BUBBLE SORT IN PARELLEL

```
code = """
#include<omp.h>
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

void swap(int *num1, int *num2) {
int temp = *num1;
*num1 = *num2;
*num2 = temp;
}

int main() {
int n = 10;
int a[10];
omp_set_num_threads(2);
for(int i=0; i<n; i++) {
a[i] = rand()% 100;
}
for(int i=0; i<n; i++)
cout<<" "<<a[i];
cout<<endl;
int i=0, j=0;
int first=0;
double start, end;
start = omp_get_wtime();
for(i=0; i<n-1; i++) {
first = i%2;
#pragma omp parallel for
for(j=first; j<n-1; j++) {
if(a[j] > a[j+1])
swap(&a[j], &a[j+1]);
}
}
end = omp_get_wtime();
cout<<"Result(parallel) : "<<endl;
for(i=0; i<n; i++)
cout<<" "<<a[i];
cout<<endl;
cout<<"Time parallel = "<<(end-start)<<endl;
return 0;
}
"""
text_file = open("/content/code.cpp", "w")
text_file.write(code)
text_file.close()
!g++ -fopenmp code.cpp
!./a.out
```

OUTPUT:::

```
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23 67 35 29 2
22 58 69 67 93 56 11 42 29 73 21 19 84 37 98 24 15 70 13 26 91 80 56 73 62 70 96 81 5 25 84 27

Result(parallel) : 2 5 11 11 13 15 15 19 21 21 22 23 24 25 26 26 26 27 27 29 29 29 30 35 35 36
37 40 42 49 56 56 58 59 62 62 62 63 67 67 67 68 69 70 70 72 73 73 77 80 81 82 83 84 84 86 86 90
91 92 93 93 96 98

Time parallel = 0.000183272
```

MERGE SORT IN PARELLEL

```
code = """
#include<iostream>
#include<omp.h>

using namespace std;

void printArray(int *arr, int size) {
for(int i=0; i<size; i++) {
cout<<arr[i]<<" ";
}
cout<<endl;
}

void merge(int* arr, int start, int mid, int end) {
int len = (end - start) + 1;
int temp[len];
int cur = 0;
int i = start;
int j = mid + 1;
while(i <= mid && j <= end){
if(arr[i] < arr[j]) {
temp[cur] = arr[i];
cur++;
i++;
}
else {
temp[cur] = arr[j];
cur++;
j++;
}
}
if(i <= mid) {
while(i <= mid) {
temp[cur] = arr[i];
i++;
cur++;
}
}
else if(j <= end) {
while(j <= end) {
temp[cur] = arr[j];
j++;
cur++;
}
}
cur = 0;
for(i=start; i<=end; i++) {
arr[i] = temp[cur];
cur++;
}

}

void mergeSort(int *arr, int start, int end) {
if(start < end) {
int mid = (start+end) / 2;
#pragma omp parallel sections
```

```
{
#pragma omp section
mergeSort(arr, start, mid);
#pragma omp section
mergeSort(arr, mid+1, end);
}
merge(arr, start, mid, end);
}
}

int main(int argc, char *argv[]) {
int size = 64;
int a[size];
double start, end;
omp_set_num_threads(2);
for(int i=0; i<size; i++) {
a[i] = rand()% 100;
}
//int a[]= {7,33,5,5,23,111,75,34,77,121,120};
for(int i=0; i<size; i++)
cout<<" "<<a[i];
cout<<endl;
start = omp_get_wtime();
mergeSort(a, 0, size-1);
printArray(a, size);
end = omp_get_wtime();
cout<<"Time parallel = "<<(end-start)<<endl;
return 0;
}

"""



text_file = open("/content/code.cpp", "w")
text_file.write(code)
text_file.close()
!g++ -fopenmp code.cpp
!./a.out
```

OUTPUT:::

```
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23 67 35 29 2
22 58 69 67 93 56 11 42 29 73 21 19 84 37 98 24 15 70 13 26 91 80 56 73 62 70 96 81 5 25 84 27

Result::: 2 5 11 11 13 15 15 19 21 21 22 23 24 25 26 26 26 27 27 29 29 29 30 35 35 36 37 40 42
49 56 56 58 59 62 62 62 63 67 67 67 68 69 70 70 72 73 73 77 80 81 82 83 84 84 86 86 90 91 92 93
93 96 98

Time parallel = 0.00489823
```