

# *The Report for Data Structure Project*

---

*Student ( 1 ) :*

أحمد ياسر محمد محمد نصار

ID : 20191616316

---

*Student ( 2 ) :*

عطية فتحي عطية أحمد محمد

ID : 20191480940

*This project was made by only two individuals.*

---

*Student ( 1 ) :*

. أَحْمَدْ يَاسِرْ مُحَمَّدْ نَصَارْ .

Contributed to this project by solving :

- Problem ( 1 ) Hashing.
- Problem ( 4 ) Minimum Spanning Tree.

And maker of this report

---

*Student ( 2 ) :*

. عَطِيَّهْ فَتَحِيْ عَطِيَّهْ أَحْمَدْ مُحَمَّدْ .

Contributed to this project by solving :

- Problem ( 2 ) Binary Search Tree.
- Problem ( 3 ) Topological Sorting.

And helped in the making of this report.

# Contents

<b>Problem ( 1 )</b>	Hashing	<b>Pages ( 1 - 10 )</b>
- Problem statement		Page 1
- Pseudocode.		Page 2,3,4,5
- Time and Space complexity		Page 6
- Sample Runs		Page 7,8,9
<b>Problem ( 2 )</b>	Binary Search Tree	<b>Pages (10 - 16)</b>
- Problem statement		Page 10
- Pseudocode.		Page 11,12
- Time and Space complexity		Page 13
- Sample Runs		Page 14,15,16
<b>Problem ( 3 )</b>	Topological Sorting	<b>Page (17 - 22)</b>
- Problem statement		Page 17
- Pseudocode.		Page 18
- Time and Space complexity		Page 19
- Sample Runs		Page 20,21,22
<b>Problem( 4 )</b>	Minimum Spanning Tree	<b>Page (23 -29 )</b>
- Problem statement		Page 23
- Pseudocode.		Page 24,25
- Time and Space complexity		Page 26
- Sample Runs		Page 27,28,29

## *First Hashing ( problem statement ) :*

-The first problem was to try to solve the problem with only arrays which made it more complicated instead of using java hash map .

-The second problem that to make the program run correctly and to make sure that the output would be one hundred percent correct as it depends on ASCII code to be able to establish the difference between the capital and small letters as well as all the special characters and numbers and the solution for this problem is to simply sort the given array from the user through any sorting algorithm .

- Pseudocode :

for the sorting Algorithm

```
public static String Ss(String x){  
    char sa[] = new char[x.length()];  
    String as = "";  
    For (int i = 0 ; i <= sa.length(); i++){  
        sa[i] = x.charAt(i); } }
```

Which is Bubble sort

```
for(int i=0;i<sa.length-1;i++){  
    for(int j=0;j<sa.length-1-i;j++){  
        if(sa[j]>sa[j+1]){  
            true then swap  
            char temp=sa[j];  
            sa[j]=sa[j+1];  
            sa[j+1]=temp;  
        } } } else terminate
```

*- Pseudocode ( CONT ):*

```
char Get-Most-Occurrence (String st){  
    char sa[] = new char[st.length()];  
    initialize String as  
    for(int i=0;i->to st size;i++){  
        sa[i]=st.charAt(i);put the chars in sa }  
    for(int i=0;i->to sa.size-1;i++){  
        for(int j=0;j->to sa.size-1-i;j++){  
            if(sa[j]>sa[j+1]) {if true then swap  
            char temp=sa[j];  
            sa[j]=sa[j+1];  
            sa[j+1]=temp;  
        } } } else terminate
```

- Pseudocode ( CONT ): Get Most Occurrence

```
for(int i=0;i->to sa.size;i++)
    as+=sa[i];} reform the old string in as
    create array int co[] = new int[256];
    for(int i=0;i->to as.size;i++)
        co[as.charAt(i)]++;
    initialize Int max=-1;
    initialize char (most occurrence)result=' '
    for(int i=0;i-> to as.size;i++){
        if(max<co[as.charAt(i)]){
            max=co[as.charAt(i)];
            result=as.charAt(i); } }
    return result(char that occurred most);}
```

## - Pseudocode ( CONT ):

Method to get the number of occurrence

```
(String st){Create Int co [ ]
```

```
For(int i=0;i->to sr.size){
```

```
co[st.charAt(i)]++;
```

```
int max=-1;
```

```
char result=' ';
```

```
create int (number of occurrence) noo=0;
```

```
for(int i=0;i<st.length();i++){
```

```
if(max<=co[st.charAt(i)]){
```

```
max=co[st.charAt(i)];
```

```
result=st.charAt(i); noo=max;
```

```
} } return (noo)number of occurrence; }
```

## Complexity

### Time analysis :

*The advantage of using bubble sort is that it doesn't create a new array to sort the elements in it sentence that is in the project which is*

*( Faculty of Computers and Data Science ) as a test*

*Took 2 seconds average case. Since the time complexity of the Bubble sort on average cases is  $O(n^2)$  it is efficient in our project as data sets is usually small and not large if we want to improve the efficiency of it we can use merge sort.*

### Space analysis :

*As before Bubble sort 's ability that it doesn't create a new array so it doesn't take a big memory space so it 's very efficient in saving memory space and other methods have for loops and arrays  $O(n)$  it really depends on the user.*

**Page ( 6 )**

## Sample Runs

Hashing - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files Start Page Hashing.java

Source History

```
1 package hashing;
2
3 /**
4 * Author : Sauthor Ahmed
5 */
6 import java.util.Scanner;
7 public class Hashing {
8     public static final String Ss(String x){
9         char sa[]=new char[x.length()];
10    String as="";
11    for(int i=0;i<x.length();i++){
12        sa[i]=x.charAt(i);
13    }
14    for(int i=0;i<sa.length-1;i++){
15        for(int j=0;j<sa.length-1-i;j++){
16            if(sa[j]>sa[j+1]){
17                char temp=sa[j];
18                sa[j]=sa[j+1];
19                sa[j+1]=temp;
20            }
21        }
22    }
23    return as;
24 }
```

Output - Hashing (run) | Terminal - localhost

```
run:
please Write a sentence
my name is ahmed and iam a student in the faculty of computer and data science
The Maximum occurrence is ( Space ) And The Num of it's occurrence is ( 15 )
BUILD SUCCESSFUL (total time: 1 second)
```

*This sample run has the sentence*

*(my name is ahmed and iam a student in the  
faculty of computer and data science)*

**And the output is : The Maximum occurrence is  
( Space ) And The Num of it's occurrence is ( 15 )**

## Another sample Run

The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The title bar says "Hashing - NetBeans IDE 8.2". The left sidebar shows the Projects view with several Java applications listed. The main workspace shows the "Hashing.java" file in the "Source" tab. The code is as follows:

```
1 package hashing;
2
3 /**
4 * 
5 * @author Ahmed
6 */
7
8 import java.util.Scanner;
9 public class Hashing {
10     public static final String Ss(String x){
11         char sa[]=new char[x.length()];
12         String as="";
13         for(int i=0;i<x.length();i++){
14             sa[i]=x.charAt(i);
15         }
16         for(int i=0;i<sa.length-1;i++){
17             for(int j=0;j<sa.length-1-i;j++){
18                 if(sa[j]>sa[j+1]){
19                     char temp=sa[j];
20                     sa[j]=sa[j+1];
21                     sa[j+1]=temp;
22                 }
23             }
24         }
25         return as;
26     }
27 }
```

The Output window shows the run command, user input "please Write a sentence", the program's response "Faculty of Computers and Data Science", and the output "The Maximum occurrence is ( Space ) And The Num of it's occurrence is ( 5 )". The status bar at the bottom indicates the build was successful.

Here the project sample input and the sentence is(Faculty of Computers and Data Science )

And the output is : *The Maximum occurrence is ( Space ) And The Num of it's occurrence is ( 5 )*

## Another sample Run

The screenshot shows the NetBeans IDE interface with the following details:

- Projects:** 8-Puzzle, Calculator, Exception, firstclass, Hashing (selected), JavaApplication1, JavaApplication2, JavaApplication22, JavaApplication23.
- Files:** Hashing.java (selected).
- Code Editor:** Displays the Java code for the Hashing class, which reads a sentence from the user and prints the character with the highest frequency and its count.
- Output:** Shows the terminal output where the user inputs a sentence containing special characters like '&&\*(%%%????&aaaAAAAA' and the program correctly identifies 'A' as the character with the maximum occurrence of 5.
- System Tray:** Shows the Windows taskbar with various icons and the date/time (8:13 PM, 6/16/2020).

The final sample run shows that you can use special characters like these in the sentence of the input (&&\*(%%%????&aaaAAAAA)

**And the output is : The Maximum occurrence is ( A ) And The Num of it's occurrence is ( 5 )**

## Second Binary Search Tree :

( problem statement )

- The second problem in the project and to make binary tree in java using only array is possible.
- The main problem that was difficult to solve for us was to find a way to allow the user to enter a repetitive number and to be aware that the calculation of the redundancy ratio doesn't include this repetitive number and keep in mind this process was made during the run time in other words while the program was running.

- Pseudocode :

```
main(String[] args) {  
    int da ; double ratio, z ;  
    BinaryTree bt ,bin;  
    bin.insert(0);  
    z=input from the user;  
    do{  
        da=User ;  
        bt.insert(da);  
        while(bin.search(da)==false&&da!= -1){  
            bin.insert(da);  
        }  
    }
```

- Pseudocode(Cont) :

```
if(bt.search(-1)==true){  
ratio=((bt.countNodes1)/(bin.countNodes-1));  
}  
}else {  
ratio=((bt.countNodes)/(bin.countNodes-1));  
}  
} }while(da!= -1 && ratio<z);  
if(bt.search(-1)==true){  
    Display(ratio);  
    Display(Only Few Repetitions);  
}  
else{  
    Display(ratio);  
    Display(Many Repetitions); } }
```

## Complexity

### Time analysis :

*From the construction and structure of the binary search tree instead of using the one that was built already took a lot of time as also the user is involved the longer he take to insert the number the longer the program will take time as it took 5 seconds to run the program so  $O(n)$*

### Space analysis :

*The class of the binary search tree has a lot of methods which are necessary for insertion and searching as well as deletion so it took a lot of memory and space so as the complexity  $O(n)$  in the worst cases but  $O(\log n)$  in the best and average cases.*

## Sample Runs

The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The title bar says "Final Project - NetBeans IDE 8.2". The left sidebar has a Projects tab, Services tab, and Files tab, showing a hierarchy of packages like P1\_Hashing, P2\_BST, P3\_TopSort, P4\_MST, and Test Packages. The main workspace shows two files: BST.java and Node.java. BST.java contains code related to binary search trees. The output window at the bottom shows the following interaction:

```
Please enter the redundancy ratio that is a real number greater than 1 as a double number as (1.1)
1.5
To finish the program before reaching the desired percentage, please enter the number: (-1)
please Enter one integer number per line from 1 to 9
1
4
2
5
2
ratio= 1.5
Many Repetitions
BUILD SUCCESSFUL (total time: 5 seconds)
```

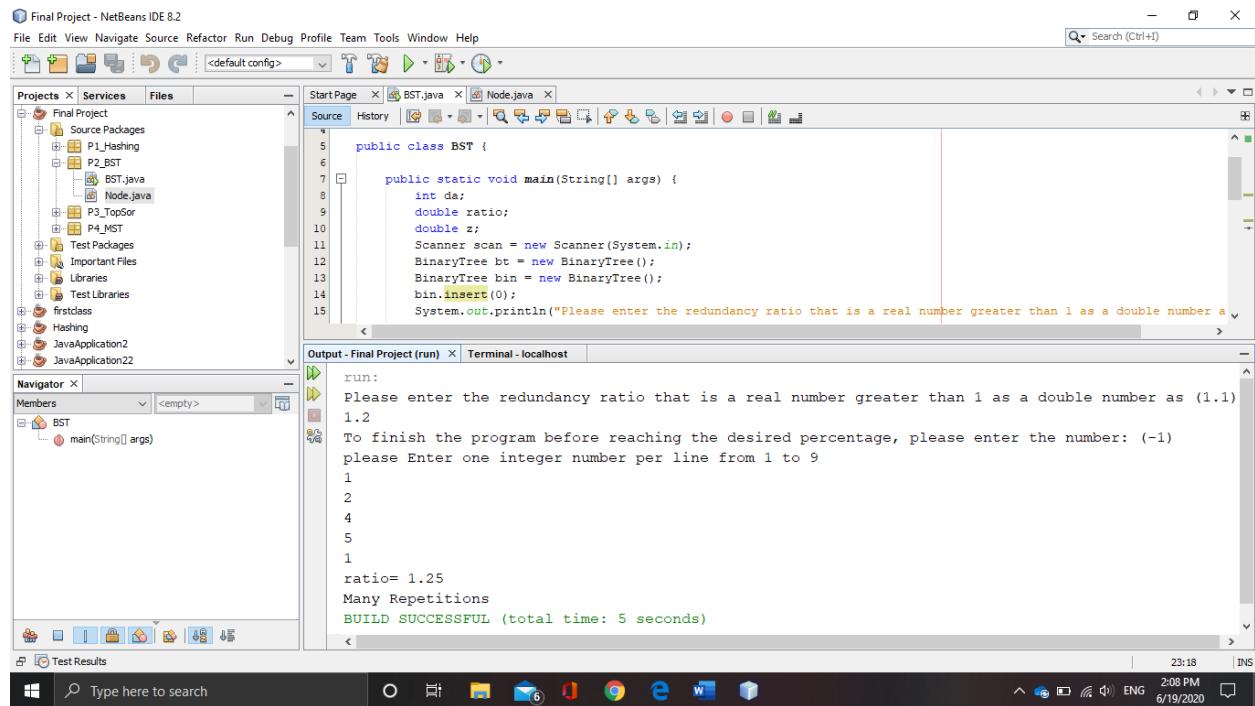
Here the project first sample test which had redundancy ratio 1.5 and the number 2 was repeated so the conclusion is that the ratio is 1.5 and it had Many repetitions.

And the output is : ratio= 1.5

Many Repetitions

Page ( 14 )

## Another sample Run



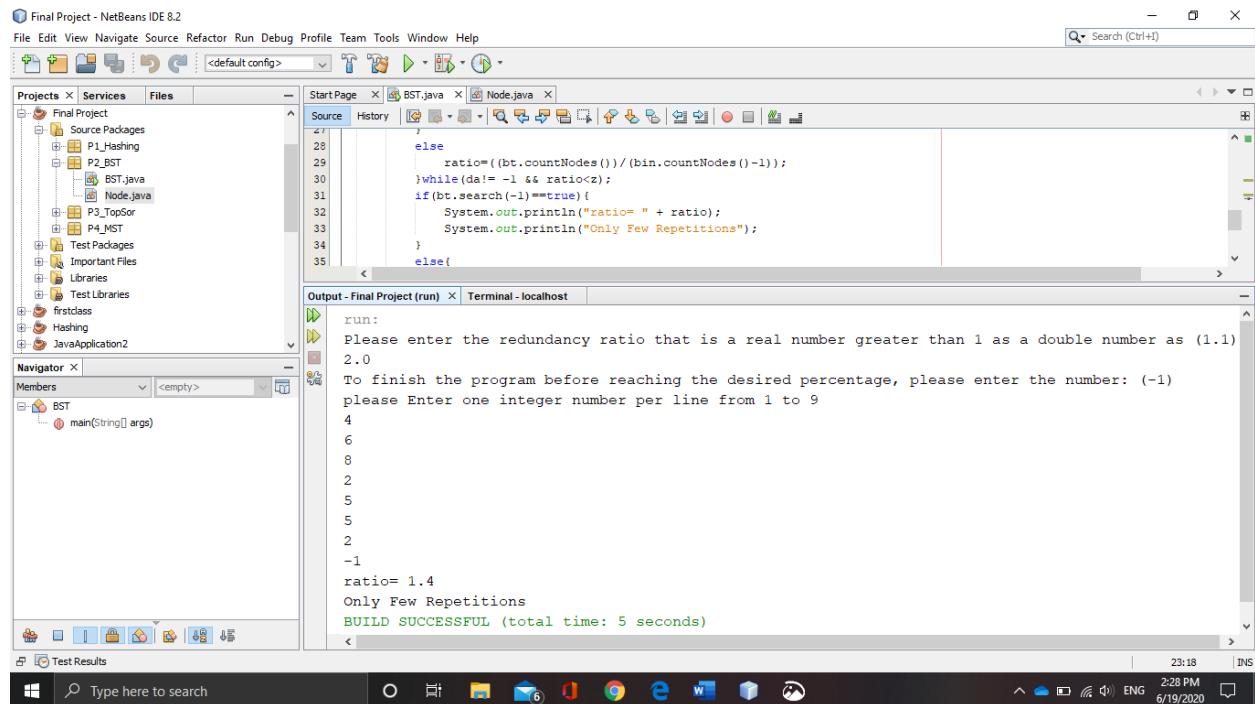
Here the project second sample test which had redundancy ratio 1.2 and the number 1 was repeated so the conclusion is that the ratio is 1.25 and it had Many repetitions.

And the output is : ratio= 1.25

# Many Repetitions

Page ( 15 )

## Another sample Run



The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The left sidebar has sections for Projects, Services, Files, and Navigator. The Projects section shows a 'Final Project' with several source packages like P1\_Hashing, P2\_BST, P3\_TopSort, P4\_MST, and Test Packages. The Navigator shows the 'main(String[] args)' method in the BST class. The main workspace displays two code files: BST.java and Node.java. BST.java contains Java code for calculating a ratio and printing it. The output window shows the program's execution, including user input and the resulting output: 'ratio= 1.4 Only Few Repetitions'. The bottom status bar shows the date and time.

Here the project second sample test which had redundancy ratio 2.0 and the number 5 was repeated as well as number 2 so the conclusion is that the ratio is 1.4 and it had Only Few repetitions. **And the output is :**

*ratio= 1.4*

*Only Few Repetitions*

**Page ( 16 )**

## Third Topological Sort: ( Problem Statement )

- *The third problem in this project which include the top sort and the problem that we faced was how to implement the topological sort as it's job was sorting so the problem was how to create a method that has the ability to get the most Node or Island that had the most paths.*
- *The second problem was the ability to make the user enter a finite and specific number for every bridge between the islands and don't allow him to enter an extra number or any thing of the sort which was solved by using arrays as usual.*

- Pseudocode :

```
int x=User;int y=user;int z=User;  
int ar[][]=new int[y][x];  
String To Get Most Visited="";  
for(int i=0;i<ar.length;i++){  
    int to,from;  
    to=User; from=User;  
    st+=from;  
    if(t>ar.size| |f>ar.size){  
        \\Error(terminate)  
    }else{      ar[to-1][from-1]=1; } }
```

Get the most visited ans = colMaxSum(ar);

int MV=ans.first+1;

int MP=gmo(st);

Page ( 18 )

## Complexity

### Time analysis :

*In the example given within the project the time completely depends on the user and how fast he can enter the information needed for this task so this example took 10 second so  $O(N+E)$  where  $N$  is the number of nodes that has to be implemented in the array as well as  $E$  which are the numbers of Edges(bridges) like topological sort.*

### Space analysis :

*This problem took some elements form the previous problems so we can say it's but since space and memory wasn't used a lot in the previous problem so it did not affect the performance of it completing the task so  $O(N)$  where  $N$  is the number of Islands*      **Page ( 19 )**

## Sample Runs

```
public static void main(String[] args) {
    Scanner sc=new Scanner (System.in);
    System.out.println("Please Enter the Nums of islands,Bridges,The initial Island");
    int x=sc.nextInt();int y=sc.nextInt();int z=sc.nextInt();
    int ar[][]=new int[y][x];
    String str="";
    for(int i=0;i<ar.length;i++){
        int t,f;
        t=sc.nextInt();f=sc.nextInt();
        str+=f;
    }
}
run:
Please Enter the Nums of islands,Bridges,The initial Island
7 9 1
1 2
1 3
2 4
2 5
3 6
4 7
5 7
6 5
6 7
The Island that has the Most chance to Get stuck on is
7
BUILD SUCCESSFUL (total time: 18 seconds)
```

In this example we used 7(islands), 9(bridges) and starting from island 1 ,The output was correct as their were 4 paths to get to island 7

That's why island 7 is where we get stuck on.

**The output :** The Island that has the Most chance to Get stuck on is

## Another Sample Run

```
public static void main(String[] args) {
    Scanner sc=new Scanner (System.in);
    System.out.println("Please Enter the Nums of islands,Bridges,The initial Island");
    int x=sc.nextInt();int y=sc.nextInt();int z=sc.nextInt();
    int ar[][]=new int[y][x];
    String str="";
    for(int i=0;i<ar.length;i++){
        int t,f;
        t=sc.nextInt();f=sc.nextInt();
        str+=t+" "+f+" ";
        if(t>ar.length||f>ar.length){
            System.out.println("Invalid Entry");
            System.out.println("Please Try Again");
        }
    }
}
```

Output - Topological-Sort (run) > Terminal - localhost

```
run:
Please Enter the Nums of islands,Bridges,The initial Island
5 7 1
1 2
1 3
1 4
1 5
2 4
2 5
3 4
The Island that has the Most chance to Get stuck on is
4
BUILD SUCCESSFUL (total time: 10 seconds)
```

This is the example from this project which took 5(islands) ,7(bridges) and starting from island 1 and the output was 4 as it has 3 paths and this is the most paths from any island to get to island 4 so we are stuck on island 4

**The output :** The Island that has the Most chance to Get stuck on is 4

**Page ( 21 )**

## Another Sample Run

The screenshot shows the NetBeans IDE interface with the following details:

- File Menu:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Search Bar:** Search (Ctrl+F)
- Projects Tab:** JavaApplicationSS, MST, sheet65, Stack, Topological-Sort (selected), Test Packages, Important Files, Libraries, Test Libraries, XoGame.
- Source Editor:** TopologicalSort.java (highlighted) and MostVisited.java. The code for TopologicalSort.java is as follows:

```
int MV=ans.first+1;
int MP=gmo(st);
if (MV==MP) {
    System.out.println("The Island that most likely to Get stuck on is");
    System.out.print(MV+" ");
    System.out.println(gmo(st));
} else {
    System.out.println("The Island that has the Most chance to Get stuck on is");
    System.out.println(gmo(st));
}
```

- Output Window:** Output - Topological-Sort (run) terminal - localhost. The output is:

```
run:
Please Enter the Nums of islands,Bridges,The initial Island
6 6 1
6 6 1
4 2
4 1
5 1
5 2
2 3
3 1
The Island that has the Most chance to Get stuck on is
1
BUILD SUCCESSFUL (total time: 16 seconds)
```
- System Tray:** Shows the date and time as 6/18/2020 8:32 PM.

Here we used 6(islands) ,6(bridges) and 1 to start with and it's clear that island 1 has more than a path so we are stuck on island 1

**The output :**

The Island that has the Most chance to Get stuck on is

## Fourth Minimum Spanning Trees :

### ( Problem Statement )

- The fourth and final problem in this project to start the first problem that we faced in this problem was to find a method that capable of identify the nodes that has more than one edge.
- The second problem was to make a method that could calculate the cost of removing edges and to make the minimum spanning tree.

- Pseudocode :

```
int MethodCalculateCost ( Edge1, Edge2 ) {  
    int cal the difference =Edge1-Edge2;  
  
    int ctd=cal the difference;  
  
    Get the Absoulte value for( ctd )  
  
    return ctd; }  
  
Main Method (String[] args) {  
    int Node =User ; int Edge=User ;  
  
    int Array of Nodes=new int[Node][Edge];  
  
    for (int i=0;i->aofn.size;i++){ int x , y;  
  
        x=user from before ; y=user from before;  
  
        if(x>aofn.size(OR)y>aofn.size){then error}  
  
        else{  
  
            aofn[x-1][y-1]=1;  
  
            aofn[y-1][x-1]=1; } }  
}
```

- Pseudocode(Cont) :

Method to get max node=colMaxSum(aofn);

int MaxNode=ans.first+1;

int temporaryMaxCostForRemoving=10;

for(int i=0;i<=aofn.size-1;i++){

    if(element in the array==1){

        if(cost(i,mn)<=tempmcfr){

            display MinimumCost(cost(i,mn));     break;

    }else{

        Tempmcfr++;}

    }else{

        continue; } }

## Complexity

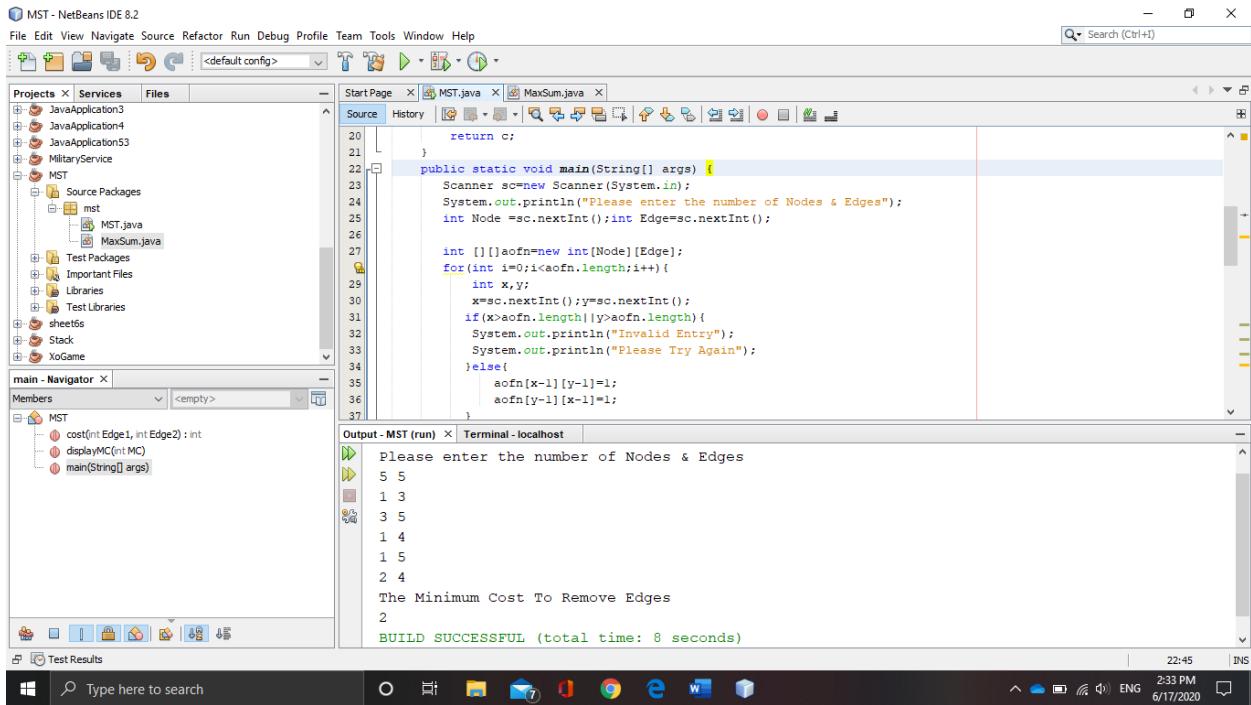
### Time analysis :

We Know that Minimum spanning tree can be implemented by the two known algorithms which are Kruskal and prim but like the hashing problem it depends on the user input like the sample test in the project took 9 seconds since it only uses arrays so that helped in reducing time so  $O(n^2)$  which n is the number of nodes.

### Space analysis :

It only uses arrays for calculations such that method that calculate the cost of removing unnecessary edges and method that display the result but also it depends on the input of the user and to make sure it was MST by checking the number of edges which equal to (nodes-1)

## Sample Runs

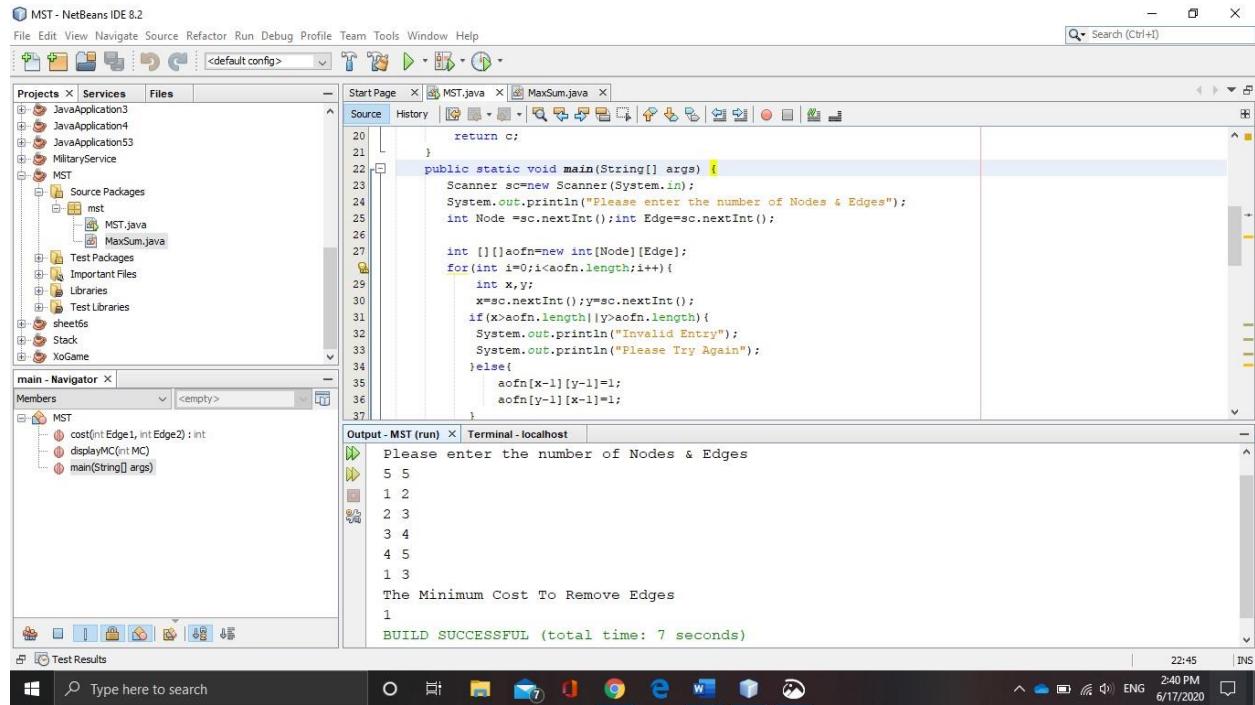


This sample run took 5 Nodes and 5 Edges and I intended to make the distance number obvious from each node to others and the minimum cost was 2 and it's correct.

## The output :

# The Minimum Cost To Remove Edges

## Another Sample Run



MST - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files <default config> Start Page MST.java MaxSum.java

Source History

```
20     }
21     return c;
22   public static void main(String[] args) {
23     Scanner sc=new Scanner(System.in);
24     System.out.println("Please enter the number of Nodes & Edges");
25     int Node =sc.nextInt();int Edge=sc.nextInt();
26
27     int [][]aofn=new int [Node][Edge];
28     for(int i=0;i<aofn.length;i++){
29       int x,y;
30       x=sc.nextInt();y=sc.nextInt();
31       if(x>aofn.length||y>aofn.length){
32         System.out.println("Invalid Entry");
33         System.out.println("Please Try Again");
34       }else{
35         aofn[x-1][y-1]=1;
36         aofn[y-1][x-1]=1;
37     }
38   }
39   Output - MST (run) Terminal - localhost
40 Please enter the number of Nodes & Edges
41 5 5
42 1 2
43 2 3
44 3 4
45 4 5
46 1 3
47 The Minimum Cost To Remove Edges
48 1
49 BUILD SUCCESSFUL (total time: 7 seconds)
```

Output - MST (run) Terminal - localhost

Please enter the number of Nodes & Edges

5 5

1 2

2 3

3 4

4 5

1 3

The Minimum Cost To Remove Edges

1

BUILD SUCCESSFUL (total time: 7 seconds)

22:45 6/17/2020

Here the project sample test which took 5 Nodes and 5 Edges and the distance number between nodes was either 1 or 2 but since we want the minimum cost we of course chose 1

**The output :**

The Minimum Cost To Remove Edges

1

**Page ( 28 )**

## Another Sample Run

MST - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files Start Page MST.java MaxSum.java

Source History

```
20     }
21     return c;
22   }
23   public static void main(String[] args) {
24     Scanner sc=new Scanner(System.in);
25     System.out.println("Please enter the number of Nodes & Edges");
26     int Node =sc.nextInt();int Edge=sc.nextInt();
27
28     int [][]aofn=new int [Node] [Edge];
29     for(int i=0;i<aofn.length;i++){
30       int x,y;
31       x=sc.nextInt();y=sc.nextInt();
32       if(x>aofn.length||y>aofn.length){
33         System.out.println("Invalid Entry");
34         System.out.println("Please Try Again");
35       }else{
36         aofn[x-1][y-1]=1;
37         aofn[y-1][x-1]=1;
38     }
39   }
40   Please enter the number of Nodes & Edges
41   6 6
42   1 2
43   2 4
44   4 1
45   6 3
46   3 5
47   5 1
48   The Minimum Cost To Remove Edges
49   1
50   BUILD SUCCESSFUL (total time: 10 seconds)
```

Output - MST (run) Terminal - localhost

main - Navigator Members

MST

Test Results

Type here to search

37:12 3:02 PM ENG 6/17/2020

Here the final sample run which took 6 Nodes and 6 edges and we mixed the distance numbers between the nodes such that costs equal to 1,2,3 and even 4 but we surely chose the minimum cost which is 1

The output :The Minimum Cost To Remove  
Edges 1 Page ( 29 )

*IN The End*

We would like to Give a Special Thanks

*To*

Prof. Dr. Amr El Masry

Dr. Mervat Mikhail

*And Every instructor & Engineer*

*That helped us during this semester*

**THANK YOU**