HPC-2A(Bubble Sort)

#include <iostream>

#include <vector>

#include <omp.h>

#include <cstdlib>

Using namespace std;

// Sequential Bubble Sort

Void sequentialBubbleSort(vector<int>& arr) {

Int n = arr.size();

For (int I = 0; I < n-1; ++i) {

For (int j = 0; j < n-i-1; ++j) {

If (arr[j] > arr[j+1]) {

Swap(arr[j], arr[j+1]);

}

}

}

}

// Parallel Bubble Sort using OpenMP (Odd-Even)

Void parallelBubbleSort(vector<int>& arr) {

Int n = arr.size();

For (int I = 0; I < n; i++) {

// Odd phase

#pragma omp parallel for

For (int j = 1; j < n

• 1; j += 2) {

If (arr[j] > arr[j + 1]) {

Swap(arr[j], arr[j + 1]);

}

}

// Even phase

#pragma omp parallel for

For (int j = 0; j < n

• 1; j += 2) {

If (arr[j] > arr[j + 1]) {

Swap(arr[j], arr[j + 1]);

}

}

}

}

Int main() {

Int n;

Cout << “Enter number of elements: “;

Cin >> n;

Vector<int> original(n);

Cout << “Enter “ << n << “ elements:\n”;

For (int I = 0; I < n; ++i)

Cin >> original[i];

Vector<int> seqArr = original;

Vector<int> parArr = original;

Double startSeq = omp\_get\_wtime();

sequentialBubbleSort(seqArr);

double endSeq = omp\_get\_wtime();

double startPar = omp\_get\_wtime();

parallelBubbleSort(parArr);

double endPar = omp\_get\_wtime();

cout << “\nTime taken by Sequential Bubble Sort: “ << (endSeq

• startSeq) << “ seconds\n”;

Cout << “Time taken by Parallel Bubble Sort : “ << (endPar • startPar) << “ seconds\n”;

Return 0;

}