

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE

“Igor Sikorsky KYIV POLYTECHNIC INSTITUTE”

Faculty of Applied Mathematics

Department of Computer Systems Software

Course project

of the discipline "*Technologies of Software System Design*"

Titled

Matrix Operations and Selection Sort

Done by AYENI OLARINDE LUCAS

Of study group KII- 92M

Project curator

Senior teacher Ruslan Hadyniak

Points

(signature)

Kyiv 2019

Contents

Overview of Task	3
Requirements and Analysis.....	3
Different Operations on Matrices.....	3
ALGORITHM.....	4
Matrices subtraction.....	5
Selection Sort Algorithm.....	7
Result	10
Add Algorithm	10
Transpose Algorithm	10
Seletion Sort Algorithm	10
Summary	11

Course Project Task Variant

1. Variant:

- User interaction type: **full command strings**
- Matrix operations: **adding, subtracting, transpose**
- Sorting algorithm: **selection**

Overview of Task

In this section, project's requirements will be determined and related analysis will be considered. The purpose of this project is to perform predetermined mathematical operations and sort using the user's matrices and array.

To be easy to understand, mathematical functions, sort function, matrices, array and the main program will be in separate titles.

The whole project is have six different files. The main file app.js which main program is taking place, the function file functions.js and sort.function.js which handles all the function running on the app.js. The fun.js , sort.js and the app.js are linked with export and import methods.

Definition of Matrix

A matrix is a collection of numbers arranged into a fixed number of rows and columns. Usually the numbers are real numbers. In general, matrices can contain complex numbers. Here is an example of a matrix with three rows and three column

$$\begin{pmatrix} 5 & 4 & 3 \\ -4 & 0 & 4 \\ 7 & 10 & 3 \end{pmatrix}$$

Each number that makes up a matrix is called an element of the matrix. The elements in a matrix have specific locations.

The upper left corner of the matrix is row 1 column 1. In the above matrix the element at row 1 col 1 is the value 5. The element at row 2 column 3 is the value 4.

Different Operations on Matrices

For introduction on matrices, you can refer the following article: [Matrix Introduction](#)

In this article, we will discuss various operations on matrices and their properties:

1. Matrices Addition –

The addition of two matrices $A_{m \times n}$ and $B_{m \times n}$ gives a matrix $C_{m \times n}$. The elements of C are sum of corresponding elements in A and B which can be shown as:

$$\begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 12 & 14 \end{bmatrix}$$

ALGORITHM

The algorithm for addition of matrices can be written as:

for i in 1 to m

 for j in 1 to n

$c_{ij} = a_{ij} + b_{ij}$

-The user will call 2 matrices created to perform matrix addition. (matrixA.json and matrixB.json)

- The command (node app.js matrix sum matrix1.json matrix2.json)

```
// The Sum of two matrixs A and B
else if (operationType === "sum")
{
  let matrixAFilePath = arrayOfCommand[2];
  let matrixA = matrix_from_file(matrixAFilePath)
  let matrixBFilePath = arrayOfCommand[3];
  let matrixB = matrix_from_file(matrixBFilePath)
  console.log("matrixA")
  matrix_print(matrixA)
  console.log("matrixB")
  matrix_print(matrixB)
  console.log("whats up \n")
  let matrixC = matrix_sum(matrixA, matrixB)
  console.log(`The Sum of the two matrixs is: \n `)
  matrix_print(matrixC)
}
```

Funtion call for the addtion of two matrix

```

3  function matrix_sum(a,b)
4  {
5  let m=[]
6  m=new Array(a.length);
7      for (let i =0 ;i<m.length;i++)
8      {
9          m[i]= new Array(a[i].length)
10
11          for(let j=0 ; j<m.length; j++)
12          {
13              m[i][j]=a[i][j]+b[i][j]
14          }
15      }
16      if(b.length != a.length)
17      {
18          return "Undifined Matrics"
19      }
20      else
21      {
22          return m
23      }
24  }

```

This Function matrix_sum is been call by app.js

Subtraction is accomplished by subtracting corresponding elements. For example, consider matrix A and matrix B.

A =

1	2	3
7	8	9

B =

5	6	7
3	4	5

Both matrices have the same number of rows and columns (2 rows and 3 columns), so they can be added and subtracted. Thus,

And,

A - B =

1 - 5	2 - 6	3 - 7
7 - 3	8 - 4	9 - 5

A - B =

-4	-4	-4
4	4	4

And finally, note that the order in which matrices are added is not important; thus, $A + B = B + A$.

Here is our implementation:

```
(operationType === "subtract")
{
  let matrixAFilePath = arrayOfCommand[2];
  let matrixA = matrix_from_file(matrixAFilePath)
  let matrixBFilePath = arrayOfCommand[3];
  let matrixB = matrix_from_file(matrixBFilePath)
  console.log("matrixA")
  matrix_print(matrixA)
  console.log("matrixB")
  matrix_print(matrixB)
  console.log("\n")
  let matrixC = matrix_sub(matrixA, matrixB)
  console.log(`The Sum of the two matrixs is: \n `)
  matrix_print(matrixC)
}
```

2. Transpose Algorithm

- To perform this operation, the selected matrix must be square matrix.
- If the selected matrix is not a square matrix, the output will be "invalid matrix: it should be a square matrix" error.
- The user will select a matrix to perform transpose (matrix1.json or matrix2.json)
- Written code to replace the row and column values of the selected matrix.
- The command (node app.js matrix transpose matrix1.json)
- The command (node app.js matrix transpose matrix2.json)

The exported files are again imported to where needed in this case to app.js. And how to import is shown on below screen shot

matrix1.json

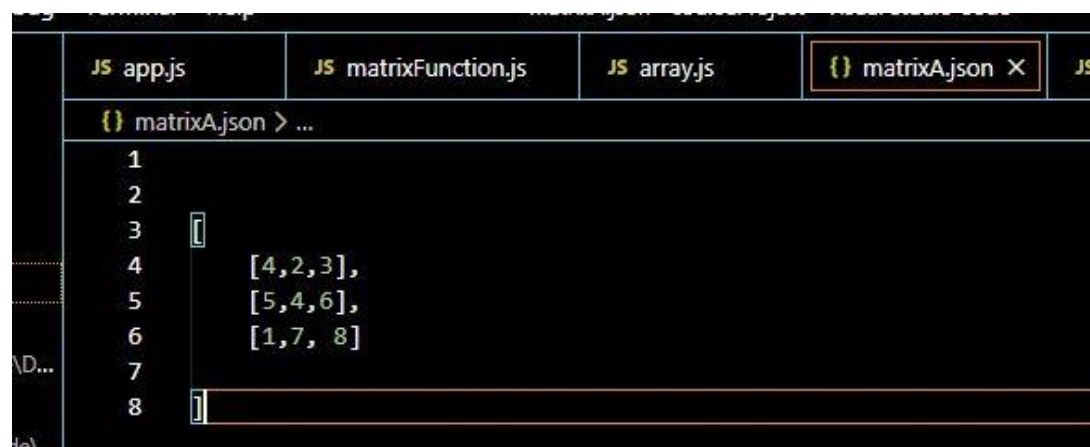
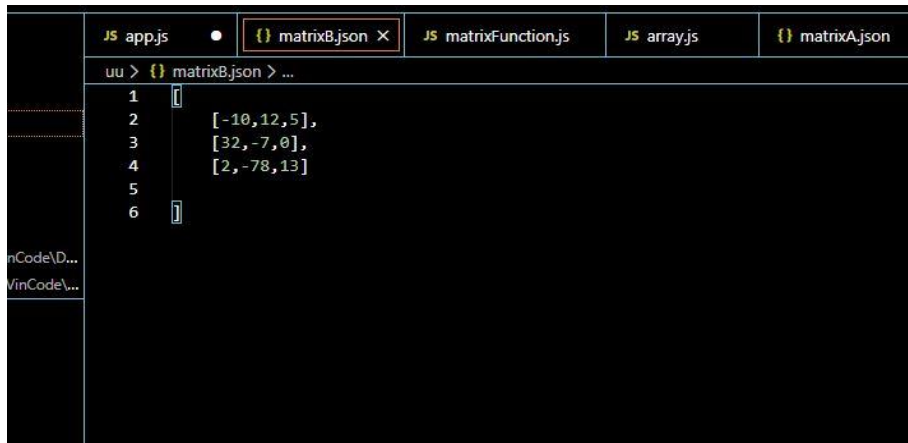


Figure 1 matrix2.json



The screenshot shows a code editor with several tabs: 'JS app.js', 'matrixBjson X', 'JS matrixFunction.js', 'JS array.js', and 'matrixAjson'. The active tab is 'matrixBjson X'. Below the tabs, a console window shows the prompt 'uu > {} matrixBjson > ...' followed by a JavaScript array: `[[-10,12,5], [32,-7,0], [2,-78,13]]`. The array is displayed across six lines, with line numbers 1 through 6 on the left.

```
function matrix_transpose(m) {  
  // performs operation  
  let height = m.length  
  let width = m[0].length  
  // console.log(height, width, m)  
  if (height !== width)  
    throw new Error("invalid matrix: it should be a square matrix")  
  let temp = matrix_create(height, width)  
  for (let i = 0; i < height; i++)  
  {  
    for (let j = 0; j < width; j++)  
    {  
      temp[i][j] = m[j][i]  
    }  
  }  
  return temp  
}
```

Transpose function

Selection Sort Algorithm

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where n is the number of items.

Algorithm

Step 1 – Set MIN to location 0

Step 2 – Search the minimum element in the list

Step 3 – Swap with value at location MIN

Step 4 – Increment MIN to point to next element

Step 5 – Repeat until list is sorted

```
else if (operationType === "sort") {  
    // one matrix only can be sorted  
    let matrixFilePath = arrayOfCommand[2];  
    let matrix = matrix_from_file(matrixFilePath)  
    let selection_Sort = selection_sort(matrix)  
    matrix_print(selection_Sort)  
}
```

```
function findMinIndex(a, startIndex)  
{  
    let minValue=a[startIndex]  
    let minIndex=startIndex  
    for (let i=startIndex+1; i<a.length; i++)  
    {  
        if (a[i]<minValue)  
        {  
            minValue= a[i]  
            minIndex=i  
        }  
    }  
    return minIndex  
}  
  
function selection_sort(a, firstIndex, secondIndex) { //swap function  
  
    let tmp = a[firstIndex]  
    a[firstIndex] = a[secondIndex]  
  
    a[secondIndex] = tmp  
}
```

Selection sort implementation

-We created an one dimensional array to implement this algorithm.

-By using this algorithm user will sort that array.

- The command (node app.js array sort array.json)

selection_sortfunction

Here you can see export of functions.js to implement in app.js

Export section

Here you can see export of function.js to implement in app.js

```
module.exports = {  
  matrix_create,  
  matrix_sum,  
  matrix_transpose,  
  matrix_print,  
  matrix_from_file,  
  matrix_sub,  
  findMinIndex,  
  selection_sort  
}
```

The exported files are again imported to where needed in this case to app.js. And how to import is shown on below screen shot.

```
let {  
  
  matrix_sub,  
  selection_sort,  
  matrix_transpose,  
  matrix_print,  
  findMinIndex,  
  matrix_sum,  
  matrix_from_file,  
  
} = require('./functions.js')
```

Import section

Result

```
Enter a command:
> matrix sum ./matrix1.json ./matrix2.json
matrix sum
matrixA
[ 1, 8, -10 ]
[ 8, 4, 3 ]
[ 13, 0, 5 ]
matrixB
[ -10, 12, 5 ]
[ 3, -7, 0 ]
[ 2, -8, 13 ]

The Sum of the two matrixs is:

[ 11, -4, -15 ]
[ 5, 11, 3 ]
[ 11, 8, -8 ]
PS C:\Users\EVinCode\Documents\Semester 1\Technology of Software Systems Design\lucas course project>
```

Figure 2 addition result

Transpose Algorithm

matrix1.json

```
PS C:\Users\EVinCode\Documents\Semester 1\Technology of Software Systems Design\lucas course project>
Enter a command:
> matrix transpose ./matrix1.json
matrix transpose
[ 1, 8, 13 ]
[ 8, 4, 0 ]
[ -10, 3, 5 ]
PS C:\Users\EVinCode\Documents\Semester 1\Technology of Software Systems Design\lucas course project>
```

Seletion Sort Algorithm

```
Enter a command:
> sort ./array.json
[
  -68, -8, -6, -3, 0,
  12, 28, 31, 73, 95
]
PS C:\Users\EVinCode\Documents\Semester 1\Technology of Software Systems Design\lucas course project>
```

Figure 3 array sorting by selection

Summary

This project is to create general functions based on modular coding, to be able to call the functions and apply whenever they are needed for matrices and arrays operations.