|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **المــمـــــــــلكــــة العـربية الســــــعودية**  **بــينبع الـــــــهـــــــــــيـــــئة الملــــــكـــــــية**  **قــــطـــاع الـكــــــــــــلــيات والـمــعــاهـد** | Description: YUC | **Kingdom of Saudi Arabia**  **Royal Commission at Yanbu**  **Colleges & Institutes Division** | | **كـــــلية ينبع الجــــــــــــــــامـــــــــــعية**  **قسم علوم وهندسة الحاسب الألي**  **قسم تقنية المعلومات والحاسب الآلي** | **Yanbu University College**  **Computer Science & Engineering Dept.**  **Information & Computer Technology Dept.** |   PROJECT |
| **ACADEMIC YEAR**  1442/1443 H(2021/2022 G)**,** SEMESTER II (212) |

|  |
| --- |
| **Design and Analysis of Algorithm**  CS302 |

|  |  |  |  |
| --- | --- | --- | --- |
| **DATE:** | Tuesday, 29 March, 2022 | **DUE DATE:** | Thursday, 12 May, 2022 |
|  |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **STUDENT NAME:** | Arwa Tammar 3910191  Assail Khalid 3910863 | | |
| **STUDENT ID:** | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  | | **SECTION:** | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FOR INSTRUCTOR USE ONLY** | | | | **GENERAL INSTRUCTIONS** |
| **Q. No.** | **CLOs** | **MAX MARK** | **MARKS OBTAINED** | * Write your name and I.D. number in the space provided above. * Support materials are not allowed in the examination Hall except those provided by instructor. * Do not use digital or printed dictionary. * Do not use pencils for answering except for drawing. * Read each question carefully before answering. * Number shown on the right-hand side against each question is the mark allocated. |
| 1 | 2.01 (A) | 5 |  |
| 1 | 2.02 (F) | 5 |  |
| 1 | 2.03 (C) | 2.5 |  |
| 1 | 2.03 (F) | 2.5 |  |
| 1 | 3.01 (E) | 2.5 |  |
| **TOTAL MARKS** | | **20** |  |

|  |  |
| --- | --- |
| **MARKED BY:** | **Signature:** |
| **CHECKED BY:** | **Signature:** |

|  |  |
| --- | --- |
| **Project Marks Distribution** | |
| **PROJECT DOCUMENT** |  |
| Algorithm | 5 |
| Time Complexity Analysis | 5 |
| Program | 10 |

**Instruction**

Choose one of the project topics provided in this document and update teacher and your class-fellows about your choice. Alternatively, you can propose your own project; seek teacher’s approval before working on your own idea.

**Group Size** Maximum 02 members

**Deliverables**

**PROJECT DOCUMENT:**

1. Title Page (page 1 and 2 – Fill up your names and IDs)
2. Problem Description (chosen Project Topic)
3. Table of Contents
4. Design of Algorithm
5. Program Codes (any language)
6. Computation of Time Complexity (use mathematical analysis)

**Project Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **What** | **Where** | **When** | **What’s next?** |
| Draft submission | Submit Project Topics Report Draft via BB. | 12 April, 2022 | Project review and feedback for completeness |
| Final submission | Submit complete project report on Blackboard(Assessment-Project) | 12 May, 2022 | Project Marks |

**Count Inversions in an array | (Using Merge Sort)**

Arwa Tammar

3910191

Assail Khalid

3910863

# Abstract

This paper examines how Merge Sort can be used to resolve and analyze Count Inversions in array problems. This research will focus on the algorithm that will be used to solve our problem, which is Enhance Merge Sort, which will be explained in more detail and its time complexity determined. Division and Conquest is the strategy used in the algorithm. In merge sorting, the array is always split into two halves, and merging them takes linear time. A time complexity analysis has been performed and it takes O(n log n).

Table of Contents

[Abstract 4](#_Toc103444102)

[1. Introduction 6](#_Toc103444103)

[1.1 problem description 6](#_Toc103444104)

[1.2 method of solving 6](#_Toc103444105)

[2. Design Algorithm 6](#_Toc103444106)

[3. Program Codes 7](#_Toc103444107)

[4. Computation of Time Complexity 7](#_Toc103444108)

[5. References 8](#_Toc103444109)

# 1. Introduction

## 1.1 problem description

Inversion count measures how far away an array is from being sorted (or how close it is). A sorted array has an inversion count of zero, while a reversed sorted array will have the

inversion count is the maximum.

## 1.2 method of solving

There are two methods to solve this problem, one is simple method that has the following approach: Traverse through the array, and for every index, find the number of smaller elements on its right side of the array. This can be done using a nested loop. Sum up the counts for all index in the array and print the sum.

Another method is the method that we will use to solve our problem, which is Enhance Merge Sort method.

# 2. Design Algorithm

Consider the number of inversions in the left half and right half of the array (let's call them inv1 and inv2). What kinds of inversions are not taken into account by inv1 + inv2? During the merge process, inversions need to be counted. For this reason, the number of inversions in the left subarray, right subarray, and merge() need to be added.

Following are the steps of the algorithm:

1) The idea is similar to merge sort, divide the array into two equal or almost equal halves in each step until the base case is reached.

2) Create a function merge that counts the number of inversions when two halves of the array are merged, create two indices i and j, i is the index for the first half, and j is an index of the second half. if a[i] is greater than a[j], then there are (mid – i) inversions. because left and right subarrays are sorted, so all the remaining elements in left-subarray (a[i+1], a[i+2] … a[mid]) will be greater than a[j].

3) Create a recursive function to divide the array into halves and find the answer by summing the number of inversions is the first half, the number of inversion in the second half and the number of inversions by merging the two.

4)The base case of recursion is when there is only one element in the given half.

5) Print the answer

# 3. Program Codes

Using C++ programing language

# 4. Computation of Time Complexity

The algorithm is based on merge sort it uses divide and conquer rule. The merge sort always divides the array into two halves and takes linear time to merge two halves.

T(N) = Time Complexity for problem size N

T(n) = Θ(1) + 2T(n/2) + Θ(n) + Θ(1)

T(n) = 2T(n/2) + Θ(n)

The following are the steps to analyze this algorithm:

T(n) = 2 \* T(n/2) + 0(n)

STEP-1 Is to divide the array into two parts of equal size.

2 \* T(n/2) --> Part 1

STEP-2 Now to merge basically traverse through all the elements.

constant \* n --> Part 2

STEP-3 --> COMBINE 1 + 2

T(n) = 2 \* T(n/2) + constant \* n --> Part 3

Now we can further divide the array into two halves if size of the partition arrays are greater than 1. So,

n/2/2--> n/4

T(N) = 2 \* (2 \* T(n/4) + constant \* n/2) + constant \* n

T(N) = 4 \* T(n/4) + 2 \* constant \* n

For this we can say that:

Where n can be substituted to 2^k and the value of k is log N

T(n) = 2^k \* T(n/(2^k)) + k \* constant \* n

Hence,

T(N) = N \* T(1) + N \* log N

= O(N \* log(N))

In conclusion, the time Complexity: O(n log n), The algorithm used is divide and conquer, So in each level, one full array traversal is needed, and there are log n levels, so the time complexity is O(n log n).

Space Complexity: O(n), Temporary array.

# 5. References

GeeksforGeeks. (2022, April 20). *Count Inversions in an array | Set 1 (Using Merge Sort)*. https://www.geeksforgeeks.org/counting-inversions/