A Project Report

On

NOVEL SORTING ALGORITHMS

BY

AKSHAY AKUTHOTA - SE20UARI013

KARTHIK CHIGULLAPALLI – SE20UCSE033

GOWTHAM TAMMANA – SE20UCSE048

PHANI SEKHAR REDDY– SE20UARI101

M NAGA SATHYA PRASAD RAJU – SE20UARI100

DOMA ADITI REDDY-SE20UARI171

BHARGAVA SAI N-SE20UCSE029

UPPALAPATI AKHIL VARMA-SE20UCSE212

Under the supervision of

DR. GARIMELLA RAMA MURTHY

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

PR 402: PROJECT-TYPE COURSE

ECOLE CENTRALE SCHOOL OF ENGINEERING

HYDERABAD

(SEP-DEC 2023)

ACKNOWLEDGMENTS

We would like to express our heartfelt gratitude and thanks to **MAHINDRA UNIVERSITY** for providing us with the opportunity to undertake this project. The support and encouragement we received from our institution have been instrumental in our successful completion of this project.

We extend our sincere thanks to **Dr. Arya Kumar Bhattacharya**, HOD of the Department of Computer Science and Engineering for giving this project. We would also like to express our gratitude to our project supervisor, **Dr. Garimella Rama Murthy**, Professor, Department of Computer Science and Engineering for his advice and continuous support. His expertise and mentorship played a crucial role in our project's success. We greatly appreciate his dedication and patience in guiding us through the challenges and providing valuable feedback that helped improve our work.

We would like to acknowledge our sincere appreciation for the knowledge and skills we have gained throughout this project. The experience of working on this project has been valuable for our upcoming future, and we are grateful for the opportunity to apply the concepts we have learned in a practical setting.

Ecole Centrale School of Engineering

Hyderabad

Certificate

This is to certify that the project report entitled **" NOVEL SORTING ALGORITHMS "** submitted by A Akshay(SE20UARI013), Phani Sekhar(SE20UARI101), C Karthik(SE20UCSE033), T Gowtham(SE20UCSE048), M Naga Sathya(SE20UARI100),Aditi Reddy (SE20UARI171), Akhil Varma(SE20UCSE212),N Bhargava Sai(SE20UCSE029) in partial fulfillment of the requirements of the course PR 402, Project Course, embodies the work done by him/her under my supervision and guidance.

Dr. Garimella Rama Murthy & Signature

Ecole Centrale School of Engineering, Hyderabad.

Date:

ABSTRACT

The project report "Novel Sorting Algorithms" presents a unique algorithm that operates on a matrix constructed from a set of integers. The matrix, built using binary digits, facilitates the extraction of information about the numbers. The report details methods for finding the median, as well as the pth maximum and minimum values from the matrix. These processes involve counting binary ones and zeros in matrix rows and adding extra columns based on specific conditions. The report also includes equations using 'OR' and 'AND' gates to calculate these values, demonstrating the application of these concepts in sorting and data analysis.

INTRODUCTION

Our algorithm takes input as a set of intergers and it helps us in getting the information about the numbers in a very unique way. We first build a matrix which only consists of 0s and 1s and draw all the information required from that matrix. We use Boolean properties to find the median, maximum and minimum of the input set.

MATRIX BUILDING

- We get an input as a set of integer elements. Let us assume the size of the input set is 'N'.
- We will find the maximum element of the input set. We will call this number as 'Max'.
- Now we will create matrix with the size of [Max+1][N+1].
- The first column of our matrix is numbers from 0 to Max.
- The first row of our matrix would be the input set of integers.
- Our next step woud be filling the matrix with binary digits following the below rule:
    - The program then compares the values in the first column (representing the values from 1 to max) with the values in the first row (representing the elements in the array a). If the value in the first column is less than or equal to the value in the first row, it sets the corresponding cell in mat to 1; otherwise, it sets it to 0.
- This is how we build our matrix. Let us look at an example on how to build the matrix.

Example:

Input: [5 4 6 7 8]

Max: 8

Our matrix will look like this:

$$\begin{bmatrix} 0 & 5 & 4 & 6 & 7 & 8 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 & 1 & 1 \\ 4 & 1 & 1 & 1 & 1 & 1 \\ 5 & 1 & 0 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 & 1 & 1 \\ 7 & 0 & 0 & 0 & 1 & 1 \\ 8 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

FINDING THE MEDIAN:

- We will count the no.of 1's and no.of 0's in the each and every row .
- If the no of 1's is greater than no.of 0's then we add a extra column (V) and fill 1 with respective of row and if no.of 0's is greater than 1's then we fill with 0 in that respective row.
- The value of Median will be the sum of elemnts of the column (V).
- $\begin{bmatrix} 0 & 5 & 4 & 6 & 7 & 8 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 & 1 & 1 \\ 4 & 1 & 1 & 1 & 1 & 1 \\ 5 & 1 & 0 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 & 1 & 1 \\ 7 & 0 & 0 & 0 & 1 & 1 \\ 8 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$
- $sum = 1 + 1 + 1 + 1 + 1 + 1 + 0 + 0 = 6$

FINDING THE Pth MAXIMUM :

- We will count the no.of 1's and no.of 0's in the each and every row .
- The P value will be taken as an input and we compare the no.of 1's in each row with the value of P
- If the no of 1's is greater than equal to P then we add a extra column (V) and fill 1 with respective of row else we fill V column with 0.
- The value of Pth maximum will be the sum of elemnts of the column (V).
- $\begin{bmatrix} 0 & 5 & 4 & 6 & 7 & 8 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 & 1 & 1 \\ 4 & 1 & 1 & 1 & 1 & 1 \\ 5 & 1 & 0 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 & 1 & 1 \\ 7 & 0 & 0 & 0 & 1 & 1 \\ 8 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ $(assuming\ p\ value\ as\ 2)$
- $sum = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 0 = 7$

FINDING THE Pth MINIMUM :

- We will count the no.of 1's and no.of 0's in the each and every row .
- The P value will be taken as an input and we compare the no.of 0's in each row with the value of P
- If the no of 0's is greater than to P then we add a extra column (V) and fill 0 with respective of row else we fill V column with 1.
- The value of Pth minimum will be the sum of elemnts of the column (V).
- $\begin{bmatrix} 0 & 5 & 4 & 6 & 7 & 8 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 & 1 & 1 \\ 4 & 1 & 1 & 1 & 1 & 1 \\ 5 & 1 & 0 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 & 1 & 1 \\ 7 & 0 & 0 & 0 & 1 & 1 \\ 8 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $(assuming\ p\ value\ as\ 2)$
- $sum = 1 + 1 + 1 + 1 + 1 + 0 + 0 + 0 = 5$

COMPLEXITIES:

Time Complexity: $O(N * MAX)$

Space Complexity:$O(N * MAX)$

CODE FOR MEDIAN:

```c
#include <stdio.h>

int main() {
    int N;
    printf("Enter the number of elements: ");
    scanf("%d", &N);

    int a[N], b[N]; // Declare two arrays of size N
    int i;

    printf("Enter %d elements:\n", N);
    for (i = 0; i < N; i++) {
        scanf("%d", &a[i]); // Read N integers into array 'a'
    }

    int max = 0; // Initialize 'max' to 0
    for (i = 0; i < N; i++) {
        if (a[i] > max) {
            max = a[i]; // Find the maximum element in array 'a'
        }
    }
    // Create a 2D matrix 'mat' with dimensions (max + 1) x (N + 1)
    int mat[max + 1][N + 1];
    mat[0][0] = 0; // Initialize the first element of 'mat' to 0

    // Initialize the first row of 'mat' with elements from array 'a'
    for (int j = 1; j <= N; j++) {
        mat[0][j] = a[j - 1];
    }

    // Initialize the first column of 'mat' with values from 1 to
'maxfor (int i = 1; i <= max; i++) {
        mat[i][0] = i;
    }

    // Fill 'mat' with 1 if a condition is met, 0 otherwise
    for (int i = 1; i <= max; i++) {
        for (int j = 1; j <= N; j++) {
            if (mat[i][0] <= mat[0][j]) {
                mat[i][j] = 1;  // Set to 1 if the condition is met
            } else {
                mat[i][j] = 0;  // Set to 0 otherwise
            }
        }
    }
    // Calculate 'b' based on counts of 1s and 0s in each row of 'mat'
    for (int i = 1; i <= max; i++) {
        int count1 = 0, count0 = 0;
        for (int j = 1; j <= N; j++) {
            if (mat[i][j] == 1) {
                count1++;
            } else {
                count0++;
            }
        }
        if (count1 > count0) {
            b[i - 1] = 1;
        } else {
            b[i - 1] = 0;
        }
    }
    int sum = 0;
    for (int i = 0; i < max; i++) {
        sum += b[i]; // Calculate the sum of 'b' elements
    }
    printf("\nmedian of the given array is: %d\n", sum);
    return 0;
}
```

CODE FOR PTH MAXIMUM:

```c
#include <stdio.h>

int main() {
    int N, p; // Declare variables for the number of elements and the pth minimum
    printf("Enter the number of elements: ");
    scanf("%d", &N);

    int a[N]; // Declare an array 'a' of size 'N' to store the elements
    int i, count = 0, count0 = 0; // Declare loop counters and counters for 'count' and
'count0'
    printf("Enter %d elements:\n", N);
    for (i = 0; i < N; i++) {
        scanf("%d", &a[i]); // Input 'N' elements and store them in the array 'a'
    }
    printf("Enter the required pth minimum: ");
    scanf("%d", &p); // Input the value of 'p'

    int max = 0; // Initialize a variable 'max' to store the maximum element

    // Find the maximum element in the array 'a'
    for (i = 0; i < N; i++) {
        if (a[i] > max) {
            max = a[i];
        }
    }

    int d[max]; // Declare an array 'd' to store results
    int mat[max + 1][N + 1]; // Declare a 2D array 'mat' for dynamic programming
    mat[0][0] = 0; // Initialize the first element of 'mat' as 0

    // Initialize the first row of 'mat' with the elements from array 'a'
    for (int j = 1; j <= N; j++) {
        mat[0][j] = a[j - 1];
    }

    // Initialize the first column of 'mat' with increasing values from 1 to 'max'
    for (int i = 1; i <= max; i++) {
        mat[i][0] = i;
    }

    // Populate 'mat' based on a condition (0 or 1) comparing rows and columns
    for (int i = 1; i <= max; i++) {
        for (int j = 1; j <= N; j++) {
            if (mat[i][0] <= mat[0][j]) {
                mat[i][j] = 1;  // Set to 1 if the condition is met
            } else {
                mat[i][j] = 0;  // Set to 0 otherwise
            }
        }
    }

    // Calculate 'd' array based on 'mat' and the value of 'p'
    for (int i = 1; i <= max; i++) {
        int count = 0, count0 = 0;
        for (int j = 1; j <= N; j++) {
            if (mat[i][j] == 1) {
                count++;
            } else {
                count0++;
            }
        }
        if (count0 >= p) {
            d[i - 1] = 0; // If count0 is greater than or equal to 'p', set 'd' to 0
        } else {
            d[i - 1] = 1; // Otherwise, set 'd' to 1
        }
    }

    // Calculate the 'p'th minimum and print the result
    int sum = 0;
    for (int i = 0; i < max; i++) {
        sum += d[i];
    }
    printf("\n%dth minimum of given is : %d\n", p, sum);
    return 0;
}
```

CODE FOR PTH MANIMUM:

```c
#include <stdio.h>

int main() {
    int N, p;

    // Prompt the user to enter the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &N);

    // Declare arrays to store input and intermediate results
    int a[N], b[N];
    int i, count = 0, count0 = 0;

    // Prompt the user to enter N elements
    printf("Enter %d elements:\n", N);
    for (i = 0; i < N; i++) {
        scanf("%d", &a[i]);
    }

    // Prompt the user to enter the desired pth max
    printf("Enter the required pth max: ");
    scanf("%d", &p);

    // Find the maximum element in the array 'a'
    int max = 0;
    for (i = 0; i < N; i++) {
        if (a[i] > max) {
            max = a[i];
        }
    }

    // Create a matrix 'mat' for dynamic programming
    int c[max];
    int mat[max + 1][N + 1];
    mat[0][0] = 0;

    // Fill the first row of 'mat' with elements from array 'a'
    for (int j = 1; j <= N; j++) {
        mat[0][j] = a[j - 1];
    }

    // Fill the first column of 'mat' with numbers from 1 to 'max'
    for (int i = 1; i <= max; i++) {
        mat[i][0] = i;
    }

    // Populate 'mat' based on a condition
    for (int i = 1; i <= max; i++) {
        for (int j = 1; j <= N; j++) {
            if (mat[i][0] <= mat[0][j]) {
                mat[i][j] = 1;  // Set to 1 if the condition is met
            } else {
                mat[i][j] = 0;  // Set to 0 otherwise
            }
        }
    }

    // Calculate 'c' based on the number of 1s in each row of 'mat'
    for (int i = 1; i <= max; i++) {
        int count = 0, count0 = 0;
        for (int j = 1; j <= N; j++) {
            if (mat[i][j] == 1) {
                count++;
            } else {
                count0++;
            }
        }
        if (count >= p) {
            c[i - 1] = 1;
        } else {
            c[i - 1] = 0;
        }
    }

    // Calculate and print the pth maximum
    int sum = 0;
    for (int i = 0; i < max; i++) {
        sum += c[i];
    }
    printf("\n %dth max of given is : %d\n", p, sum);
    return 0;
}
```

HTML INTERFACE

- We have also created an interface for this algorithm. The HTML codes for which are there in attached Github Repository.

  Github Link: https://github.com/SekharReddy07/Sorting-Algorithm-Using-Boolean-functions.git

EQUATION FOR MEDIAN:

- The equation for median is a combination of 'OR' gates , 'AND' gates. If there are N integers in our input array then we take all combinations of N+1/2 elements of each row and perform 'AND' operation between the elements and then we perform 'OR' operation for every individual 'AND' components.
- The equation for median if we take 5 elements in input array looks like:

$$Y = (X1.X2.X3) + (X1.X3.X4) + (X1.X4.X5) + (X2.X3.X4) + (X2.X4.X5) + (X3.X4.X5) + (X1.X2.X4) + (X1.X2.X5) + (X2.X3.X5) + (X1.X3.X5)$$

- We calculate the above Y for each row and then add the values of Y for to get our median.

EQUATION FOR Pth MAXIMUM:

- The equation for Pth Maximum is a combination of 'OR' gates , 'AND' gates. If there are N integers in our input array and we want to find Pth maximum then we take all combinations of P elements of each row and perform 'AND' operation between the elements and then we perform 'OR' operation for every individual 'AND' components.
- The equation for Pth maximum if we take 5 elements in input array looks like:
  Where P=2

$$Y = (X1.X2) + (X1.X3) + (X1.X4) + (X1.X5) + (X2.X3) + (X2.X4) + (X2.X5) + (X3.X4) + (X3.X5) + (X4.X5)$$

- We calculate the above Y for each row and then add the values of Y for to get our 2nd maximum.

EQUATION FOR Pth MINIMUM:

- The equation for Pth Minimum is a combination of 'OR' gates , 'AND' gates. If there are N integers in our input array and we want to find Pth minimum then we take all combinations of P elements of each row and perform 'OR' operation between the elements and then we perform 'AND' operation for every individual 'OR' components.
- The equation for Pth minimum if we take 5 elements in input array looks like:
  Where P=2

$$Y = (X1 + X2).(X1 + X3).(X1 + X4).(X1 + X5).(X2 + X3).(X2 + X4).(X2 + X5).(X3 + X4).(X3 + X5).(X4 + X5)$$

- We calculate the above Y for each row and then add the values of Y for to get our 2nd minimum.