15.05.2023

CSE1120 Discrete Structures

2022-2023 SPRING SEMESTER

COMPUTER PROJECT

Deadline: 02.06.2023/23:00

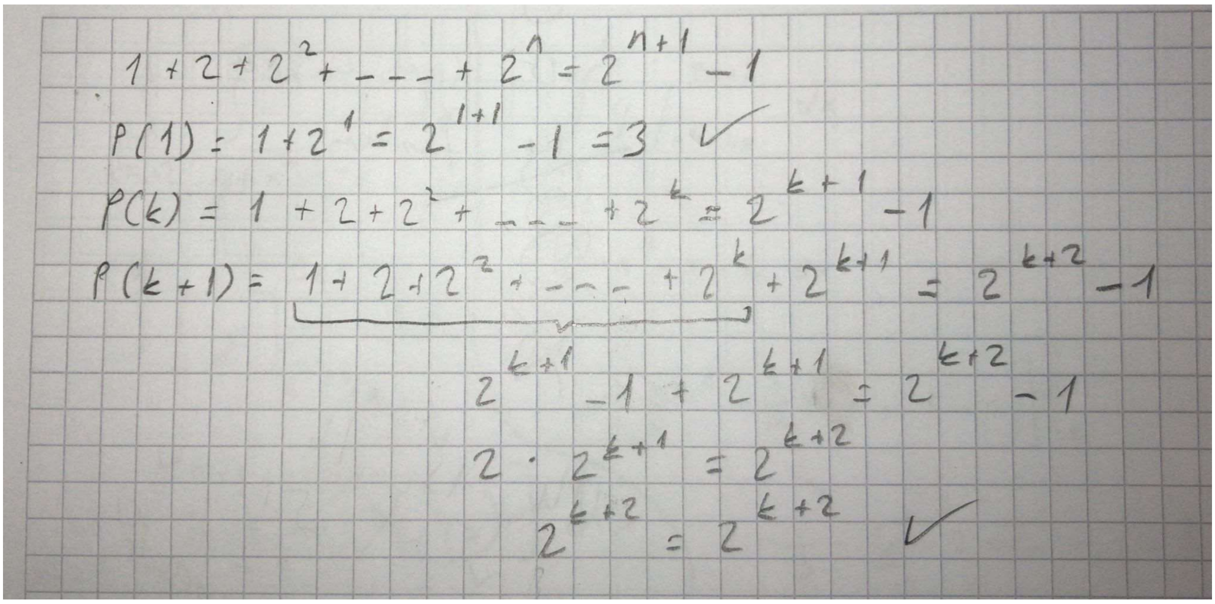
- **Upload your project to Mic. Teams.**
- **You have to be a group of 2 students for this project.**
- **Please write the solutions and codes of each quesitons that included the screenshots of the outputs of the programs to the report.**
- **Report cover page is indicated below.**
- **Please upload the source files of each question as .rar or .zip file.**

Question 1:

a) Use mathematical induction to show that

$$1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$$

for all nonnegative integers *n*.



$1+2+2^2+---+2^n = 2^{n+1}-1$

$P(1) = 1+2^1 = 2^{1+1}-1 = 3$ ✓

$P(k) = 1+2+2^2+---+2^k = 2^{k+1}-1$

$P(k+1) = \underbrace{1+2+2^2+---+2^k} + 2^{k+1} = 2^{k+2}-1$

$2^{k+1}-1+2^{k+1} = 2^{k+2}-1$

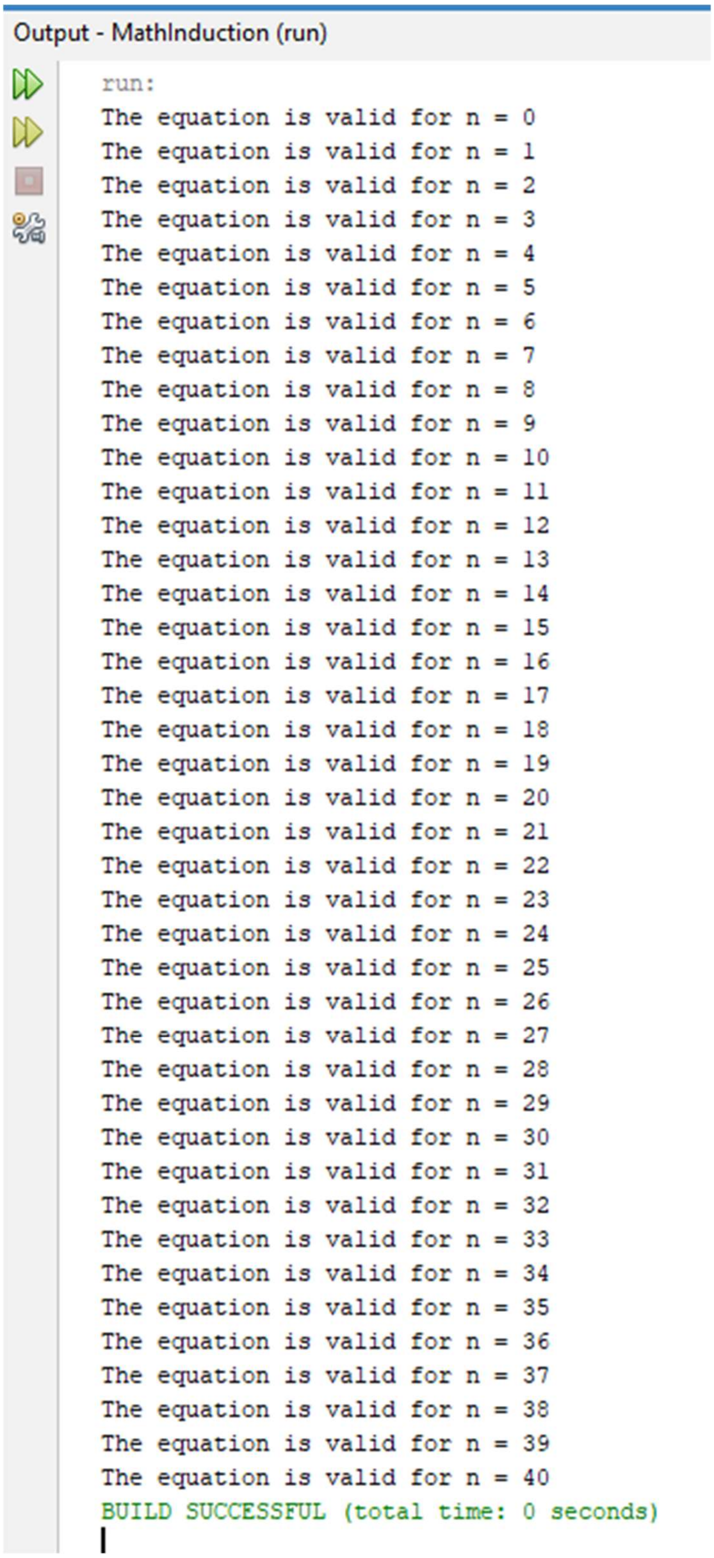
$2\cdot 2^{k+1} = 2^{k+2}$

$2^{k+2} = 2^{k+2}$ ✓

b) Write a computer program that validates this equation.

```
public class Main {
  public static void main(String[] args) {
    // Test the equation for different values of n
    for (int n = 0; n <= 40; n++) {
      int sum = calculateSum(n);
      int rhs = calculateRHS(n);

      if (sum == rhs) {
        System.out.println("The equation is valid for n = " + n);
```

```
            } else {
                System.out.println("The equation is NOT valid for n = " + n);
            }
        }
    }

    // Calculate the sum of the series using a loop
    public static int calculateSum(int n) {
        int sum = 0;
        for (int i = 0; i <= n; i++) {
            sum += Math.pow(2, i);
        }
        return sum;
    }

    // Calculate the right-hand side of the equation
    public static int calculateRHS(int n) {
        return (int) (Math.pow(2, n + 1) - 1);
    }
}
```

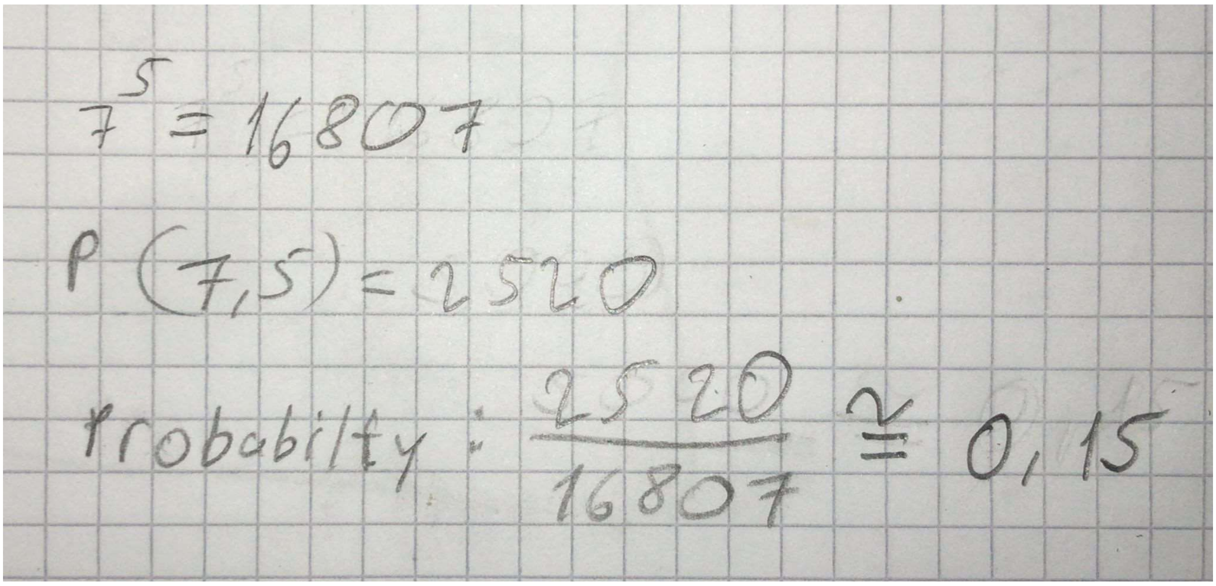


```
run:
The equation is valid for n = 0
The equation is valid for n = 1
The equation is valid for n = 2
The equation is valid for n = 3
The equation is valid for n = 4
The equation is valid for n = 5
The equation is valid for n = 6
The equation is valid for n = 7
The equation is valid for n = 8
The equation is valid for n = 9
The equation is valid for n = 10
The equation is valid for n = 11
The equation is valid for n = 12
The equation is valid for n = 13
The equation is valid for n = 14
The equation is valid for n = 15
The equation is valid for n = 16
The equation is valid for n = 17
The equation is valid for n = 18
The equation is valid for n = 19
The equation is valid for n = 20
The equation is valid for n = 21
The equation is valid for n = 22
The equation is valid for n = 23
The equation is valid for n = 24
The equation is valid for n = 25
The equation is valid for n = 26
The equation is valid for n = 27
The equation is valid for n = 28
The equation is valid for n = 29
The equation is valid for n = 30
The equation is valid for n = 31
The equation is valid for n = 32
The equation is valid for n = 33
The equation is valid for n = 34
The equation is valid for n = 35
The equation is valid for n = 36
The equation is valid for n = 37
The equation is valid for n = 38
The equation is valid for n = 39
The equation is valid for n = 40
BUILD SUCCESSFUL (total time: 0 seconds)
```

Question 2:

$A = \{1, 2, 3, 4 ,5\}$ and $B = \{t, u, v, w, x, y, z\}$

a) If a fuction $f: A \to B$ is randomly generated, what is th probability that it is one to one?



$7^5 = 16807$

$P(7,5) = 2520$

Probabilty : $\frac{2520}{16807} \cong 0,15$

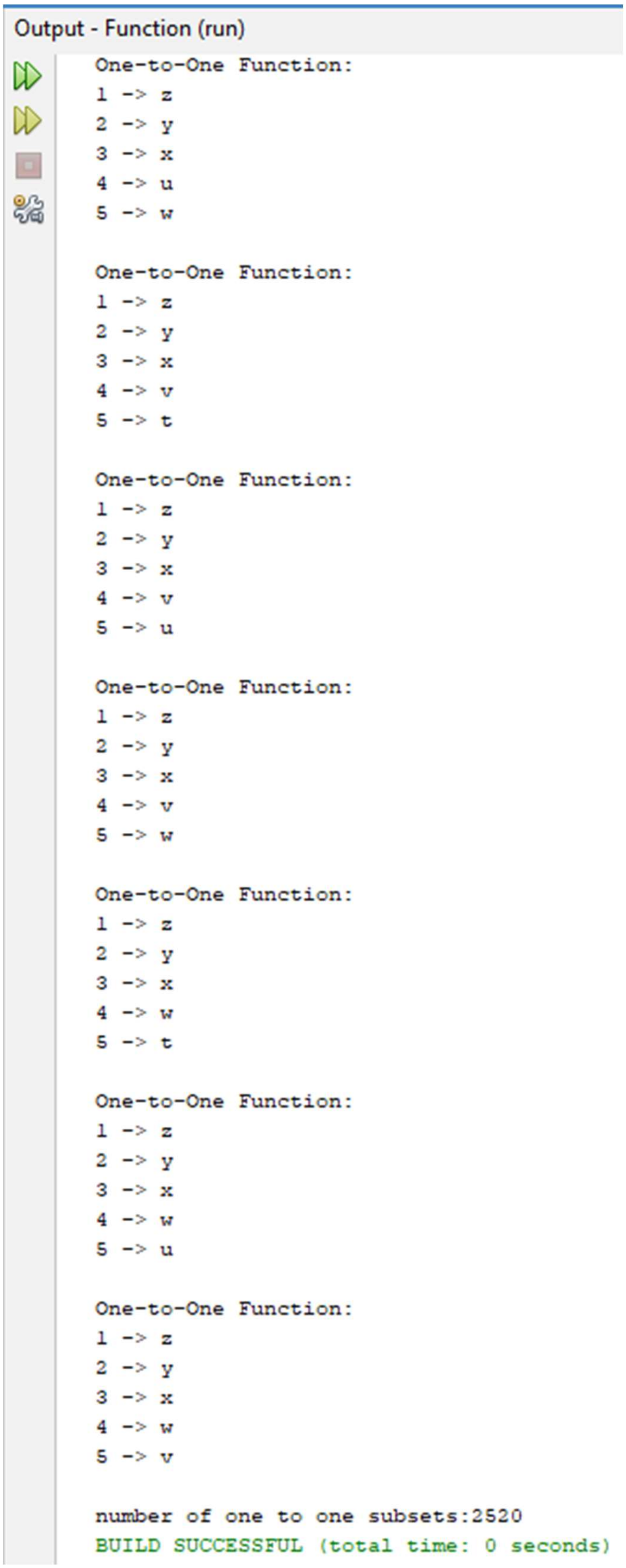
b) Write a computer program to generate random functions $f: A \rightarrow B$ and have the program print out how many functions it generates that is one to one?

```
import java.util.ArrayList;

import java.util.List;


public class Main {


   public static void main(String[] args) {

      List<Integer> setA = new ArrayList<>();

      setA.add(1);

      setA.add(2);

      setA.add(3);

      setA.add(4);

      setA.add(5);


      List<Character> setB = new ArrayList<>();

      setB.add('t');

      setB.add('u');

      setB.add('v');
```

```
        setB.add('w');
        setB.add('x');
        setB.add('y');
        setB.add('z');

        List<List<Pair>> oneToOneFunctions = generateOneToOneFunctions(setA, setB);
        int count=0;
        for (List<Pair> function : oneToOneFunctions) {
            System.out.println("One-to-One Function:");
            count++;
            for (Pair pair : function) {
                System.out.println(pair.key + " -> " + pair.value);
            }
            System.out.println();
        }
        System.out.println("Number of one to one subsets:"+count);
    }

    private static List<List<Pair>> generateOneToOneFunctions(List<Integer> setA, List<Character> setB) {
        List<List<Pair>> result = new ArrayList<>();
        generateOneToOneFunctionsHelper(setA, setB, new ArrayList<>(), result);
        return result;
    }

    private static void generateOneToOneFunctionsHelper(List<Integer> setA, List<Character> setB, List<Pair> currentFunction, List<List<Pair>> result) {
        if (currentFunction.size() == setA.size()) {
```

```
    result.add(new ArrayList<>(currentFunction));
    return;
  }

  for (Character element : setB) {
    if (!isElementUsed(element, currentFunction)) {
      currentFunction.add(new Pair(setA.get(currentFunction.size()), element));
      generateOneToOneFunctionsHelper(setA, setB, currentFunction, result);
      currentFunction.remove(currentFunction.size() - 1);
    }
  }
}

private static boolean isElementUsed(Character element, List<Pair> currentFunction) {
  for (Pair pair : currentFunction) {
    if (pair.value == element) {
      return true;
    }
  }
  return false;
}

private static class Pair {
  int key;
  char value;

  Pair(int key, char value) {
```

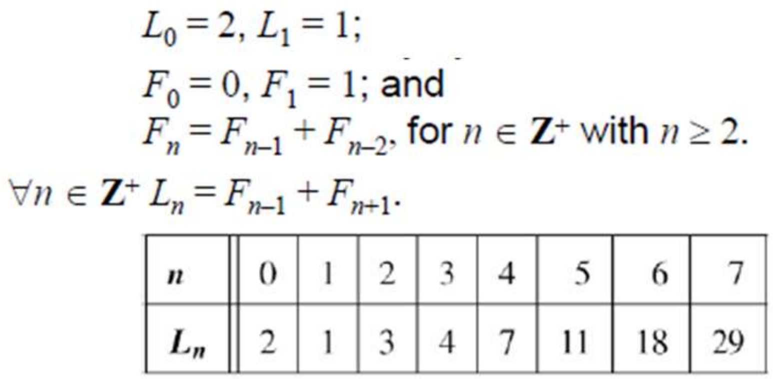
```
            this.key = key;
            this.value = value;
        }
    }
}
```



```
One-to-One Function:
1 -> z
2 -> y
3 -> x
4 -> u
5 -> w

One-to-One Function:
1 -> z
2 -> y
3 -> x
4 -> v
5 -> t

One-to-One Function:
1 -> z
2 -> y
3 -> x
4 -> v
5 -> u

One-to-One Function:
1 -> z
2 -> y
3 -> x
4 -> v
5 -> w

One-to-One Function:
1 -> z
2 -> y
3 -> x
4 -> w
5 -> t

One-to-One Function:
1 -> z
2 -> y
3 -> x
4 -> w
5 -> u

One-to-One Function:
1 -> z
2 -> y
3 -> x
4 -> w
5 -> v

number of one to one subsets:2520
BUILD SUCCESSFUL (total time: 0 seconds)
```

Question 3:

Write a recursive method to generate Lucas numbers.



$L_0 = 2, L_1 = 1;$

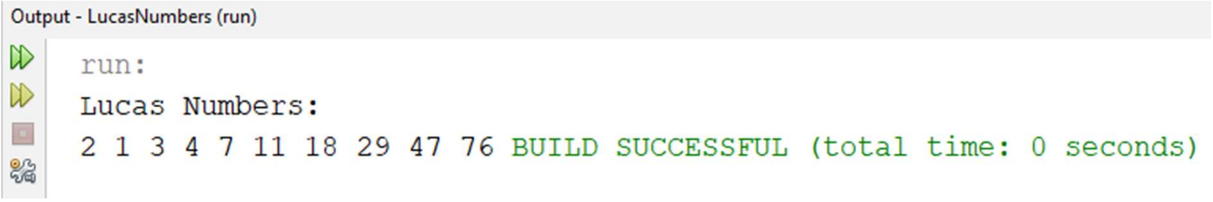
$F_0 = 0, F_1 = 1;$ and

$F_n = F_{n-1} + F_{n-2},$ for $n \in \mathbf{Z}^+$ with $n \geq 2.$

$\forall n \in \mathbf{Z}^+ \; L_n = F_{n-1} + F_{n+1}.$

n	0	1	2	3	4	5	6	7
L_n	2	1	3	4	7	11	18	29

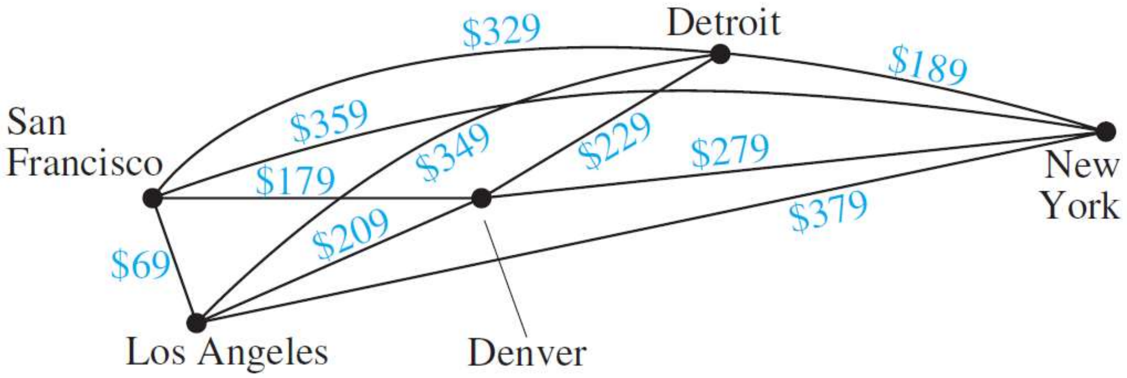
```
public class Main {


  public static void main(String[] args) {
    int n = 10;
    System.out.println("Lucas Numbers:");
    for (int i = 0; i < n; i++) {
      System.out.print( generateLucasNumber(i) + " ");
    }
  }


  public static int generateLucasNumber(int n) {
    if (n == 0) {
      return 2;
    } else if (n == 1) {
      return 1;
    } else {
      return generateLucasNumber(n - 1) + generateLucasNumber(n - 2);
    }
  }
}
```



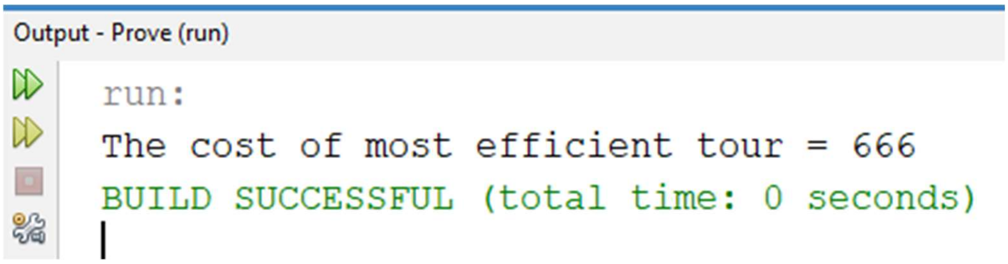
```
run:
Lucas Numbers:
2 1 3 4 7 11 18 29 47 76 BUILD SUCCESSFUL (total time: 0 seconds)
```

Question 4:

Write a computer program to find a route with the least total airfare that visits each of the cities in this graph, where the weight on an edge is the least price available for a flight between the two cities.



```
import java.io.*;

import java.util.*;


public class Prove {

   // there are four nodes in example graph (graph is

   // 1-based)


   static int n = 4;

   // give appropriate maximum to avoid overflow


   static int MAX = 1000000;


   // dist[i][j] represents shortest distance to go from i
   // to j this matrix can be calculated for any given
   // graph using all-pair shortest path algorithms
   static int[][] dist = {
                 {  0, 0, 0, 0,0 },    { 0, 0, 69, 179,189 },
            { 0, 69, 0, 209, 209 }, {0, 179, 209, 0, 229 },
            { 0, 189, 209, 209, 0 },
   };

   // memorization for top down recursion

   static int[][] memo = new int[n + 1][1 << (n + 1)];

   static int fun(int i, int mask)
   {
     // base case
     // if only i.th bit and 1st bit is set in our mask,
     // it implies we have visited all other nodes
     // already
     if (mask == ((1 << i) | 3))
       return dist[1][i];
     // memoization
     if (memo[i][mask] != 0)
       return memo[i][mask];

     int res = MAX; // result of this sub-problem

     // we have to travel all nodes j in mask and end the
     // path at ith node so for every node j in mask,
     // recursively calculate cost of travelling all
     // nodes in mask
     // except i and then travel back from node j to node
     // i taking the shortest path take the minimum of
     // all possible j nodes

     for (int j = 1; j <= n; j++)
       if ((mask & (1 << j)) != 0 && j != i && j != 1)
         res = Math.min(res,
                 fun(j, mask & (~(1 << i)))
                   + dist[j][i]);
     return memo[i][mask] = res;
   }

   // Driver program to test above logic
   public static void main(String[] args)
   {
     int ans = MAX;
     for (int i = 1; i <= n; i++)
       // try to go from node 1 visiting all nodes in
       // between to i then return from i taking the
       // shortest route to 1
       ans = Math.min(ans, fun(i, (1 << (n + 1)) - 1)
                   + dist[i][1]);

     System.out.println(
           "The cost of most efficient tour = " + ans);

     }

   }
```



```
run:
The cost of most efficient tour = 666
BUILD SUCCESSFUL (total time: 0 seconds)
```

MANİSA
CELAL BAYAR
ÜNİVERSİTESİ

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMANT

2022-2023 SPRING

CSE1120 DISCRETE STRUCTURES

COMPUTER PROJECT

Student Id	Name Surname	1st /2nd Ed.
220315027	UTKU YÜKSEL	1
220315086	BATIN TAHA ÖNAL	1