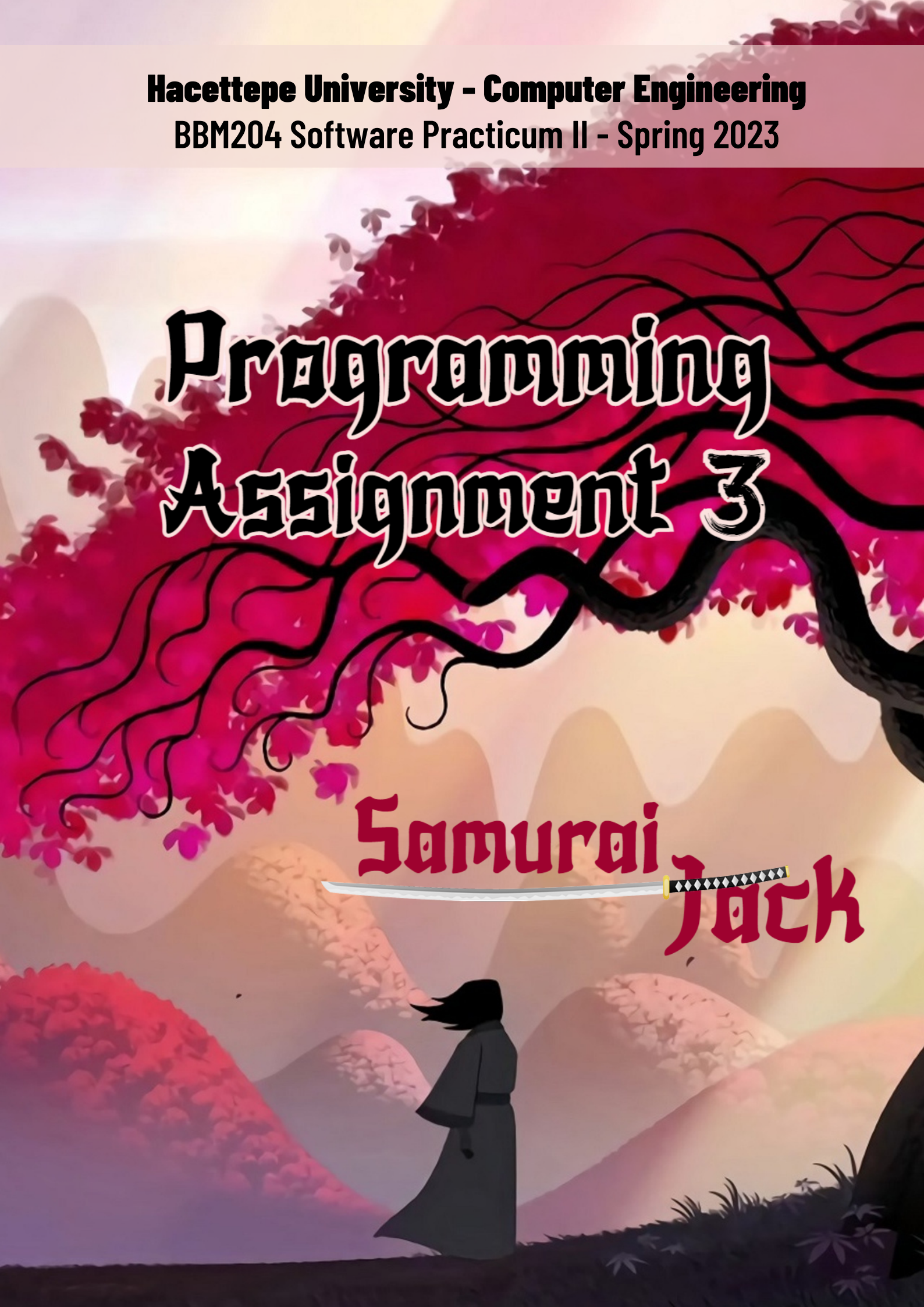
Hacettepe University - Computer Engineering

BBM204 Software Practicum II - Spring 2023

Programming Assignment 3



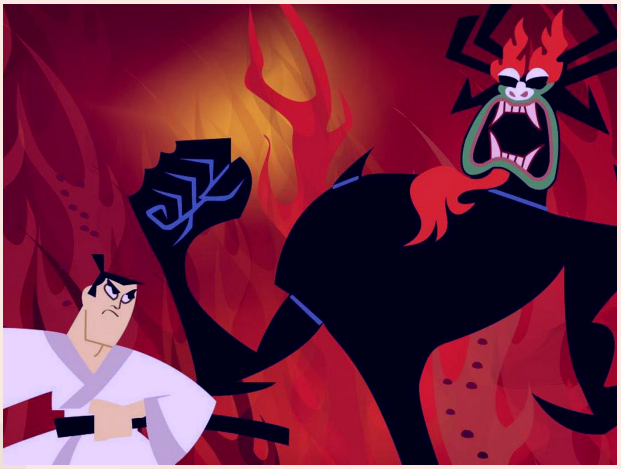
Topics: Graphs - Connected Components, Digraph Traversal and Cycle Detection, Minimum Spanning Trees

Course Instructors: Assoc. Prof. Dr. Erkut Erdem, Prof. Dr. Suat Özdemir, Asst. Prof. Dr. Adnan Özsoy
TAs: Ali Burak Erdoğan, Alperen Çakın, Dr. Selma Dilek
Programming Language: Java (OpenJDK 11)
Due Date: **Thursday, 11.05.2023 (23:59:59)**

A Noble Journey

Once upon a time, the young prince of Japan bore witness to the emergence of the shapeshifting demon **Aku**, who destroyed his kingdom. With the help of his mother, the prince managed to escape and had to dedicate his life to training with skilled warriors from across the world. However, the travel routes were not always safe, even without the Aku's evil grasp over the world. For instance, using ships was dangerous as there was always the danger of a storm and pirates, or some hidden shortcuts could be hazardous as they could easily host bandits. To receive the training that would help the prince face the evil entity, various map-makers of the world have gathered together to construct the safest travel path for the prince. They knew what was at stake, as the silliest of their mistakes could lead to the apocalypse itself. However, they managed to craft this map with the help of some mysterious outsiders who are good with algorithmic magic.

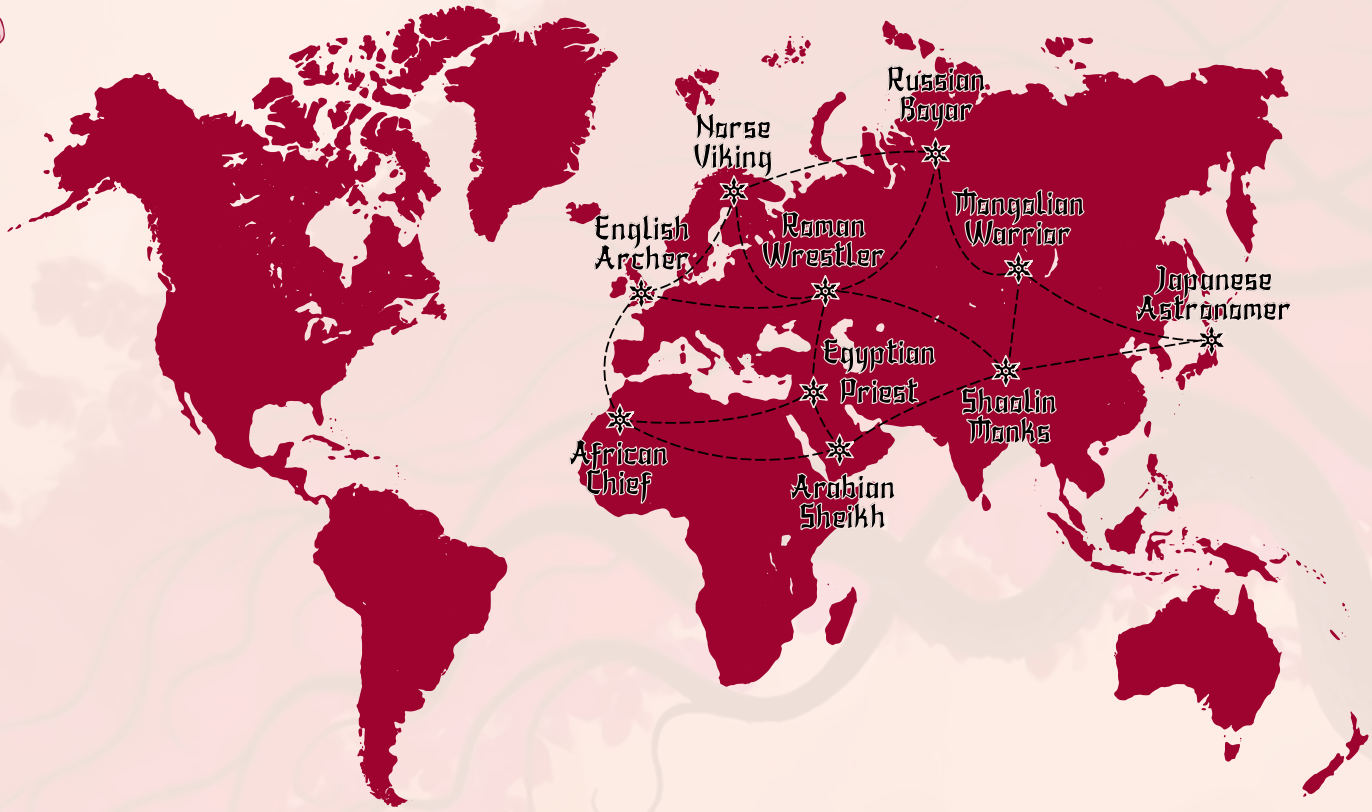
After years of rigorous training, the prince returned to his homeland as a mighty samurai warrior: **Samurai Jack**, armed with a magical sword passed down from his father, with the sole mission of defeating Aku. However, as fate would have it, Aku managed to fling the samurai into the distant future, where he discovered that Aku had not only conquered his entire world but had also extended his rule beyond.



Now, in this dystopian future ruled by Aku, the warrior must fight his way through each of Aku's territories to find a time portal that will take him back in time to the moment before Aku's tyranny began. But Aku has scattered his minions across the land to prevent the warrior's progress. Therefore, some regions became occupied and possessed by Aku's evil.

To locate the time portal, the warrior must first identify the reachable regions. Then, for each territory, the warrior must find a safe path. Aku, being the evil entity himself, has set some time traps even in the relatively safe regions. If the warrior tries to follow such routes in a region with a trap, he will end up in the same place he started, eventually leading to exhaustion, hunger, and death. Therefore, such traps should be identified before he can safely search for a portal.

1 Chapter I - Path of the Warrior



You have been selected by an elite group of computer science magicians, who call themselves **The High TurBo Order**, to help the ancient map-makers craft a safe travel map. This map will be crucial for the young prince to safely reach all training destinations around the world.

1.1 Background Information and Mission Objectives

You will be given data about each training region's name and ID, and the data about the existing trails among them. You may assume that all regions are reachable from any other region, either through a direct trail or via a sequence of trails that go through other regions. Your mission is to implement an algorithm that will create a map of trails among the given training regions such that the total amount of danger for the young prince will be minimum. **The safe map of trails will be fully established when each region can be reached from any other region either through a direct trail, or through a sequence of trails that connect them indirectly via other regions.**

Mission Objective:
Construct the safest map that connects all regions with skillful masters, so that the prince is able to reach all of them with minimum risk.

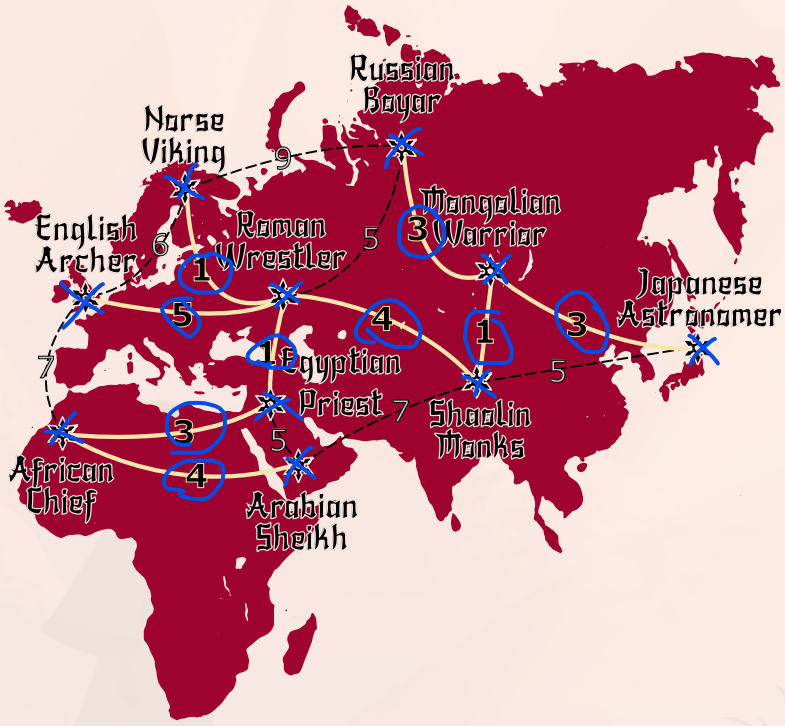
1.2 Input File Format

The input file with the data about the regions to travel will be given in the XML format as the *first command line argument*. Your program should parse the contents of the file and use it to initialize *locations* and *trails* instance variables of the *TravelMap* class. A sample input file is given as *path_of_the_warrior_input_1.xml*. An excerpt from the sample input file is given below to illustrate its format. **Note that all trails are bidirectional (a source region is also the destination and vice versa)**.

```
<TravelMap>
  <Locations>
    <Location><Name>Japanese Astronomer</Name><Id>0</Id></Location>
    <Location><Name>Arabian Sheikh</Name><Id>1</Id></Location>
    <Location><Name>African Chief</Name><Id>2</Id></Location>
        ...
  </Locations>
  <Trails>
    <Trail>
      <Source>0</Source>
      <Destination>8</Destination>
      <Danger>3</Danger>
    </Trail>
    <Trail>
      <Source>0</Source>
      <Destination>9</Destination>
      <Danger>5</Danger>
    </Trail>
        ...
  </Trails>
</TravelMap>
```

1.3 Expected Solution and Output Format

For the given sample input, the expected collection of the safest trails is illustrated in the figure on the right (yellow trails). If you carefully inspect the solution, you may observe that the selected trails form a network such that **all regions are reachable from any other region**, either through a direct trail or through a sequence of trails that pass over other regions. Furthermore, the selected trails form **the safest map** for the young prince in terms of the amount of danger that awaits travelers along the trails. Finally, you can observe that there are no cycles in the final map, as a cycle of trails would provide more than one way to reach some destinations. Such an implementation would not result in the optimum, i.e., the safest solution.



You are expected to complete:

- ***initializeMap()***, ***printSafestTrails()*** *and* ***getSafestTrails()*** *methods of the* ***TravelMap*** *class.*

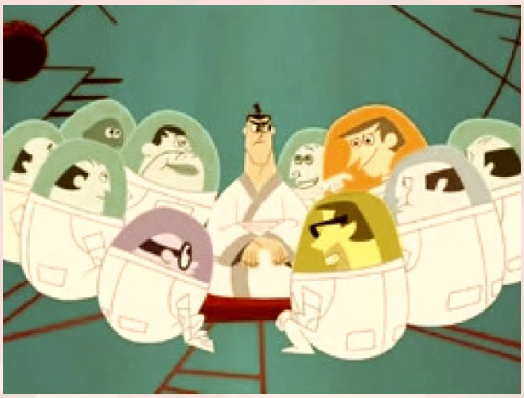
The expected STDOUT output format for the sample input is given below. Each trail's source, destination and danger values should be printed in the same format one after another without extra blank lines. **Note that your output must match the given output format for full credit, however, the order of the lines with trails is not important**.

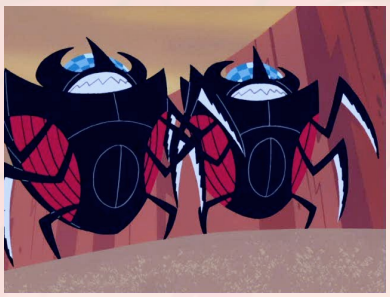
```
The Path of the Warrior
######################
Safest trails are:
The trail from Japanese Astronomer to Mongol Warrior with danger 3
The trail from Mongol Warrior to Shaolin Monks with danger 1
The trail from Russian Boyar to Mongol Warrior with danger 3
The trail from Roman Wrestler to Shaolin Monks with danger 4
The trail from Roman Wrestler to Egyptian Priest with danger 1
The trail from Norse Viking to Roman Wrestler with danger 1
The trail from African Chief to Egyptian Priest with danger 3
The trail from African Chief to Arabian Sheikh with danger 4
The trail from English Archer to Roman Wrestler with danger 5
Total danger: 25
```

2 Chapter II - Finding Hope in the Darkest of Times

After years of rigorous training, the great samurai warrior Jack returned to his land to serve his people and defend them against evil forces, armed with a magical sword passed down from his father, with the sole mission of defeating Aku. His swordsmanship was unmatched, and his reputation as a fearless and just warrior was known throughout the land. But one day, the powerful demon Aku launched an attack on Jack's kingdom. Despite Jack's best efforts, Aku proved too strong. He managed to fling the samurai into the distant future, where he discovered that Aku had not only conquered his entire world but had also extended his rule beyond.

In the distant future, Jack has found himself in a world that was almost unrecognizable. Aku had taken over and enslaved Jack's people, scattering them across the kingdom to work in terrible conditions with no hope of escape. Jack is determined to save his people and bring down Aku's evil empire, but he knows that he cannot do it alone. He wants to send a message to all his people to raise their morale and let them know that he is coming to save them. He cannot go by himself as it would be too dangerous. With the help of the most brilliant **scientists** in the land, Jack devised a plan to use Aku's own technology against him.





They will reprogram some of Aku's evil **beetle drones** to deliver the message because it will be easy for them to move around the cities that are under Aku's control without being suspected. But it won't be easy. Aku has scattered Jack's people far and wide, and some of the cities where they are being held are not connected by roads. **Jack knows that he would need to find a way to identify which cities were connected and which were not so that they would know exactly how many beetle drones they need to send.**

2.1 Background Information and Mission Objectives

A group of cities that are connected to each other via one-way roads is considered to be a **colony**. **You need to implement a program that finds exactly how many colonies there are in the kingdom**. Since colonies are not connected to each other by any roads, the scientists will have to reprogram as many beetle drones as there are colonies, so that a drone carrying the message of hope can be sent to each colony.

Mission Objective: ***Given the data about the cities and the existing one-way roads between them, identify all different colonies in the kingdom.***

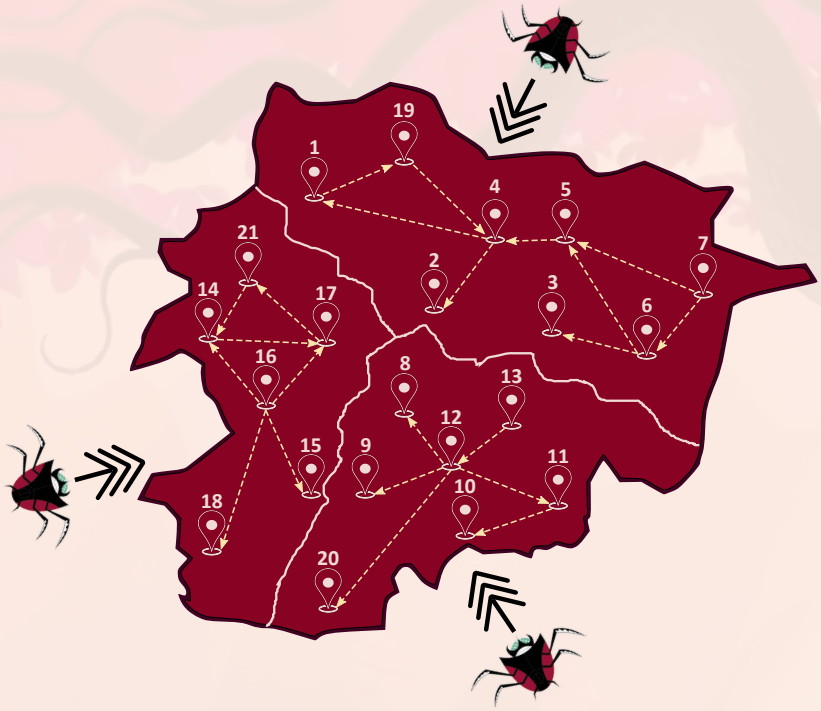
2.2 Input File Format

The input file with the data about the cities and one-way roads between them (represented as an adjacency matrix) will be given in the `TXT` format as the *second command line argument*. Your program should parse the contents of the file to get the information about the cities and roads, which will be used to discover the colonies (instances of the *Colony* class) in the *Kingdom* class. The content of a sample input file given as *kingdom_input_1.txt* with 21 cities is illustrated on the right.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
```

2.3 Expected Solution and Output Format

For the given sample input, the expected number of colonies is illustrated in the figure on the right. If you carefully inspect the solution, you may observe that there are three colonies in total. Therefore, the scientists need to reprogram three beetle drones to deliver Jack's message of hope to each separate colony.



You are expected to complete:

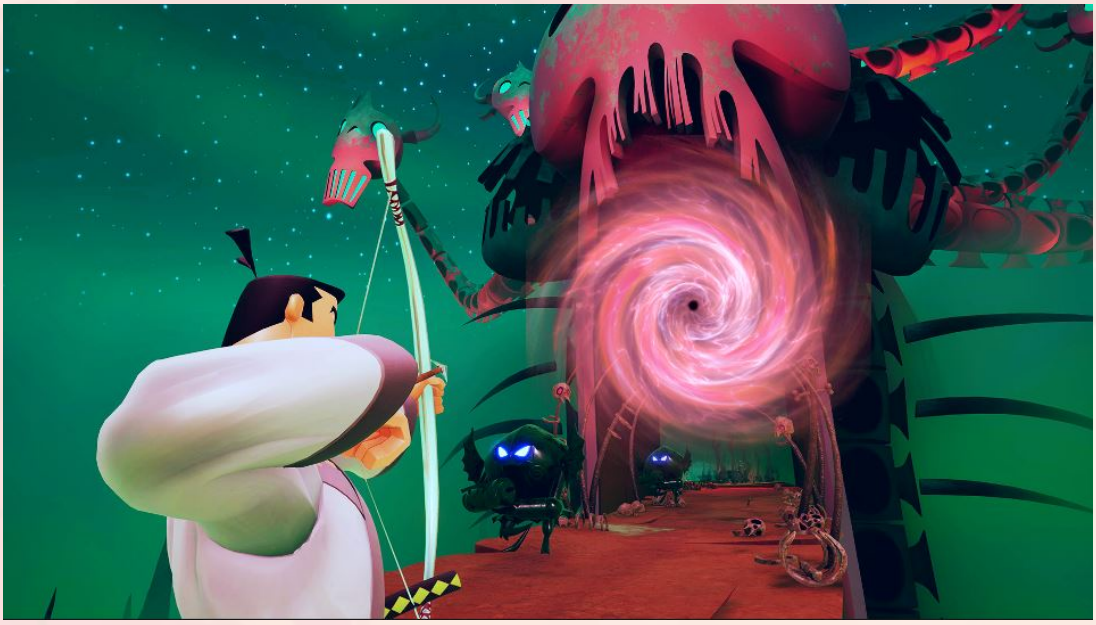
- ***initializeKingdom()***, ***getColonies()*** *and* ***printColonies()*** *methods of the* ***Kingdom*** *class.*

The expected STDOUT format for the sample input is given below. Note that your output must match the given output format for full credit; however, your numbering of the discovered colonies may differ, but the city list must be printed sorted according to city IDs in ascending order.

```
Finding Hope in the Darkest of Times
####################################
Discovered colonies are:
Colony 1: [1, 2, 3, 4, 5, 6, 7, 19]
Colony 2: [8, 9, 10, 11, 12, 13, 20]
Colony 3: [14, 15, 16, 17, 18, 21]
```

3 Chapter III - Escaping the Time Trap Peril

After delivering the message of hope, Samurai Jack must fight his way through Aku's territories to find a time portal that will take him back in time to the moment before Aku's tyranny began, so he can liberate his people. As he travels through Aku's kingdom, he finds himself facing an unexpected challenge: **Aku has created a series of time traps throughout the land, hidden within the cycles of the kingdom's network of roads**. These time traps were designed to ensnare any traveler who dared to venture too far down a cycle of roads in search of the time portal. Once caught in the trap, the traveler will find themselves looping endlessly back to the same point, unable to escape. For Jack, this would be a deadly obstacle, as it would lead to exhaustion, hunger, and eventually death.



To avoid being caught in the time traps that could prevent him from finding the time portal to his own time, Samurai Jack again needs the help of scientists and their expertise in graph theory to analyze the kingdom's road network, detect the cycles containing the traps, and plan a safer route.

3.1 Background Information and Objectives

In some of the colonies, Aku has placed time traps hidden as cycles within the colony's network of roads. A colony can have at most one cycle of roads in its road network, on which a time trap is placed. You must check each colony for a possible time trap (cycle of roads). If a colony does not have a trap, it is labeled as *safe*, whereas if a cycle of roads is detected, the colony should be labeled as *dangerous*.

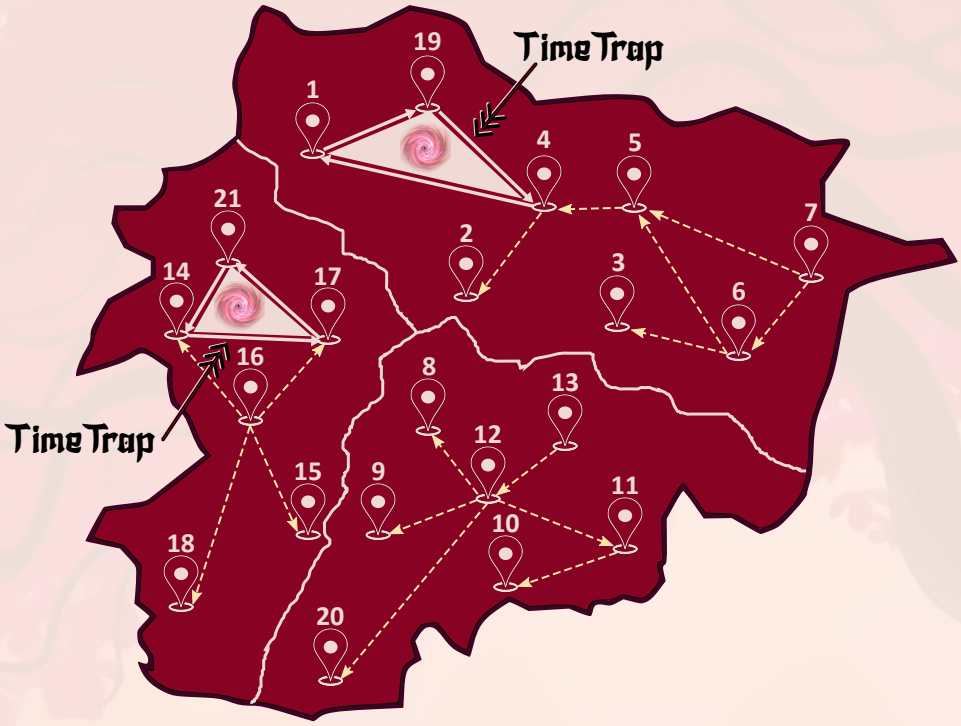
Mission Objective:
Check each colony for a time trap (cycle of roads) to discover if it is SAFE or DANGEROUS, and detect which cities are on the dangerous path.

3.2 Input File Format

Your program should use the List of *Colony* objects obtained in the previous chapter as the input.

3.3 Expected Solution and Output Format

For the given sample input, the expected solution is illustrated in the figure on the right. If you carefully inspect the solution, you may observe that there are two colonies that have a time trap (cycles of roads).



You are expected to complete:

- ***revealTraps()*** *and* ***printTraps()*** *methods of the* ***TrapLocator*** *class.*

The expected STDOUT output format for the given sample input is given below. Note that your output must match the given output format for full credit; however, your numbering of the discovered colonies may differ as in the previous chapter, but the list of the cities on the dangerous path in a dangerous colony must be printed sorted according to city IDs in ascending order.

```
Escaping the Time Trap Peril
############################
Danger exploration conclusions:
Colony 1: Dangerous. Cities on the dangerous path: [1, 4, 19]
Colony 2: Safe
Colony 3: Dangerous. Cities on the dangerous path: [14, 17, 21]
```

Must-Use Starter Codes

You MUST use **this starter code**. All classes should be placed directly inside your **zip** archive. Feel free to create other additional classes if necessary, but they should also be directly inside **zip**.

Grading Policy

- Submission: 1%
- Implementation of the algorithms: 90%
  - Chapter I: 30%
  - Chapter II: 30%
  - Chapter III: 30%
- Output test: 9%

Important Notes

- Do not miss the deadline: **Thursday, 11.05.2023 (23:59:59)**.
- Save all your work until the assignment is graded.
- The assignment solution you submit must be your original, individual work. Duplicate or similar assignments are both going to be considered as cheating.
- You can ask your questions via Piazza (https://piazza.com/hacettepe.edu.tr/spring2023/bbm204), and you are supposed to be aware of everything discussed on Piazza.
- You must test your code via **Tur^{3}Bo Grader** https://test-grader.cs.hacettepe.edu.tr/ (**does not count as submission!**).
- You must submit your work via https://submit.cs.hacettepe.edu.tr/ with the file hierarchy given below:
  - **b<studentID>.zip**
    - Colony.java <FILE>
    - Kingdom.java <FILE>
    - Location.java <FILE>
    - Main.java <FILE>
    - Trail.java <FILE>
    - TrapLocator.java <FILE>
    - TravelMap.java <FILE>
- The name of the main class that contains the main method should be **Main.java**. **You MUST use this starter code**. The main class and all other classes should be placed directly in your **zip** archive. Feel free to create other additional classes if necessary, but they should also be inside the **zip**.
- This file hierarchy must be zipped before submitted (not .rar, only .zip files are supported).
- **Usage of any external libraries is forbidden.**
- Do not submit any .jar or .class files.

Run Configuration

Your code will be compiled and run as follows:

Linux

```
javac -cp *.jar *.java -d .
java -cp .:* Main path_of_the_warrior_input_1.xml kingdom_input_1.txt
```

Windows Powershell

```
javac -cp *.jar *.java -d .
java -cp '.;*' Main path_of_the_warrior_input_1.xml kingdom_input_1.txt
```

Windows CMD

```
javac -cp *.jar *.java -d .
java -cp .;* Main path_of_the_warrior_input_1.xml kingdom_input_1.txt
```

Academic Integrity Policy

All work on assignments **must be done individually**. You are encouraged to discuss the given assignments with your classmates, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in pseudocode) **will not be tolerated**. In short, turning in someone else's work (including work available on the internet), in whole or in part, as your own will be considered as **a violation of academic integrity**. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

The submissions will be subjected to a similarity check. Any submissions that fail the similarity check will not be graded and will be reported to the ethics committee as a case of academic integrity violation, which may result in suspension of the involved students.

References

[1] hdqwalls, "Samurai Jack Artwork", https://hdqwalls.com/wallpaper/1280x2120/samurai-jack-artwork, Last Accessed: 15/04/2023.

[2] Polygon, "Samurai Jacks Aku has a slightly different look this season thats tripping people up", https://www.polygon.com/2017/4/13/15287994/samurai-jack-aku-change, Last Accessed: 15/04/2023.

[3] Samurai Jack Wiki, "The Scientists", https://samuraijack.fandom.com/wiki/The_Scientists, Last Accessed: 15/04/2023.

[4] steemit, "SteemMonsters Origins [Fire Beetle] | Pyro and The Royal Guard", https://steemit.com/steemit/@brandonk/steemmonsters-origins-fire-beetle-or-pyro-and-the-royal-guard, Last Accessed: 15/04/2023.

[5] pluggedin, "Samurai Jack: Battle Through Time", https://www.pluggedin.com/game-reviews/samurai-jack-battle-through-time/, Last Accessed: 15/04/2023.

[6] dafont, "Karasha Font", https://www.dafont.com/karasha.font, Last Accessed: 15/04/2023.