The Report for Data Structure Project

Student (1) :

أحمد ياسر محمد محمد نصار

ID : 20191616316

Student (2) :

عطيه فتحي عطيه أحمد محمد

ID : 20191480940

This project was made by only two individuals.

Student (1) :

أحمد ياسر محمد محمد نصار .

Contributed to this project by solving :

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

And maker of this report

Student (2) :

عطيه فتحي عطيه أحمد محمد .

Contributed to this project by solving :

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

And helped in the making of this report.

Contents

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 ->to  sa []size ;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

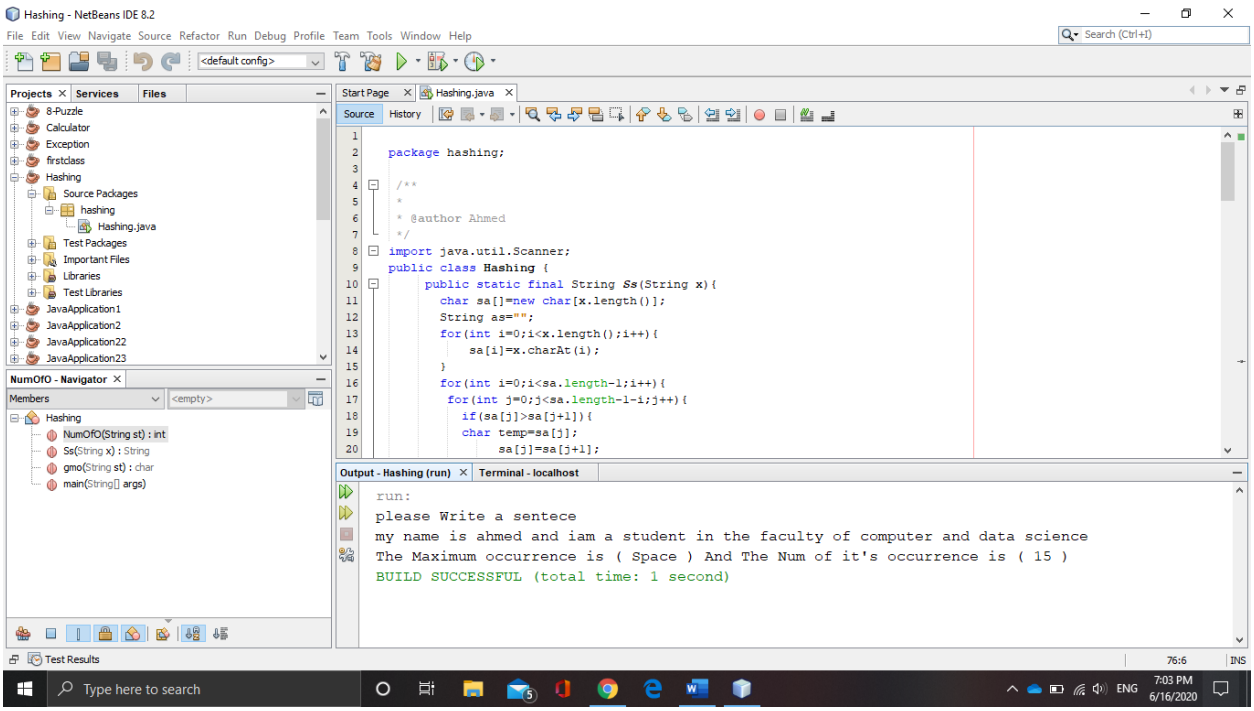
(Faculty of Computers and Data Science *) as a test*

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.

Sample Runs

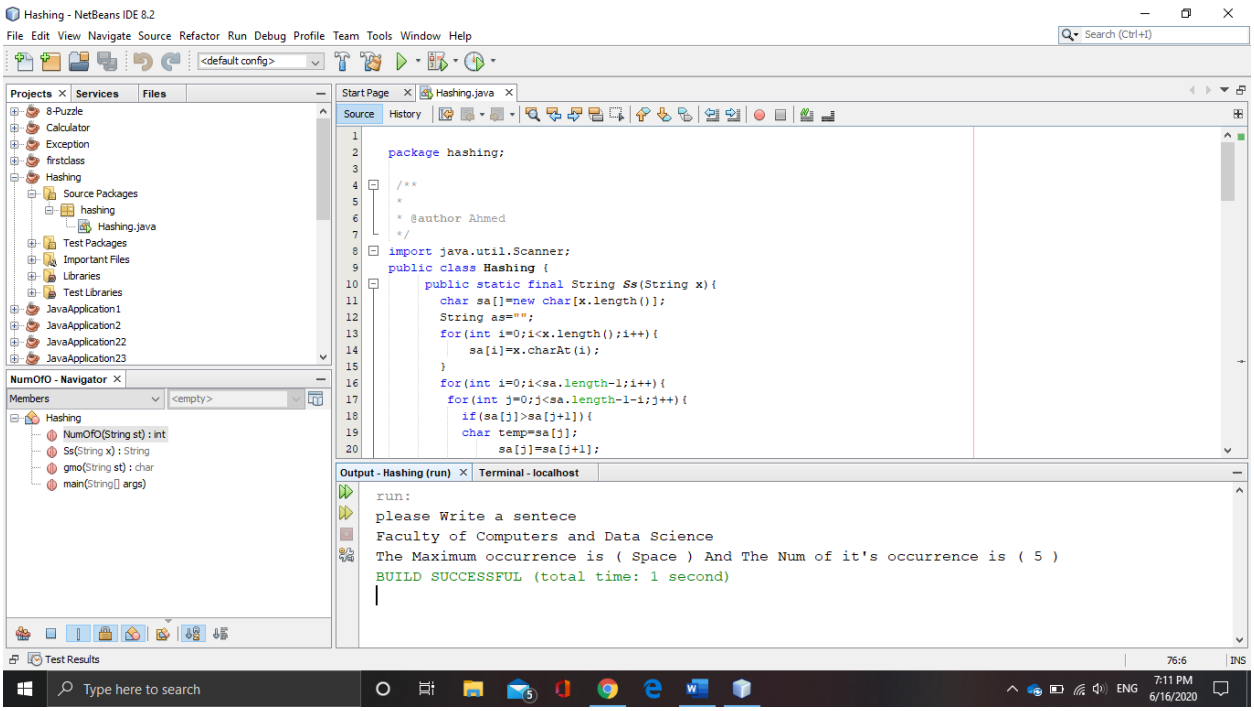


This sample run has the sentence

(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)*

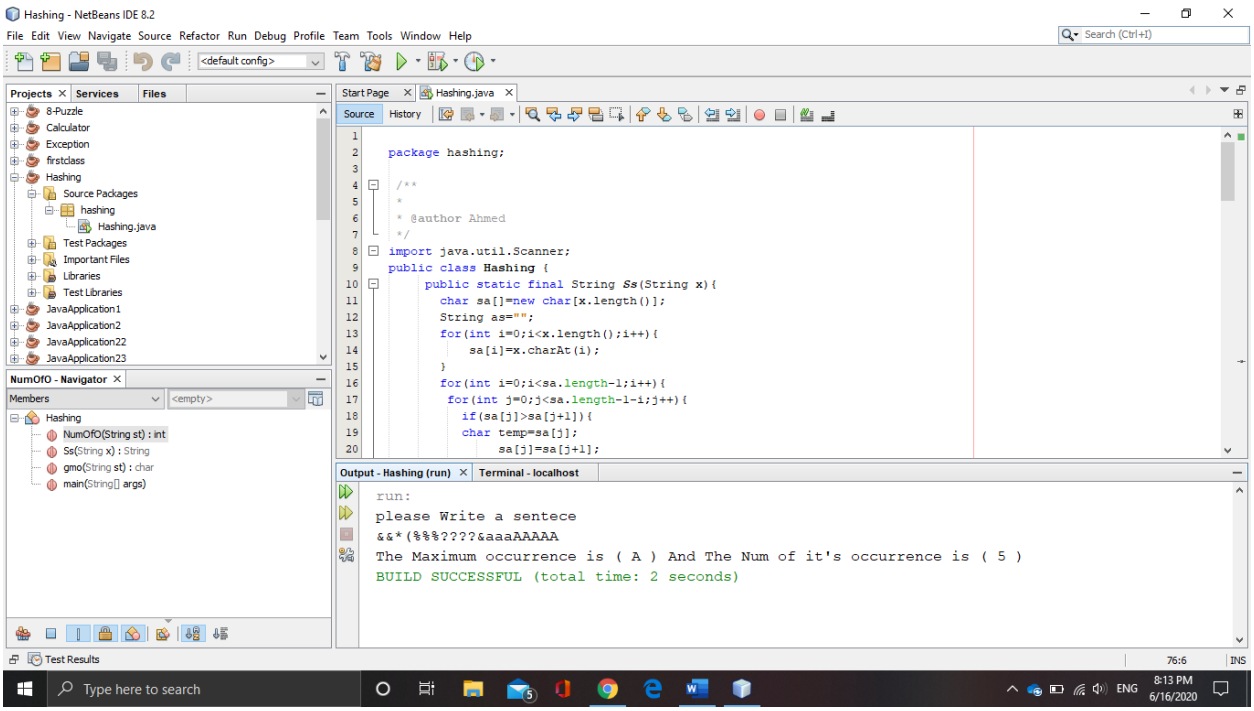
Another sample Run



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 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

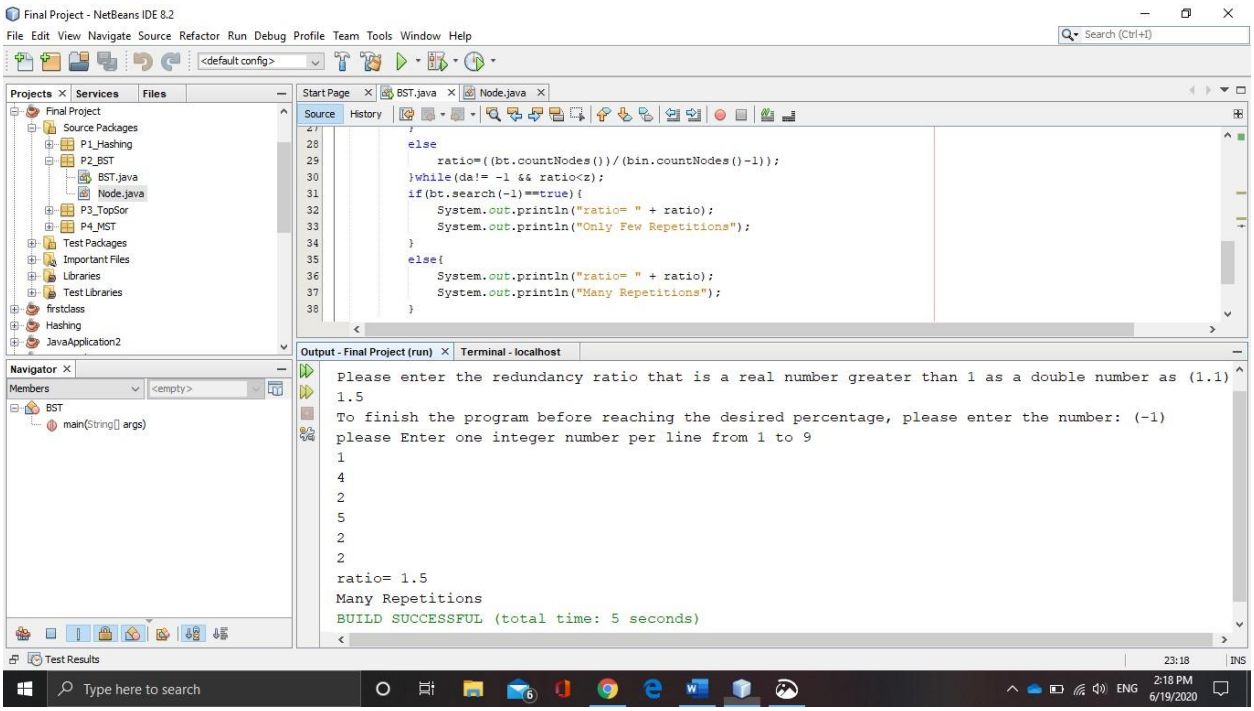
Time analysis :

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

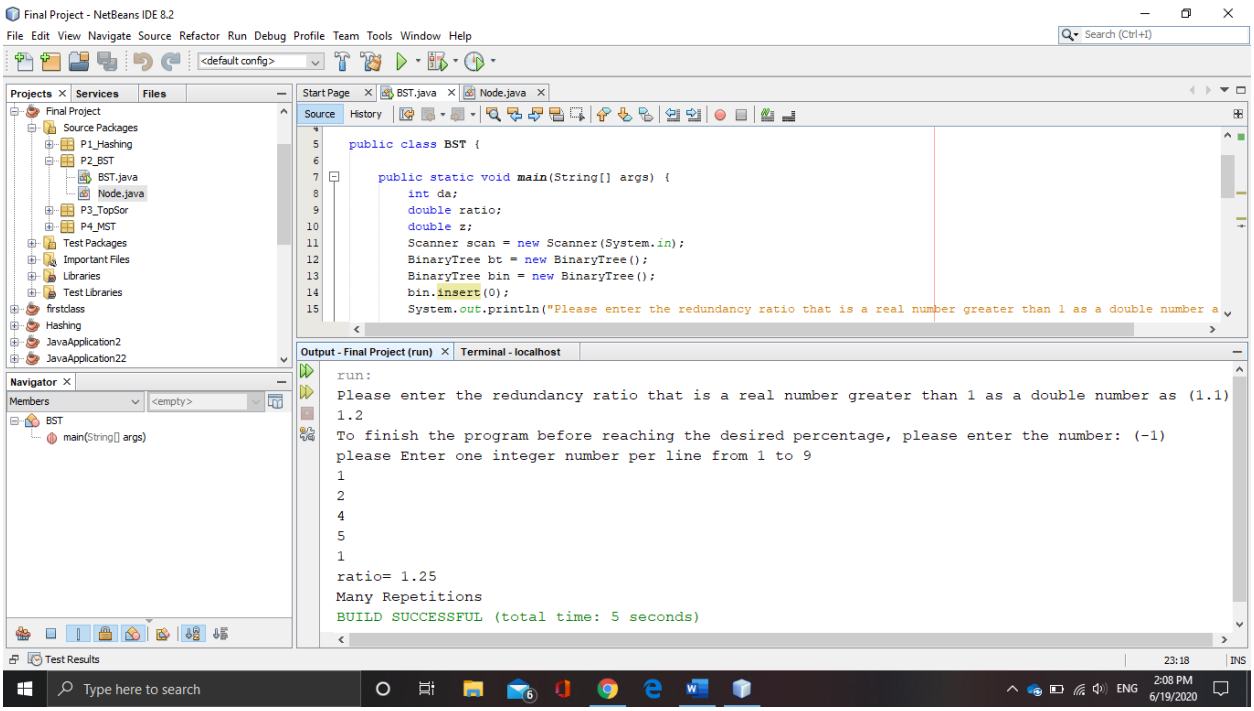


Here the project first sample test which had redundancy ratio 1.5 and the number 2 was repeated so the conclusion is that the ratio is 1.5 and it had Many repetitions.

And the output is : ratio= 1.5

Many Repetitions

Another sample Run

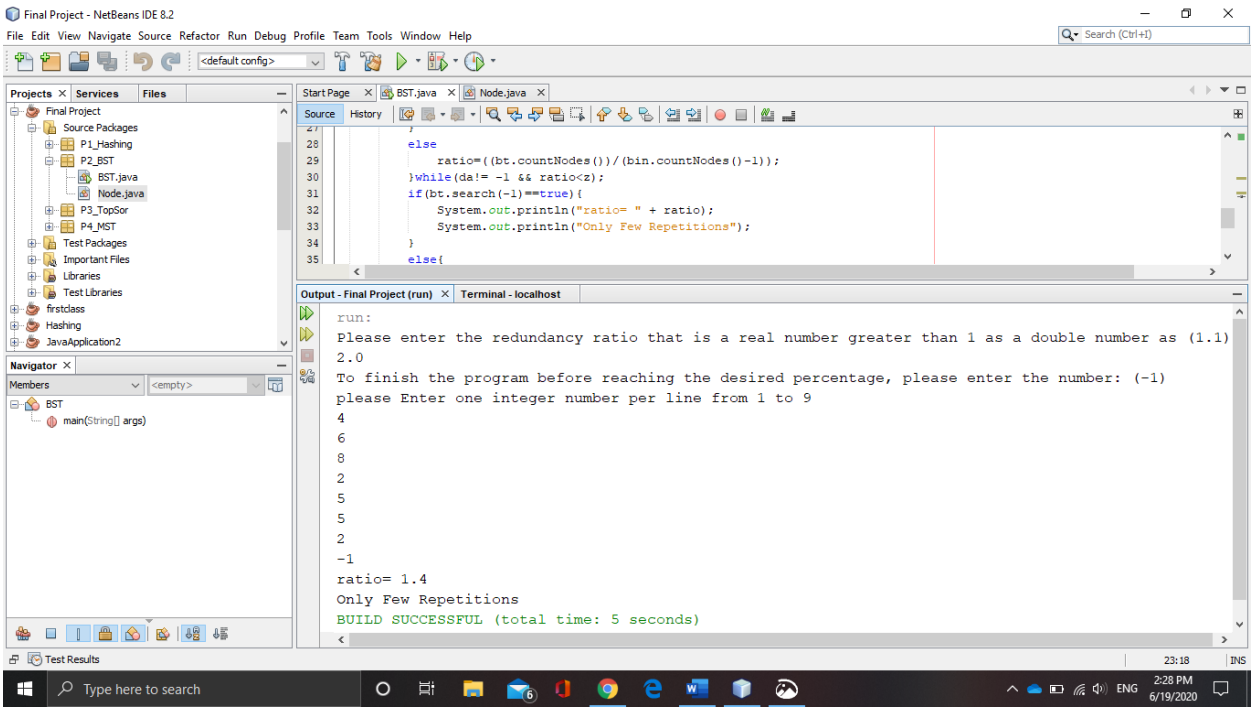


Here the project second sample test which had redundancy ratio 1.2 and the number 1 was repeated so the conclusion is that the ratio is 1.25 and it had Many repetitions.

And the output is : ratio= 1.25

Many Repetitions

Another sample Run



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

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);
```

Complexity

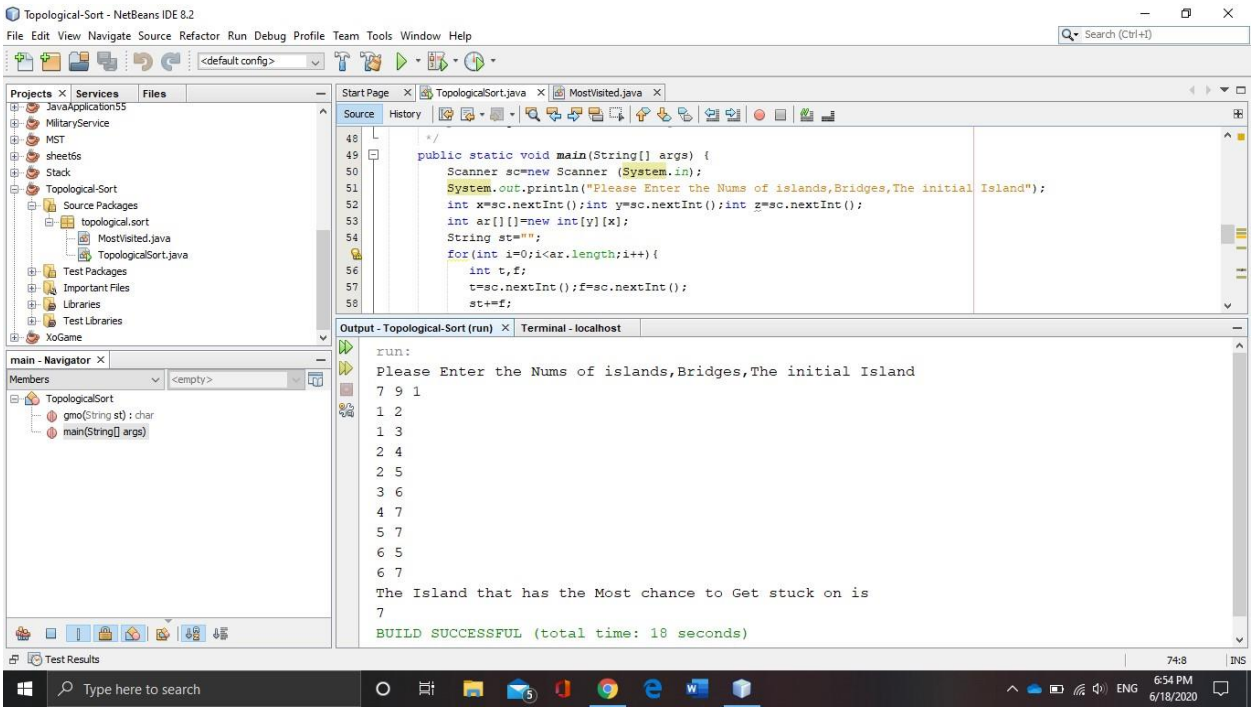
Time analysis :

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

Sample Runs



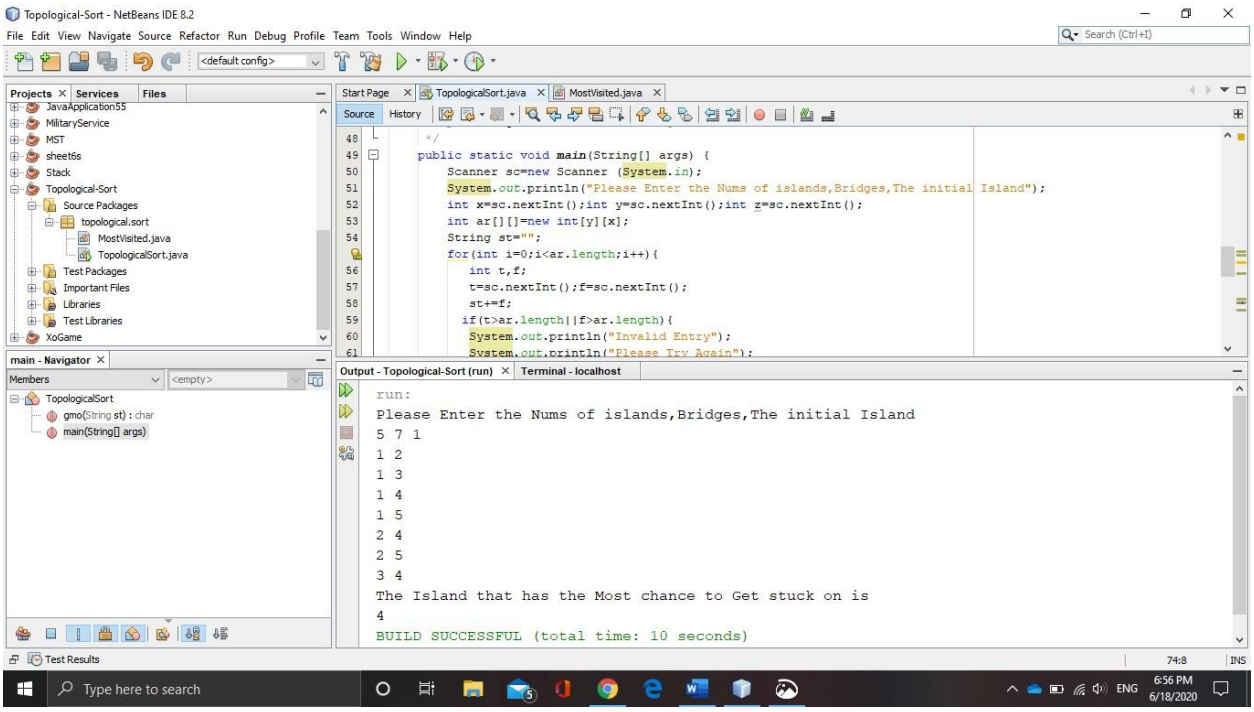
In this example we used 7(islands), 9(bridges) and starting from island 1 ,The output was correct as their were 4 paths to get to island 7

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

7

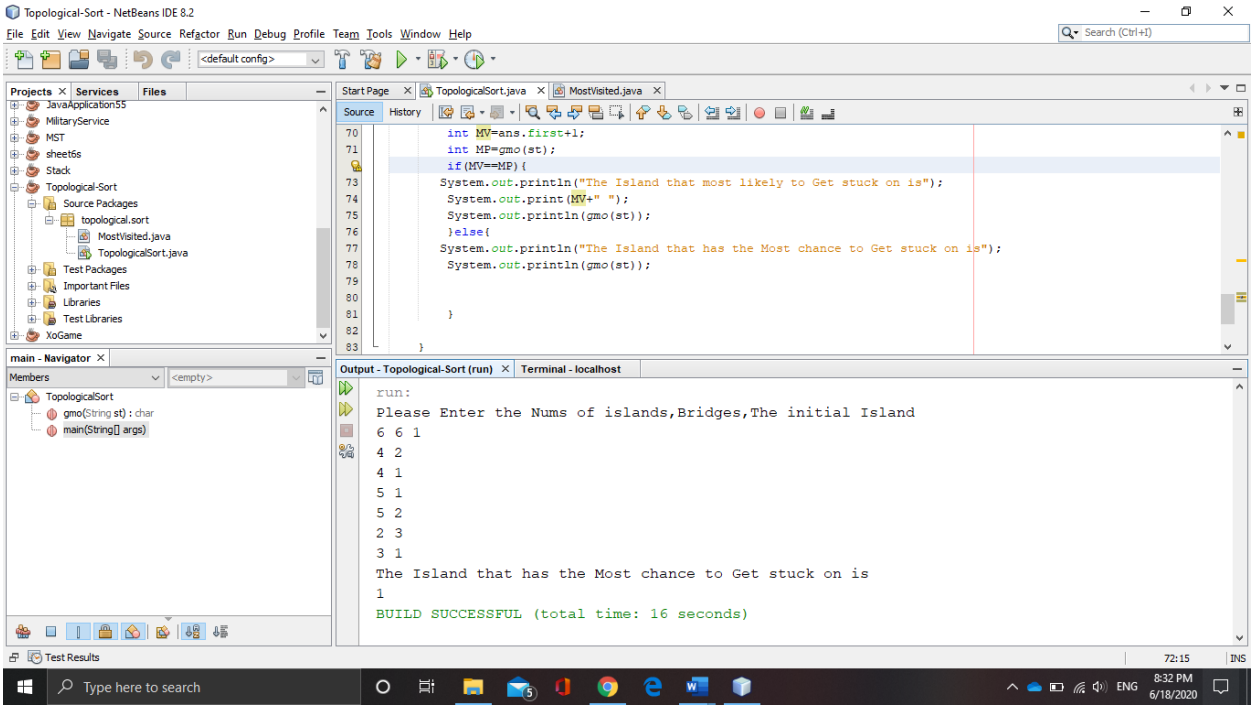
Another Sample Run



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 toget 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

Another Sample Run



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

1

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

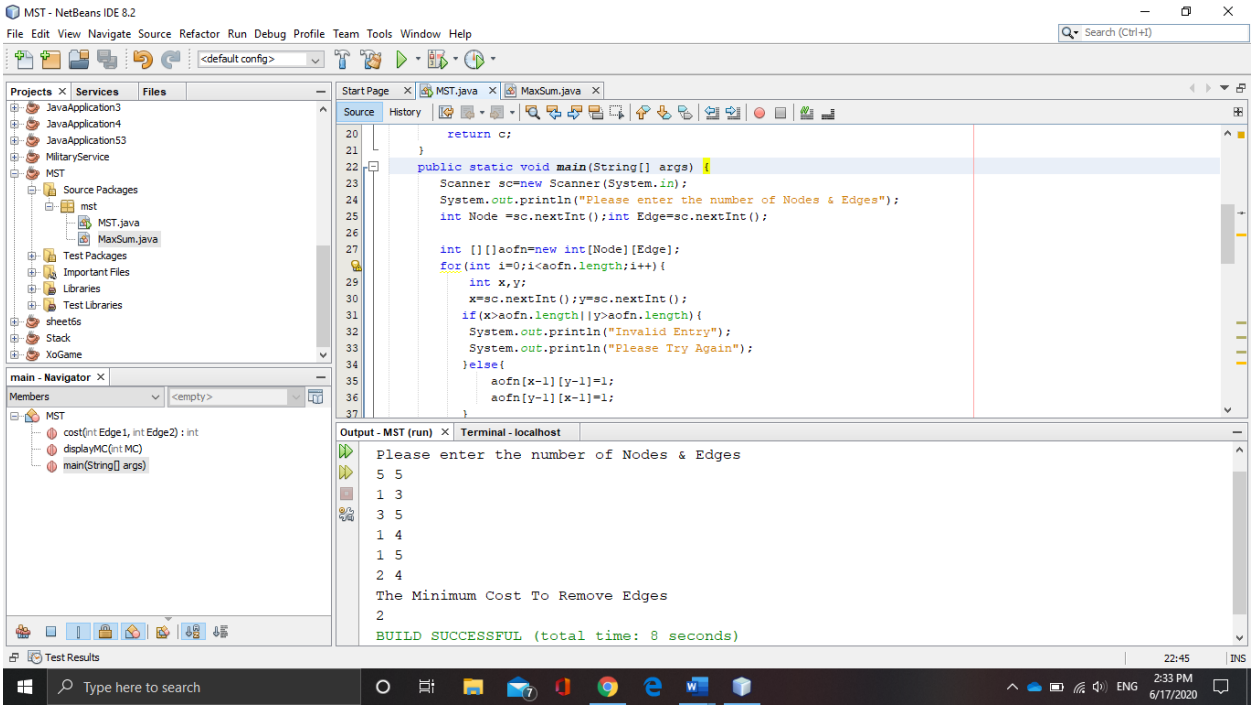
Time analysis :

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



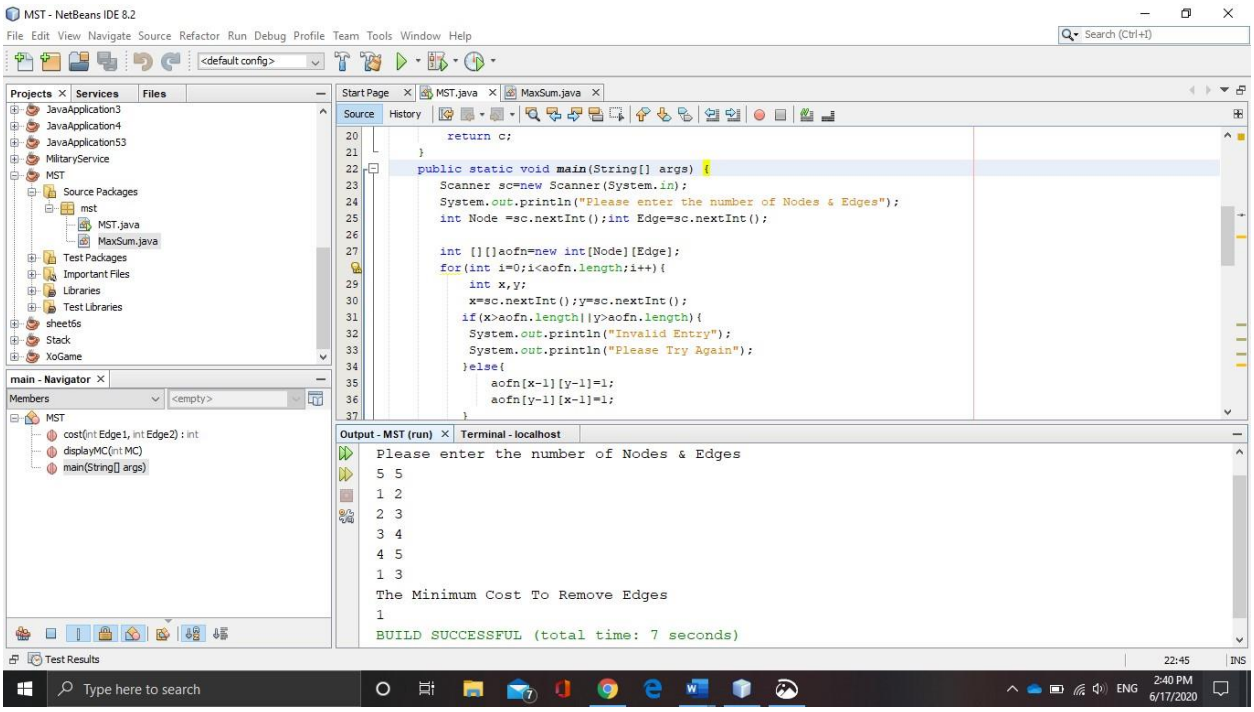
This sample run took 5 Nodes and 5 Edges and I intended to make the distance number obvious from each node to others and the minimum cost was 2 and it ’s correct.

The output :

The Minimum Cost To Remove Edges

2

Another Sample Run



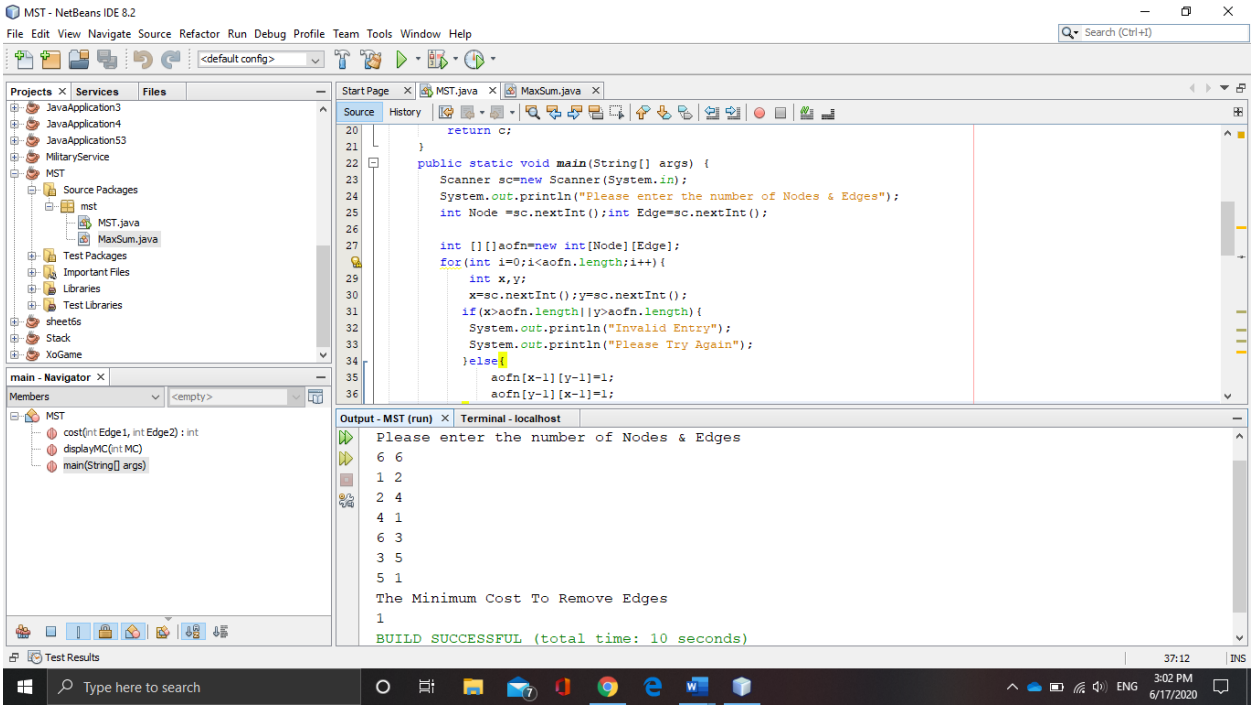
Here the project sample test which took 5 Nodes and 5 Edges and the distance number between nodes was either 1 or 2 but since we want the minimum cost we of course chose 1

The output :

The Minimum Cost To Remove Edges

1

Another Sample Run



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 surly chose the minimum cost which is 1

The output :The Minimum Cost To Remove Edges 1

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