Pune Institute of Computer Technology

Department of Computer Engineering
(2022- 2023)

Laboratory Practice III – 410246

Design and Analysis of Algorithm

Submitted by: -
Saloni Mittal (41251)
Vidula Magdum (41248)
Ashwini Kurhe (41245)

Batch: - R2
Guided by: - Prof. Yogesh Handge
Date: - 04/11/2022

DEPARTMENT OF COMPUTER ENGINEERING

Pune Institute of Computer Technology

Dhankawadi, Pune-43

CERTIFICATE

This is to certify that the Mini Project report entitled

"Implement merge sort and multithreaded merge sort"

Submitted by

Saloni Mittal	Roll No.: 41251
Vidula Magdum	Roll No.: 41248
Ashwini Kurhe	Roll No.: 41245

has successfully completed a mini project report under the guidance of Prof. Yogesh Handge towards the fulfilment of final year Computer Engineering Semester I, Academic Year 2022-23 of Savitribai Phule Pune University.

Prof. Yogesh Handge
Internal Guide

Place: Pune
Date: 04/11/2022

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.

Learning Objectives:

1. Implement merge sort and multi-threaded merge sort. Compare their time complexities and analyze performance.

Learning Outcomes:

After performing the mini-project student will be able to

1. To implement merge sort and multi-threaded merge sort, and compare their time complexities.

Software and Hardware Requirements:

1. 8 GB Ram
2. Ubuntu 20.4
3. Jupyter Notebook
4. Python

Concepts related Theory:

Merge Sort

Merge sort is a sorting technique which is based on the divide and conquer technique where we divide the array into equal halves and then combine them in a sorted manner.

The algorithm to implement merge sort is

- To check if there is one element in the list then return the element.
- Else, Divide the data recursively into two halves until it can't be divided further.
- Finally, merge the smaller lists into new lists in sorted order.

In merge sort, the problem is divided into two subproblems in every iteration.

Hence efficiency is increased drastically.

It follows the divide-and-conquer approach

- Divide break the problem into 2 subproblem which continues until the problem set is left with one element only.
- Conquer basically merges the 2 sorted arrays into the original array.

Multi-Threading

In the operating system, Threads are the lightweight process which is responsible for executing the part of a task. Threads share common resources to execute the task concurrently.

Multi-threading is an implementation of multitasking where we can run multiple threads on a single processor to execute the tasks concurrently. It subdivides specific operations within a single application into individual threads. Each of the threads can run in parallel.

Merge sort is a good design for multi-threaded sorting because it allocates sub-arrays during the merge procedure thereby avoiding data collisions. This implementation breaks the array up into separate ranges and then runs its algorithm on each of them, but the data must be merged (sorted) in the end by the main thread. The more threads there are, the more unsorted the second to last array is thereby causing the final merge to take longer.

Algorithm:

step 1: start

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

step 3: perform merge function.

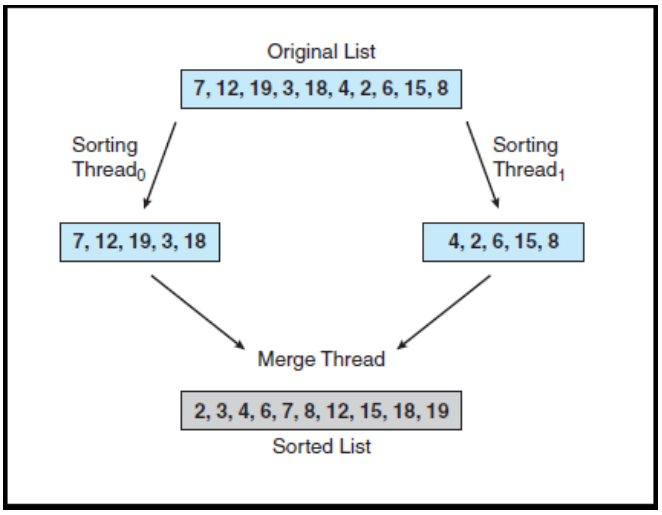
```
if left > right
    return
        mid= (left+right)/2
        mergesort(array, left, mid)
        mergesort(array, mid+1, right)
        merge(array, left, mid, right)
```

step 4: Stop

Multi-threaded Merge sort

Multi-threading is a 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.

Example:



Implementation screenshots:

Mergesort:

```
Problems  Tasks  Console ×  Properties  Call Graph
<terminated> (exit value: 0) Merge_sort.exe [C/C++ Application] D:\PICT\BE\41201_LP3_Assignments\DAA\Merge_sort\Debug\Merge_sort.exe (02/11/22, 11:03 pm)
Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13
```

Mergesort using multithreading:

```
isha@isha-Lenovo:~$ g++ -pthread mct.cpp -o mct
mct.cpp: In function 'void* merge_sort(void*)':
mct.cpp:75:1: warning: no return statement in function returning non-void [-Wreturn-type]
   75 | }
      | ^
isha@isha-Lenovo:~$ ./mct
Enter an element12
Enter an element30
Enter an element14
Enter an element55
Enter an element23
Enter an element1
Enter an element12
Enter an element3
Enter an element5
Enter an element4
Enter an element11
Enter an element19
Enter an element10
Enter an element17
Enter an element18
Enter an element14
Enter an element13
Enter an element22
Enter an element43
Enter an element56
Sorted array: 1 3 4 5 10 11 12 12 13 14 14 17 18 19 22 23 30 43 55 56 Time taken: 0.000744
isha@isha-Lenovo:~$
```

Time Complexity and Performance

Merge Sort

1. Time complexity: O(n*logn)
2. Space complexity: O(n)

It is a stable sorting algorithm.

Multi-threaded Merge Sort

Multithread merge sort, creates thread recursively, and stops work when it reaches a certain size, with each thread locally sorting its data. Then threads merge their data by joining threads into a sorted main list. The multithread merge sort that have array of 4 elements to be sorted. Merge sort in multithread is based on the fact that the recursive calls run in parallel, so there is only one n/2 term with the time complexity (2): $T(n) = \Theta \log(n) + \Theta(n) = \Theta(n)$

Conclusion:

Thus, We have implemented and compared time complexity and analysed performance of the Merge Sort and Multi-threaded Merge Sort.
