array[], int n)
{
```

```
      for(int i=0;i<n;i++) cout<<array[i]<<" ";
}
int main()
{
      int n;
      cout<<"Enter the number of elements : ";
      cin>>n;
     int array[n] = {0};
      for(int i=0;i<n;i++)
      {
            array[i]=rand()%32;
      }
      cout<<"Original Array: ";
      display(array,n);
      cout<<endl;
      clock_t start = clock();
      mergesort(array,0,n-1,n);
      clock_t stop = clock();
      cout<<"Final Array: ";
      display(array,n);
      cout<<endl;
      cout<<"Time required :
"<<(double)(stop-start)*1000/CLOCKS_PER_SEC<<" ms";
     return 0;
}
```

Bubble Sort

```
#include<omp.h>
#include<iostream>
#include<time.h>
using namespace std;
int main()
{
      // freopen("input.txt","r",stdin);
     // freopen("output.txt","w",stdout);
      int n;
      cout<<"Enter the number of elements : ";
      cin>>n;
      cout<<endl;
      int array[n] = {0};
      for(int i=0;i<n;i++) array[i]=rand()%32;
      cout<<"Original Array: ";
      for(int i=0; i<n; i++) cout<<array[i]<<" ";
      cout<<endl;
      clock_t start=clock();
      int var = 0;
      for(int i=0; i<n; i++)
      {
            #pragma omp parallel for
            for(int j=var; j<n-1; j+=2)
            {
                  if(array[j] > array[j+1])
                  {
                        int temp = array[j];
                        array[j] = array[j+1];
                        array[j+1] = temp;
                  }
            }
            if(var == 0) var = 1;
            else var = 0;
      }
      clock_t stop=clock();
      cout<<"\nFinal Array: ";
      for(int i=0; i<n; i++) cout<<array[i]<<" ";
      cout<<endl;
      cout<<"\nTime required : "<<(double)(stop-start)<<" ms"<<endl;
      return 0;
}
```

HPC 3 Final:-

```
#include <iostream>
#include <vector>
#include <numeric>
#include <omp.h>
using namespace std;
int main() {
int n;
cout << "Enter the size of the array: ";
cin >> n;
vector<int> arr(n);
cout << "Enter the elements of the array:\n";
for (int i = 0; i < n; ++i) {
cin >> arr[i];
}
int min_val = arr[0], max_val = arr[0], sum = 0;
#pragma omp parallel for reduction(min:min_val)
reduction(max:max_val) reduction(+:sum)
for (int i = 0; i < n; ++i) {
if (arr[i] < min_val) min_val = arr[i];
if (arr[i] > max_val) max_val = arr[i];
sum += arr[i];
}
double average = static_cast<double>(sum) / n;
cout << "Min value: " << min_val << endl;
cout << "Max value: " << max_val << endl;
cout << "Sum: " << sum << endl;
cout << "Average: " << average << endl;
return 0;
}
```