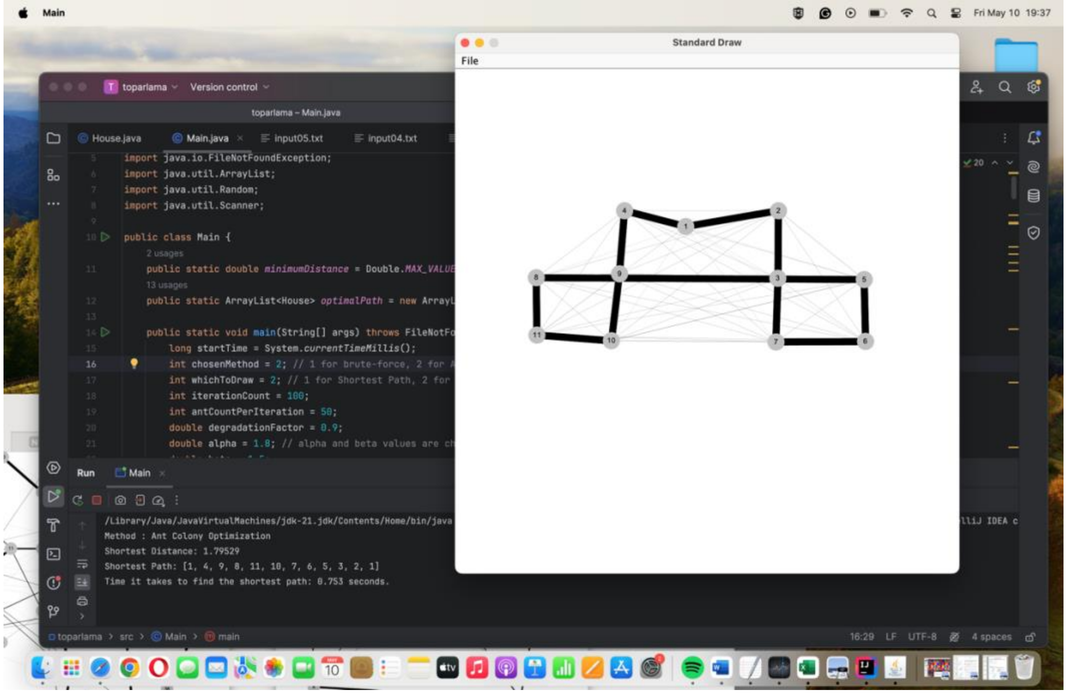
KEREM OĞUZ   ID: 2022400270

USED PARAMETERS FOR ANT COLONY OPTİMİZATİON:

- ⇒ iterationCount = 100;
- ⇒ antCountPerIteration = 50;
- ⇒ degradationFactor = 0.9;
- ⇒ alpha = 1.8;
- ⇒ beta = 1.5;
- ⇒ pheromoneIntensity = 0.1;
- ⇒ qValue = 0.0001;

Pheromone Itensity Graphs:

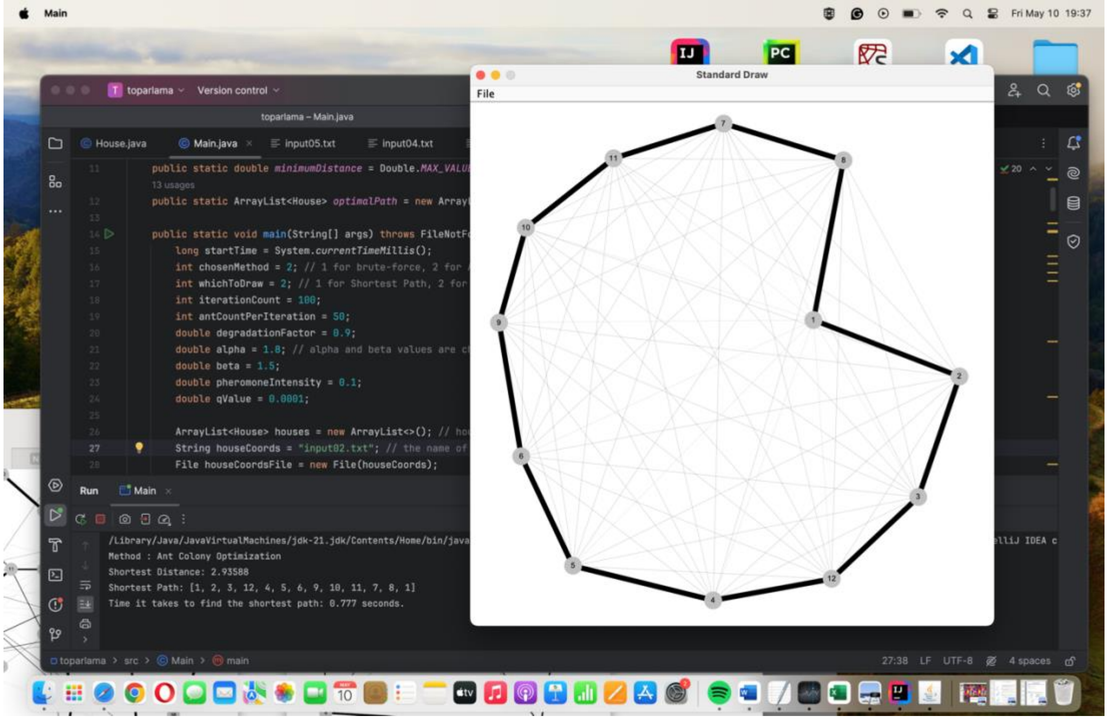
INPUT1:



```java
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

public class Main {
    2 usages
    public static double minimumDistance = Double.MAX_VALUE
    13 usages
    public static ArrayList<House> optimalPath = new ArrayL

    public static void main(String[] args) throws FileNotFo
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for A
        int whichToDraw = 2; // 1 for Shortest Path, 2 for
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are ch
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java
Method : Ant Colony Optimization
Shortest Distance: 1.79529
Shortest Path: [1, 4, 9, 8, 11, 10, 7, 6, 5, 3, 2, 1]
Time it takes to find the shortest path: 0.753 seconds.
```

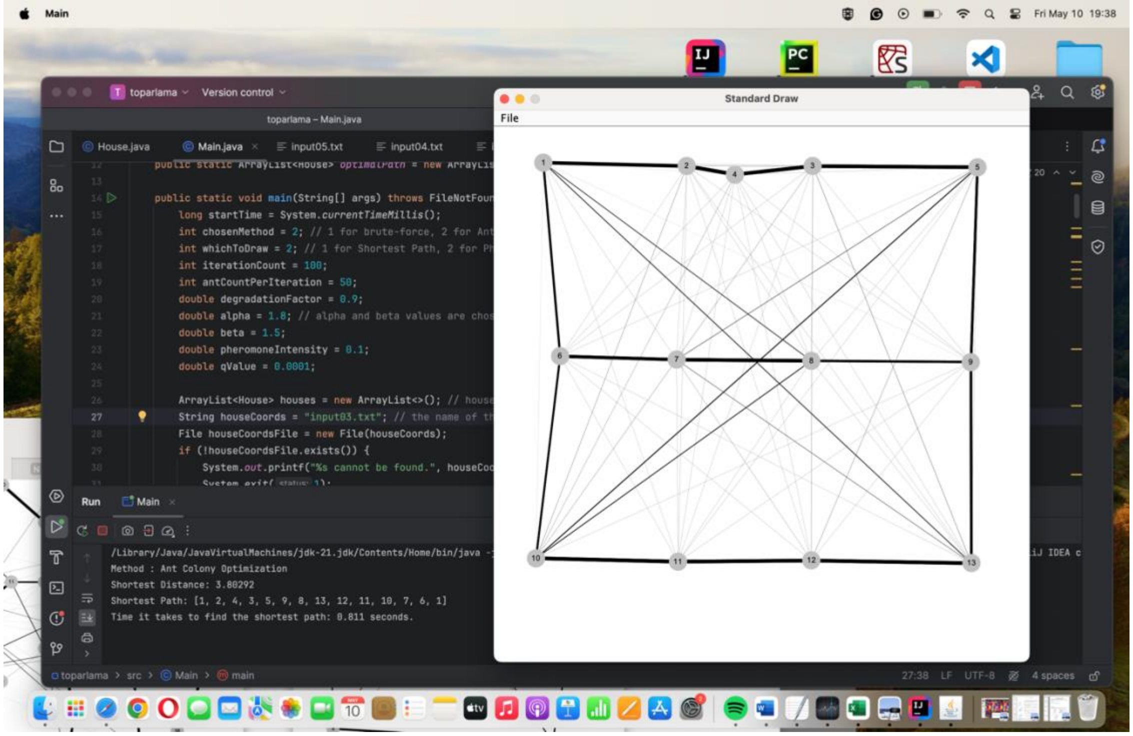
INPUT 2:



```java
    public static double minimumDistance = Double.MAX_VALU
    13 usages
    public static ArrayList<House> optimalPath = new Array

    public static void main(String[] args) throws FileNotF
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for
        int whichToDraw = 2; // 1 for Shortest Path, 2 for
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are c
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // ho
        String houseCoords = "input02.txt"; // the name of
        File houseCoordsFile = new File(houseCoords);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java
Method : Ant Colony Optimization
Shortest Distance: 2.93588
Shortest Path: [1, 2, 3, 12, 4, 5, 6, 9, 10, 11, 7, 8, 1]
Time it takes to find the shortest path: 0.777 seconds.
```

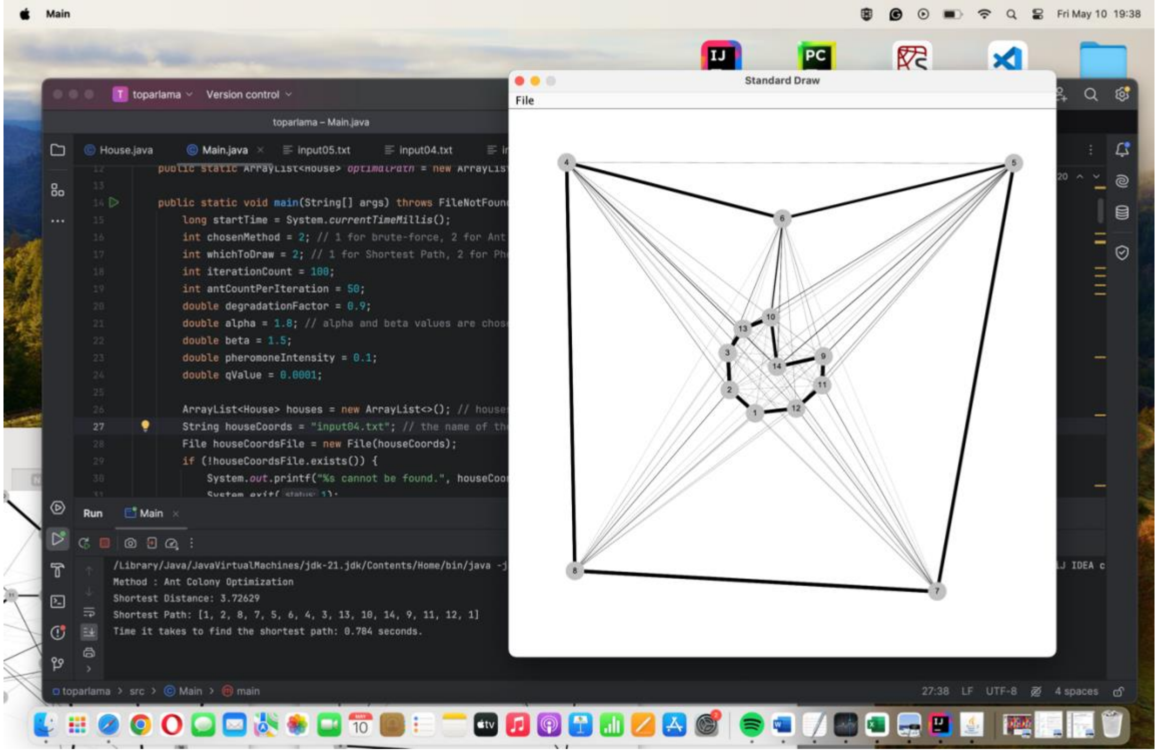
INPUT 3:



```java
    public static ArrayList<House> optimalPath = new ArrayLis

    public static void main(String[] args) throws FileNotFoun
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for Ant
        int whichToDraw = 2; // 1 for Shortest Path, 2 for Ph
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chos
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // house
        String houseCoords = "input03.txt"; // the name of th
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoo
            System.exit( status: 1);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -
Method : Ant Colony Optimization
Shortest Distance: 3.80292
Shortest Path: [1, 2, 4, 3, 5, 9, 8, 13, 12, 11, 10, 7, 6, 1]
Time it takes to find the shortest path: 0.811 seconds.
```

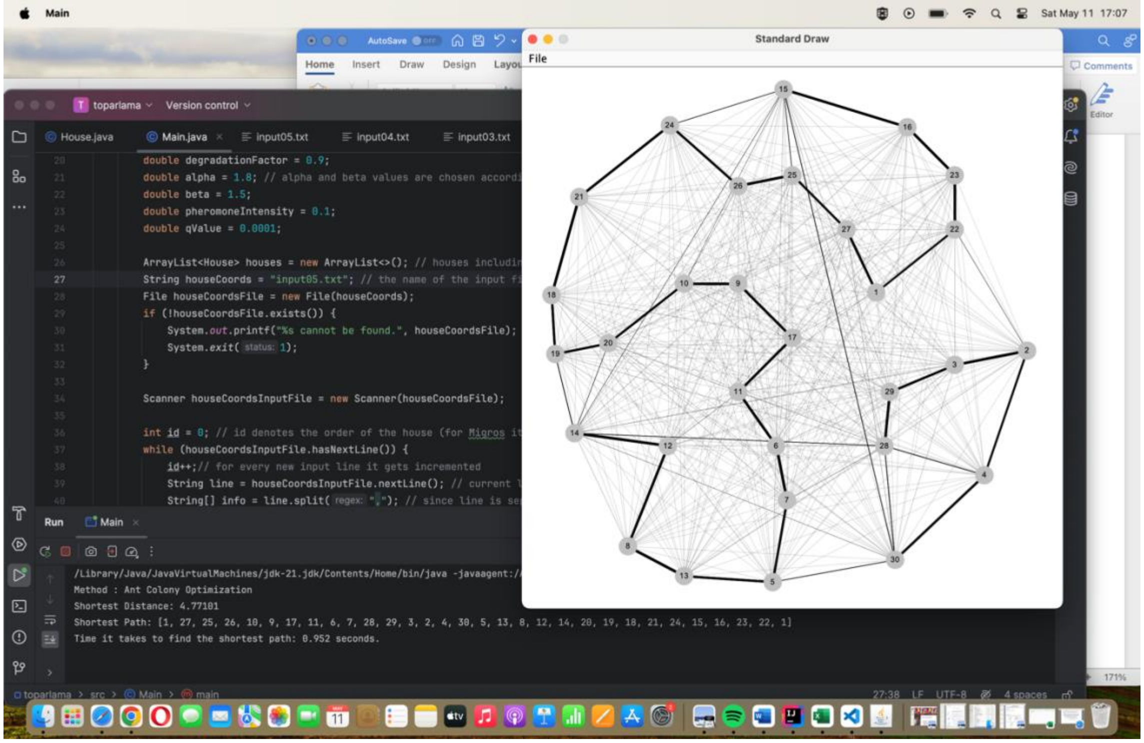
INPUT 4:



```java
    public static ArrayList<House> optimalPath = new ArrayLis

    public static void main(String[] args) throws FileNotFoun
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for Ant
        int whichToDraw = 2; // 1 for Shortest Path, 2 for Ph
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chos
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // house
        String houseCoords = "input04.txt"; // the name of th
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoo
            System.exit( status: 1);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -j
Method : Ant Colony Optimization
Shortest Distance: 3.72629
Shortest Path: [1, 2, 8, 7, 5, 6, 4, 3, 13, 10, 14, 9, 11, 12, 1]
Time it takes to find the shortest path: 0.784 seconds.
```

INPUT 5:



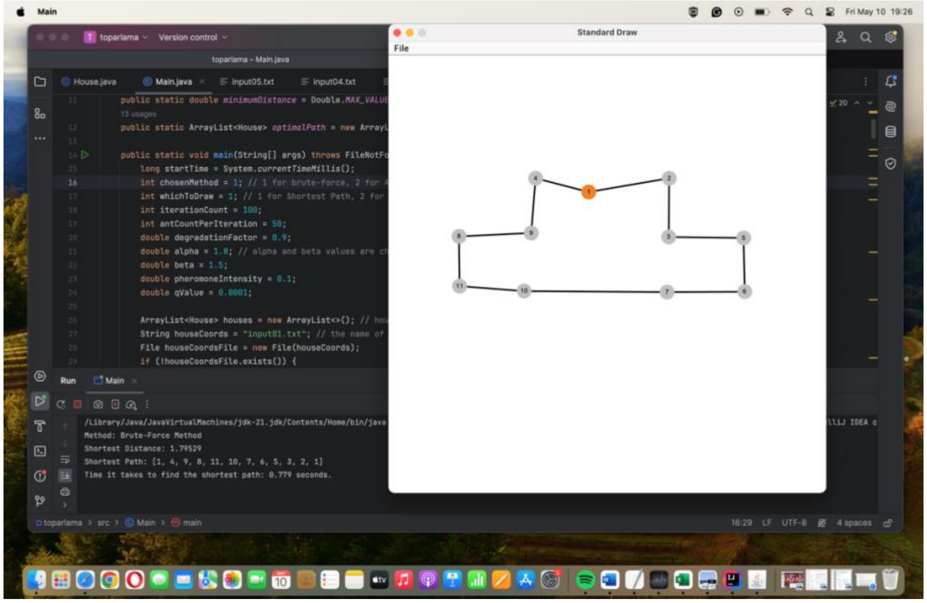
```
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chosen accordi
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // houses includin
        String houseCoords = "input05.txt"; // the name of the input fi
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoordsFile);
            System.exit( status: 1);
        }

        Scanner houseCoordsInputFile = new Scanner(houseCoordsFile);

        int id = 0; // id denotes the order of the house (for Migros it
        while (houseCoordsInputFile.hasNextLine()) {
            id++;// for every new input line it gets incremented
            String line = houseCoordsInputFile.nextLine(); // current l
            String[] info = line.split( regex: ","); // since line is se
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/
Method : Ant Colony Optimization
Shortest Distance: 4.77101
Shortest Path: [1, 27, 25, 26, 10, 9, 17, 11, 6, 7, 28, 29, 3, 2, 4, 30, 5, 13, 8, 12, 14, 20, 19, 18, 21, 24, 15, 16, 23, 22, 1]
Time it takes to find the shortest path: 0.952 seconds.
```

SHORTEST PATH GRAPHS AND CONSOLE OUTPUTS:

BRUTE FORCE:

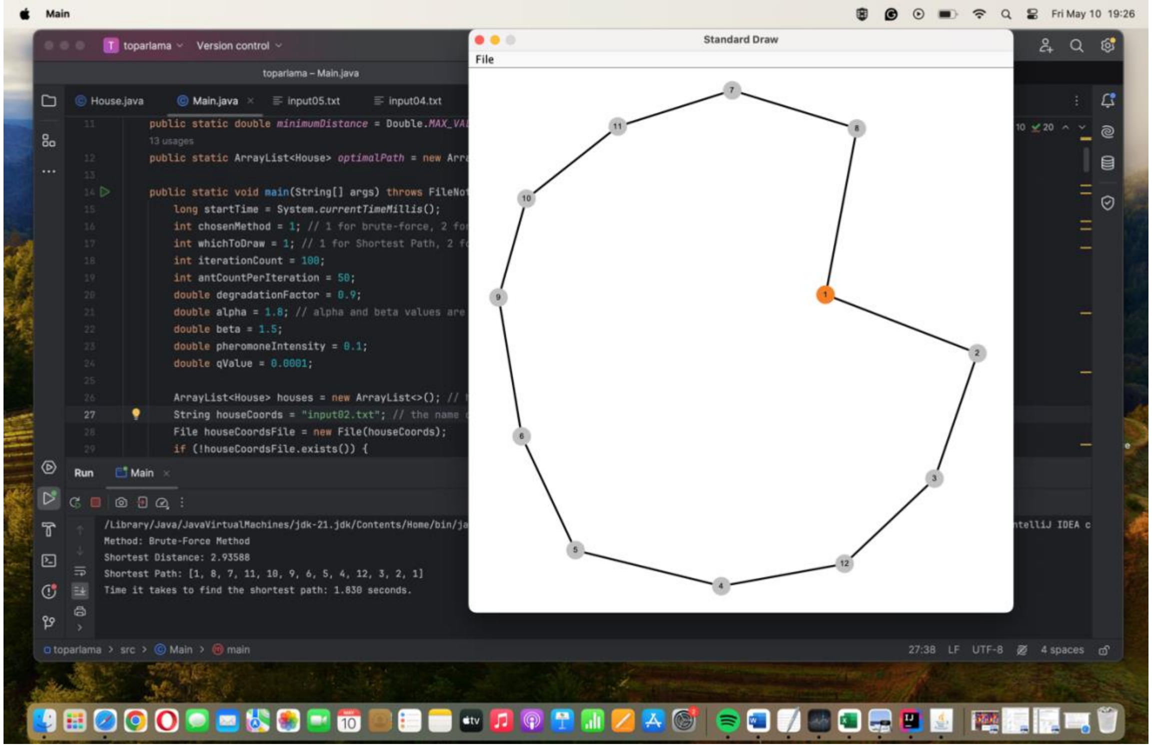
INPUT 1:



```
    public static double minimumDistance = Double.MAX_VALUE
    13 usages
    public static ArrayList<House> optimalPath = new ArrayL

    public static void main(String[] args) throws FileNotFo
        long startTime = System.currentTimeMillis();
        int chosenMethod = 1; // 1 for brute-force, 2 for A
        int whichToDraw = 1; // 1 for Shortest Path, 2 for
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are ch
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // hou
        String houseCoords = "input01.txt"; // the name of
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java
Method: Brute-Force Method
Shortest Distance: 1.79529
Shortest Path: [1, 4, 9, 8, 11, 10, 7, 6, 5, 3, 2, 1]
Time it takes to find the shortest path: 0.779 seconds.
```

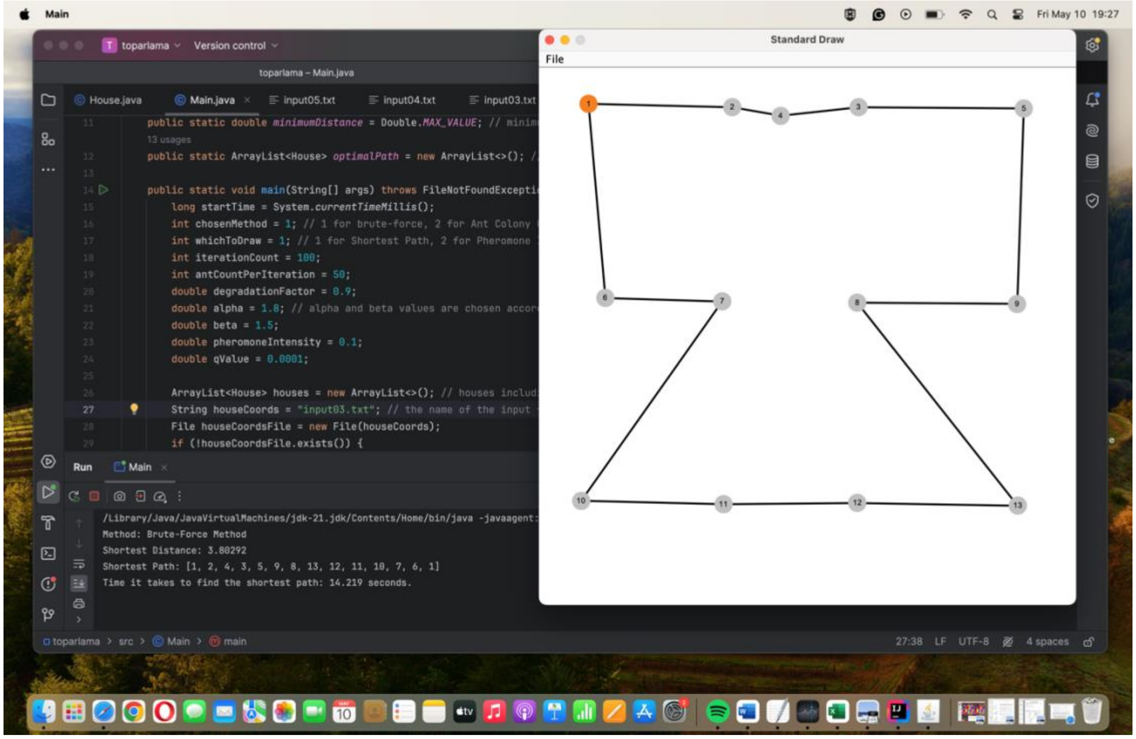
INPUT 2:



```
public static double minimumDistance = Double.MAX_VA
13 usages
public static ArrayList<House> optimalPath = new Arr

public static void main(String[] args) throws FileNot
    long startTime = System.currentTimeMillis();
    int chosenMethod = 1; // 1 for brute-force, 2 fo
    int whichToDraw = 1; // 1 for Shortest Path, 2 f
    int iterationCount = 100;
    int antCountPerIteration = 50;
    double degradationFactor = 0.9;
    double alpha = 1.8; // alpha and beta values are
    double beta = 1.5;
    double pheromoneIntensity = 0.1;
    double qValue = 0.0001;

    ArrayList<House> houses = new ArrayList<>(); // 
    String houseCoords = "input02.txt"; // the name 
    File houseCoordsFile = new File(houseCoords);
    if (!houseCoordsFile.exists()) {
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/ja
Method: Brute-Force Method
Shortest Distance: 2.93588
Shortest Path: [1, 8, 7, 11, 10, 9, 6, 5, 4, 12, 3, 2, 1]
Time it takes to find the shortest path: 1.830 seconds.
```

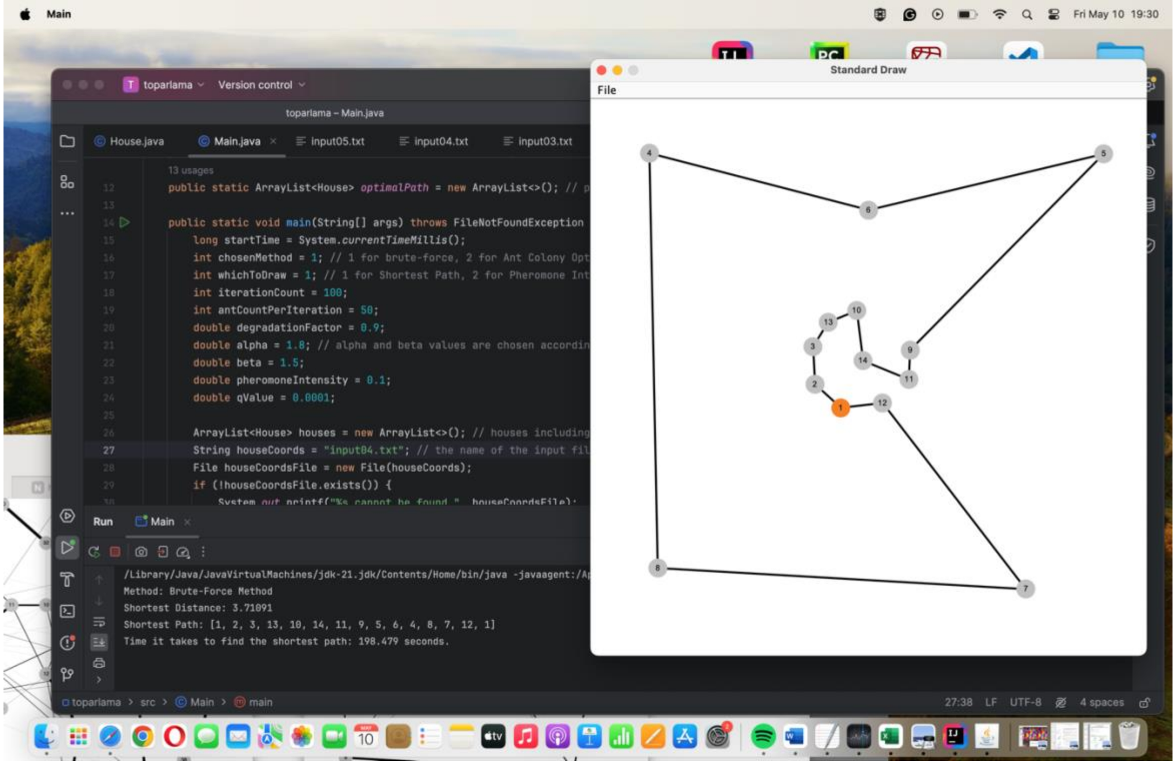
INPUT 3:



```
public static double minimumDistance = Double.MAX_VALUE; // minim
13 usages
public static ArrayList<House> optimalPath = new ArrayList<>(); /

public static void main(String[] args) throws FileNotFoundExcepti
    long startTime = System.currentTimeMillis();
    int chosenMethod = 1; // 1 for brute-force, 2 for Ant Colony
    int whichToDraw = 1; // 1 for Shortest Path, 2 for Pheromone
    int iterationCount = 100;
    int antCountPerIteration = 50;
    double degradationFactor = 0.9;
    double alpha = 1.8; // alpha and beta values are chosen accor
    double beta = 1.5;
    double pheromoneIntensity = 0.1;
    double qValue = 0.0001;

    ArrayList<House> houses = new ArrayList<>(); // houses includ
    String houseCoords = "input03.txt"; // the name of the input
    File houseCoordsFile = new File(houseCoords);
    if (!houseCoordsFile.exists()) {
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:
Method: Brute-Force Method
Shortest Distance: 3.80292
Shortest Path: [1, 2, 4, 3, 5, 9, 8, 13, 12, 11, 10, 7, 6, 1]
Time it takes to find the shortest path: 14.219 seconds.
```

INPUT 4:



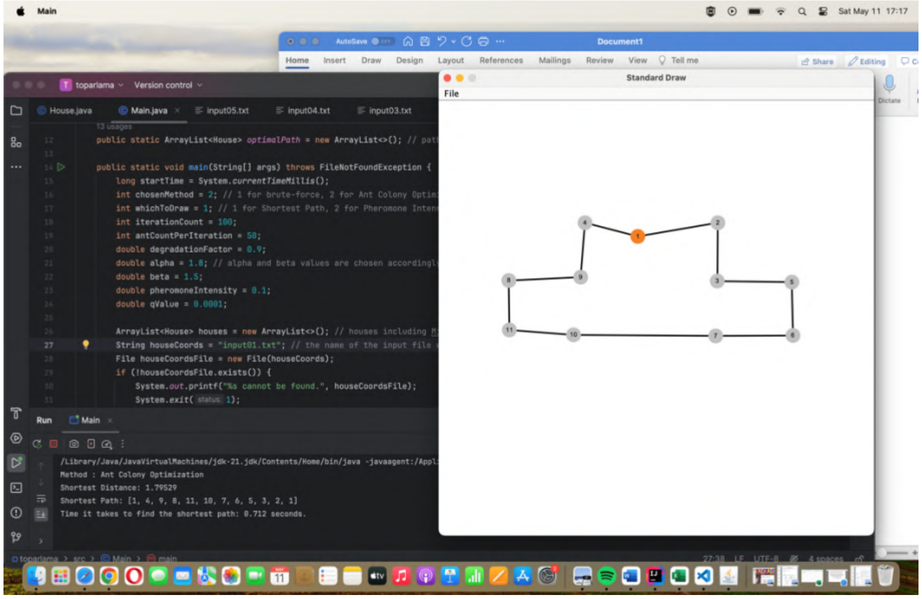
```java
    13 usages
    public static ArrayList<House> optimalPath = new ArrayList<>(); // p

    public static void main(String[] args) throws FileNotFoundException
        long startTime = System.currentTimeMillis();
        int chosenMethod = 1; // 1 for brute-force, 2 for Ant Colony Opt
        int whichToDraw = 1; // 1 for Shortest Path, 2 for Pheromone Int
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chosen accordin
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // houses including
        String houseCoords = "input04.txt"; // the name of the input fil
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoordsFile);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Ap
Method: Brute-Force Method
Shortest Distance: 3.71091
Shortest Path: [1, 2, 3, 13, 10, 14, 11, 9, 5, 6, 4, 8, 7, 12, 1]
Time it takes to find the shortest path: 198.479 seconds.
```

INPUT 5: Too long runtime.

ANT COLONY OPTİMİZATİON SHORTEST PATH GRAPHS:

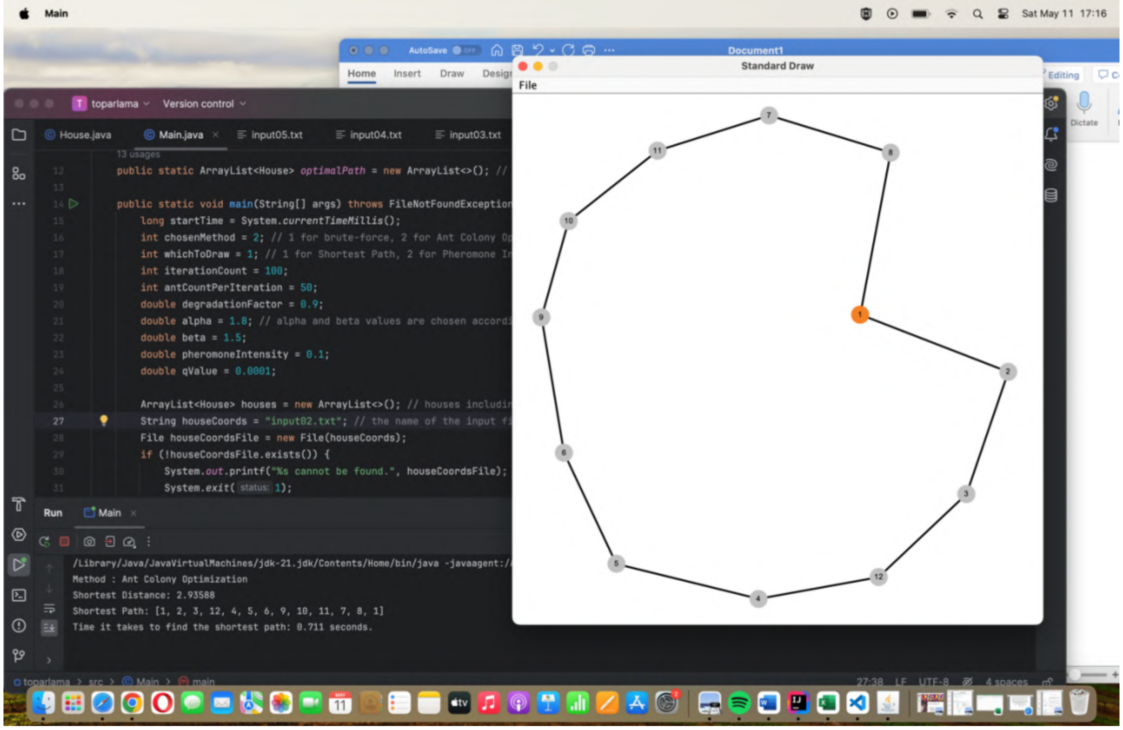
INPUT 1:



```java
    13 usages
    public static ArrayList<House> optimalPath = new ArrayList<>(); // pat

    public static void main(String[] args) throws FileNotFoundException {
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for Ant Colony Optim
        int whichToDraw = 1; // 1 for Shortest Path, 2 for Pheromone Inten
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chosen accordingl
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // houses including M
        String houseCoords = "input01.txt"; // the name of the input file
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoordsFile);
            System.exit( status: 1);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Appl
Method : Ant Colony Optimization
Shortest Distance: 1.79529
Shortest Path: [1, 4, 9, 8, 11, 10, 7, 6, 5, 3, 2, 1]
Time it takes to find the shortest path: 0.712 seconds.
```

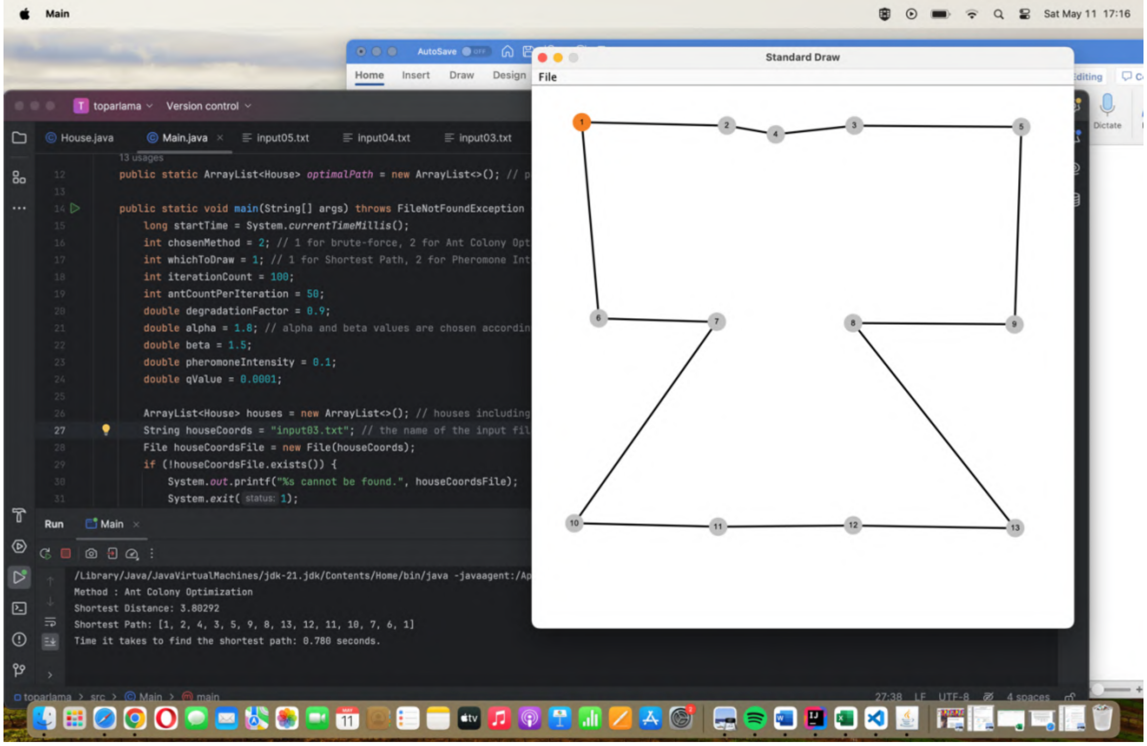
INPUT 2:



```
    public static ArrayList<House> optimalPath = new ArrayList<>(); //

    public static void main(String[] args) throws FileNotFoundException
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for Ant Colony Op
        int whichToDraw = 1; // 1 for Shortest Path, 2 for Pheromone In
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chosen accordi
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // houses includin
        String houseCoords = "input02.txt"; // the name of the input fi
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoordsFile);
            System.exit( status: 1);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/
Method : Ant Colony Optimization
Shortest Distance: 2.93588
Shortest Path: [1, 2, 3, 12, 4, 5, 6, 9, 10, 11, 7, 8, 1]
Time it takes to find the shortest path: 0.711 seconds.
```

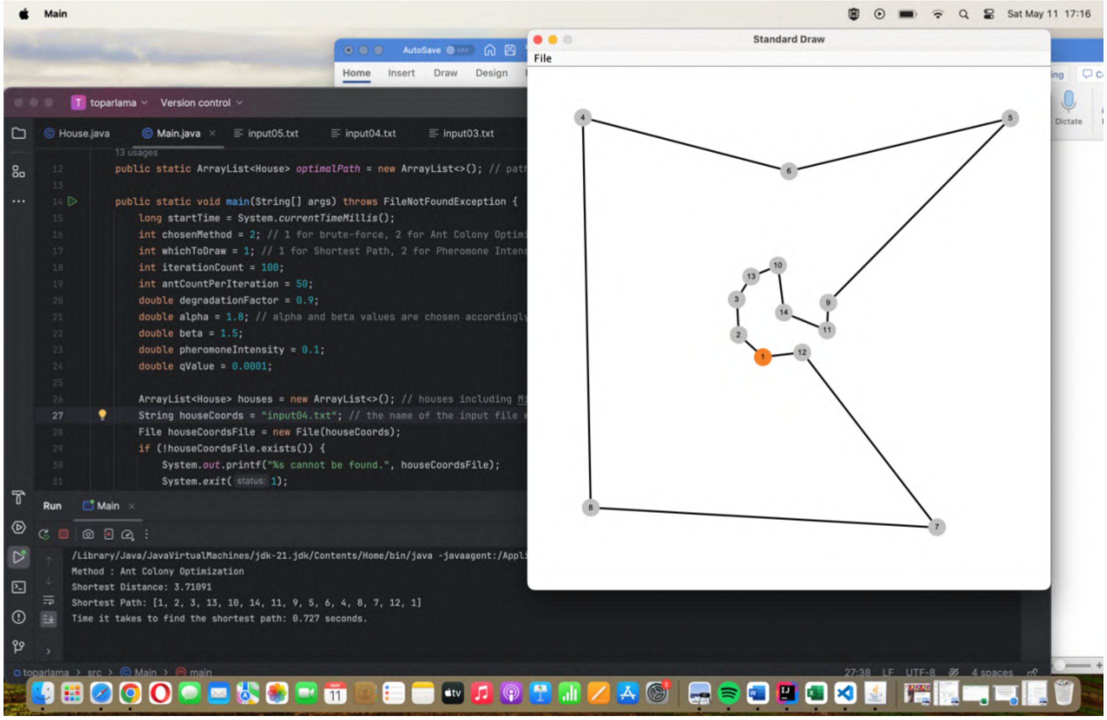
INPUT 3:



```
    public static ArrayList<House> optimalPath = new ArrayList<>(); // p

    public static void main(String[] args) throws FileNotFoundException
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for Ant Colony Opt
        int whichToDraw = 1; // 1 for Shortest Path, 2 for Pheromone Int
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chosen accordin
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // houses including
        String houseCoords = "input03.txt"; // the name of the input fil
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoordsFile);
            System.exit( status: 1);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Ap
Method : Ant Colony Optimization
Shortest Distance: 3.80292
Shortest Path: [1, 2, 4, 3, 5, 9, 8, 13, 12, 11, 10, 7, 6, 1]
Time it takes to find the shortest path: 0.780 seconds.
```

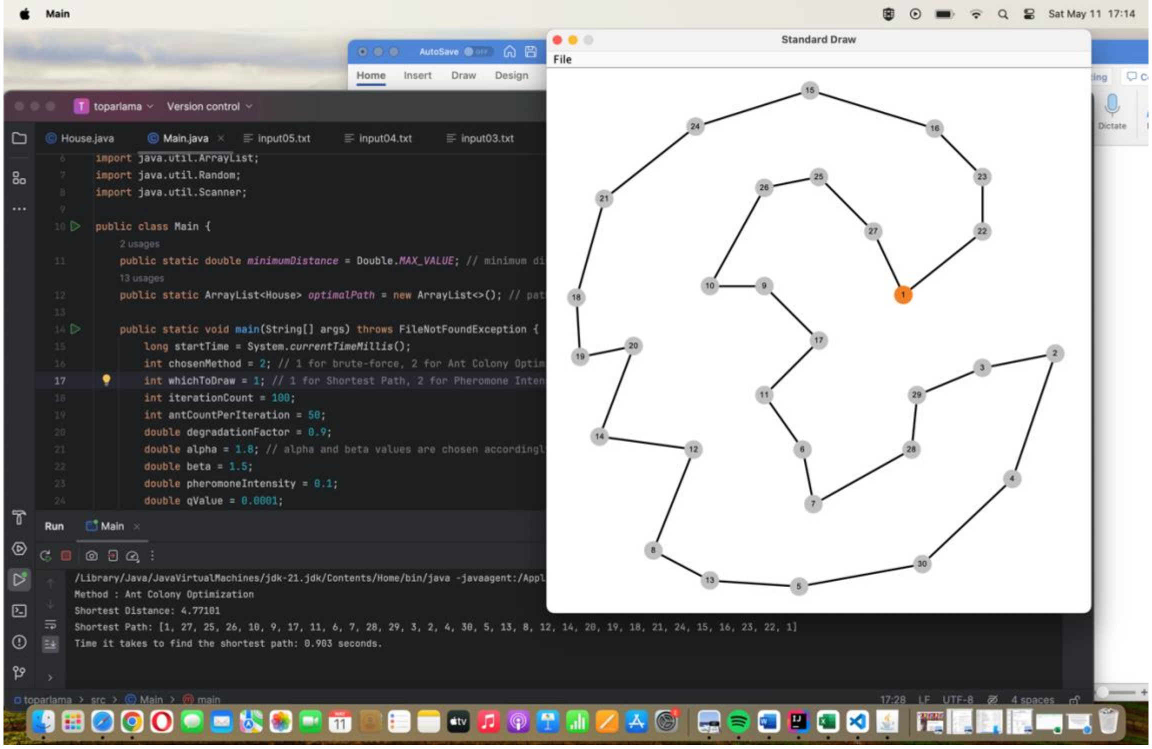
INPUT 4:



```java
    13 usages
    public static ArrayList<House> optimalPath = new ArrayList<>(); // path

    public static void main(String[] args) throws FileNotFoundException {
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for Ant Colony Optimi
        int whichToDraw = 1; // 1 for Shortest Path, 2 for Pheromone Intens
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chosen accordingly
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;

        ArrayList<House> houses = new ArrayList<>(); // houses including Mi
        String houseCoords = "input04.txt"; // the name of the input file w
        File houseCoordsFile = new File(houseCoords);
        if (!houseCoordsFile.exists()) {
            System.out.printf("%s cannot be found.", houseCoordsFile);
            System.exit( status: 1);
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Appl
Method : Ant Colony Optimization
Shortest Distance: 3.71091
Shortest Path: [1, 2, 3, 13, 10, 14, 11, 9, 5, 6, 4, 8, 7, 12, 1]
Time it takes to find the shortest path: 0.727 seconds.
```

INPUT 5:



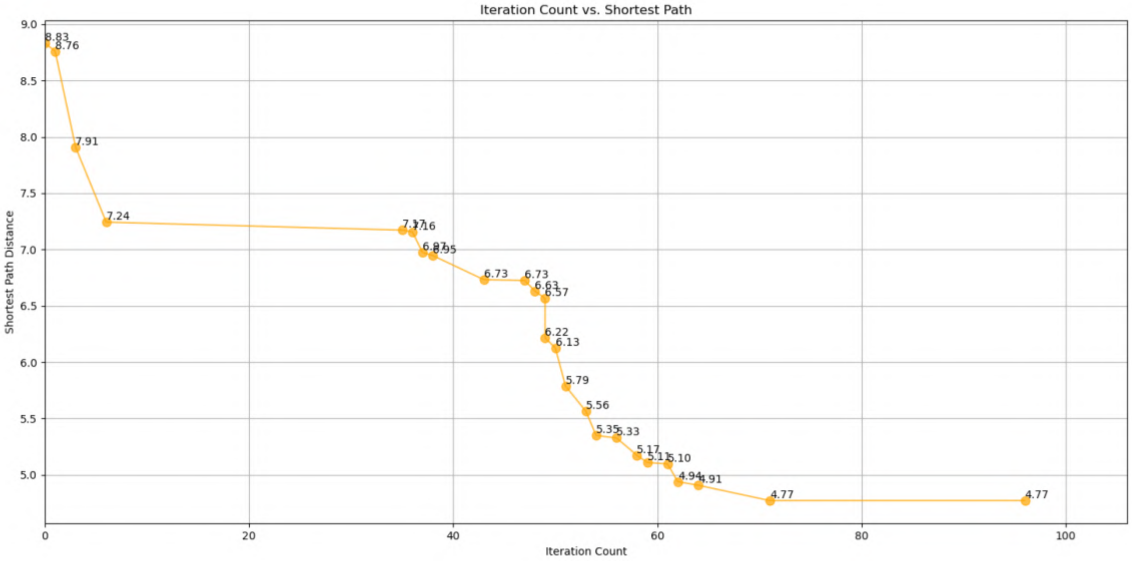
```java
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

public class Main {
    2 usages
    public static double minimumDistance = Double.MAX_VALUE; // minimum di
    13 usages
    public static ArrayList<House> optimalPath = new ArrayList<>(); // pat

    public static void main(String[] args) throws FileNotFoundException {
        long startTime = System.currentTimeMillis();
        int chosenMethod = 2; // 1 for brute-force, 2 for Ant Colony Optim
        int whichToDraw = 1; // 1 for Shortest Path, 2 for Pheromone Inten
        int iterationCount = 100;
        int antCountPerIteration = 50;
        double degradationFactor = 0.9;
        double alpha = 1.8; // alpha and beta values are chosen accordingl
        double beta = 1.5;
        double pheromoneIntensity = 0.1;
        double qValue = 0.0001;
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Appl
Method : Ant Colony Optimization
Shortest Distance: 4.77101
Shortest Path: [1, 27, 25, 26, 10, 9, 17, 11, 6, 7, 28, 29, 3, 2, 4, 30, 5, 13, 8, 12, 14, 20, 19, 18, 21, 24, 15, 16, 23, 22, 1]
Time it takes to find the shortest path: 0.903 seconds.
```

Input File	Number of (Houses+Migros)	Brute-Force Time(seconds)	Ant Colony Time(seconds)	Speed Up Factor
Input1	11	0.787 (Distance: 1.79529	0.764 (Distance: 1.79529)	1.03 times faster
Input2	12	1.846 (Distance: 2.93588)	0.723 (Distance: 2.93588)	2.55 times faster
Input3	13	14.197 (Distance: 3.80292)	0.749 (Distance: 3.80292)	18.95 times faster
Input4	14	198.888 (Distance: 3.71091)	0.747 (Distance: 3.71091)	266.24 times faster
Input5	15	Too long to compute	0.839 (Distance: 4.77101)	A lot faster

- Especially in inputs 4 and 5, the minimum distance sometimes may oscillate a little.

FOR INPUT FILE 5:



REFERENCES:

- Matplotlib (utilized its plotting libraries)
- Instructor's(Berk Gökberk's) explanation of the algorithm at the lecture .(Mostly)
- Youtube → Simulife Hub, HK Lam (reinforced what I learned at the lecture)
- www.upperinc.com (searched advantages and disadvantages of the algorithm)

Pros and Cons of Ant Colony Optimization:

Ant Colony Optimization (ACO) is a metaheuristic inspired by the foraging behavior of ants.

ADVANTAGES:

1. ACO can efficiently explore large solution spaces by focusing on promising areas through pheromone trails, which helps in finding good solutions quickly.

2. ACO excels at finding near-optimal solutions, even in complex problem spaces. By leveraging the collective intelligence of artificial ants and the reinforcement of pheromone trails, ACO can efficiently explore and exploit promising regions of the solution space.

3. ACO is adaptable to dynamic environments where the problem or its constraints change over time. The algorithm can dynamically adjust the pheromone trail intensities, allowing it to quickly respond to changes and find updated optimal or near-optimal solutions.

DİSADVANTAGES:

1. ACO may struggle to converge to the global optimum in complex problem spaces with multiple local optima. The algorithm's reliance on pheromone trails and local information exchange can lead to premature convergence to suboptimal solutions.

2. ACO's performance is sensitive to the appropriate tuning of its parameters. The choice of parameters, such as the pheromone evaporation rate and exploration-exploitation trade-off, can significantly impact the algorithm's convergence speed and solution quality.

3. ACO can be computationally expensive, especially when dealing with large problem instances. As the number of ants and iterations increases, the algorithm's execution time and memory requirements grow, making it challenging to handle real-time or time-critical applications.