**K. K. Wagh Institute of Engineering Education and Research, Nashik.**

**Department of Computer Engineering**

**Academic Year 2022-23**

**Course: Laboratory Practice III Course Code: 410246**

**PID: 27 Class: BE -B**

**Group Members: 1. Abhishek Ahire (BE-B-21)**

**2. Vaishnavi kale (BE-B-22)**

**3. Atharva khaire (BE-B-23)**

**4. Eishita Pawar(BE-B-24)**

**DAA miniproject**

**Problem statement :** Implement merge sort and multithreaded merge sort. Compare time required by both the algorithms. Also analyze the performance of each algorithm for the best case and the worst case.

**Objective:** To implement merge sort and merge sort using multithreading

**Merge sort:**

The Merge Sort algorithm is a sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner.

Algorithm:

step 1: start

step 2: declare array and left, right, mid variable

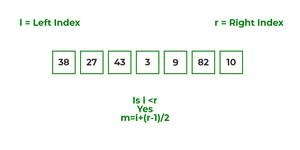
step 3: perform merge function.  
    if left > right  
        return  
    mid= (left+right)/2  
    mergesort(array, left, mid)  
    mergesort(array, mid+1, right)  
    merge(array, left, mid, right)

step 4: Stop

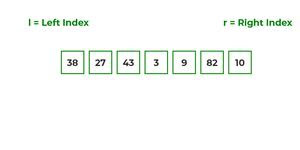
Illustration:

To know the functioning of merge sort, lets consider an array arr[] = {38, 27, 43, 3, 9, 82, 10}

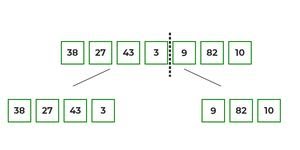
1. At first, check if the left index of array is less than the right index, if yes then calculate its mid point



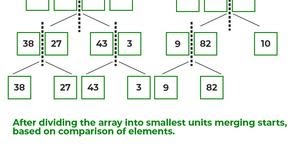
1. Now, as we already know that merge sort first divides the whole array iteratively into equal halves, unless the atomic values are achieved.
2. Here, we see that an array of 7 items is divided into two arrays of size 4 and 3 respectively.



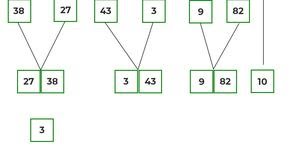
1. Now, again find that is left index is less than the right index for both arrays, if found yes, then again calculate mid points for both the arrays.



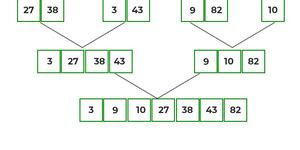
1. Now, further divide these two arrays into further halves, until the atomic units of the array is reached and further division is not possible.



1. After dividing the array into smallest units, start merging the elements again based on comparison of size of elements
2. Firstly, compare the element for each list and then combine them into another list in a sorted manner.



1. After the final merging, the list looks like this:



Time complexities:

|  |  |
| --- | --- |
| Best Case | O(n\*logn) |
| Average Case | O(n\*logn) |
| Worst Case | O(n\*logn) |

Space complexity: O(n)

**Merge sort using multithreading:**

Multi-threading is way to improve parallelism by running the threads simultaneously in different cores of your processor. In this program, we’ll use 4 threads but you may change it according to the number of cores your processor has

**Code**

Merge sort

#include <iostream>

using namespace std;

void merge(int arr[], int p, int q, int r) {

int n1 = q - p + 1;

int n2 = r – q;

int L[n1], M[n2];

for (int i = 0; i < n1; i++)

L[i] = arr[p + i];

for (int j = 0; j < n2; j++)

M[j] = arr[q + 1 + j];

// Maintain current index of sub-arrays and main array

int i, j, k;

i = 0;

j = 0;

k = p;

while (i < n1 && j < n2) {

if (L[i] <= M[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = M[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = M[j];

j++;

k++;

}

}

// Divide the array into two subarrays, sort them and merge them

void mergeSort(int arr[], int l, int r) {

if (l < r) {

// m is the point where the array is divided into two subarrays

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

// Merge the sorted subarrays

merge(arr, l, m, r);

}

}

// Print the array

void printArray(int arr[], int size) {

for (int i = 0; i < size; i++)

cout << arr[i] << " ";

cout << endl;

}

// Driver program

int main() {

int arr[] = {6, 5, 12, 10, 9, 1};

int size = sizeof(arr) / sizeof(arr[0]);

mergeSort(arr, 0, size - 1);

cout << "Sorted array: \n";

printArray(arr, size);

return 0;

}

**Output**

Sorted array:

1 5 6 9 10 12

Merge sort using multithreading

#include <iostream>

#include <pthread.h>

#include <time.h>

#define size 20

#define thread\_size 4

using namespace std;

int arr[size];

int temp\_val = 0;

void combine\_array(int first, int mid\_val, int end){

int\* start = new int[mid\_val - first + 1];

int\* last = new int[end - mid\_val];

int temp\_1 = mid\_val - first + 1;

int temp\_2 = end - mid\_val;

int i, j;

int k = first;

for(i = 0; i < temp\_1; i++){

start[i] = arr[i + first];

}

for (i = 0; i < temp\_2; i++){

last[i] = arr[i + mid\_val + 1];

}

i = j = 0;

while(i < temp\_1 && j < temp\_2){

if(start[i] <= last[j]){

arr[k++] = start[i++];

}

else{

arr[k++] = last[j++];

}

}

while (i < temp\_1){

arr[k++] = start[i++];

}

while (j < temp\_2){

arr[k++] = last[j++];

}

}

void Sorting\_Threading(int first, int end){

int mid\_val = first + (end - first) / 2;

if(first < end){

Sorting\_Threading(first, mid\_val);

Sorting\_Threading(mid\_val + 1, end);

combine\_array(first, mid\_val, end);

}

}

void\* Sorting\_Threading(void\* arg){

int set\_val = temp\_val++;

int first = set\_val \* (size / 4);

int end = (set\_val + 1) \* (size / 4) - 1;

int mid\_val = first + (end - first) / 2;

if (first < end){

Sorting\_Threading(first, mid\_val);

Sorting\_Threading(mid\_val + 1, end);

combine\_array(first, mid\_val, end);

}

}

int main(){

for(int i = 0; i < size; i++){

arr[i] = rand() % 100;

}

pthread\_t P\_TH[thread\_size];

for(int i = 0; i < thread\_size; i++){

pthread\_create(&P\_TH[i], NULL, Sorting\_Threading, (void\*)NULL);

}

for(int i = 0; i < 4; i++){

pthread\_join(P\_TH[i], NULL);

}

combine\_array(0, (size / 2 - 1) / 2, size / 2 - 1);

combine\_array(size / 2, size/2 + (size-1-size/2)/2, size - 1);

combine\_array(0, (size - 1)/2, size - 1);

cout<<"Merge Sort using Multi-threading: ";

for (int i = 0; i < size; i++){

cout << arr[i] << " ";

}

return 0;

}

**Output**

Merge Sort using Multi-threading: 15 21 26 26 27 35 36 40 49 59 62 63 72 77 83 86 86 90 92 93

**Conclusion:**

Hence ,implemented Merge sort and merge sort using multithreading.