UNIVERSITY OF ŁÓDŹ



ADVANCED ALGORITHMS

assıgnment 2

OGUZHAN OSMA

table of contents

[INTRODUCTION 4](#_Toc43378349)

[CHAPTER 1 - TRIWIZARD TOURNAMENT 5](#_Toc43378350)

[GIVING LABYRINTH MAP 5](#_Toc43378351)

[SETTING EXIT NODE 6](#_Toc43378352)

[CREATING WIZARDS 7](#_Toc43378353)

[AN EXAMPLE FROM THE APPLICATION 8](#_Toc43378354)

[CHAPTER 2 – AUNT’S NAMEDAY 10](#_Toc43378355)

lıst of fıgures

[Figure 1 – Creating Labyrinth 5](#_Toc43378435)

[Figure 2 - Labyrinth 6](#_Toc43378436)

[Figure 3 – End Point 6](#_Toc43378437)

[Figure 4 –Creating Wizards 7](#_Toc43378438)

[Figure 5 – Sleep Method 7](#_Toc43378439)

[Figure 6 – https://youtu.be/YdFgmiKK28A 8](#_Toc43378440)

[Figure 7 – Voldemort’s Visited Nodes 8](#_Toc43378441)

[Figure 8- Harry Potter’s Visited Node 9](#_Toc43378442)

[Figure 9 –Ron Weasley’s Visited Nodes 9](#_Toc43378443)

[Figure 10 –Dislike Network 10](#_Toc43378444)

[Figure 11 – Table Result 10](#_Toc43378445)

[Figure 12 – Table Operation 11](#_Toc43378446)

# ıntroductıon

In this assignment there are two parts which I added. One of them is Triwizard Tournament and the second one is Aunt’s Nameday. Both of them are coded in JAVA programming language. In the first part, the main purpose is usage of BFS Algorithm for the second one, it’s about non-recursive DFS.

# chapter 1 – tRIWIZARD TOURNAMENT

One competition in the TT is to get out of a labyrinth as quickly as possible. Your task as the tournament supervisor is: given the labyrinth map, initial (current) positions of the three competing wizzards and their speeds (in corridors per minute) predict which of them will reach the exit first. Assume that the magical wands used in the play are capable of guiding the wizards to the exit along a shortest possible path.

As a first thing I used threads to determine the winner. For each wizard, one thread is working. If one of wizards has the maximum speed, it means that the thread which is belong to wizard who has max speed will be slept least by program.

## Giving Labyrinth Map

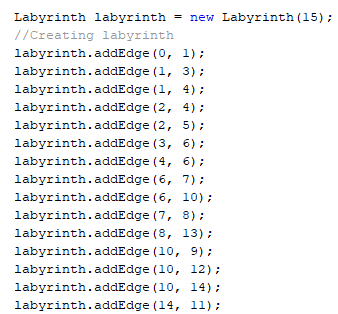
****

Figure 1 – Creating Labyrinth

For this part I drew something on paper and then I created the labyrinth.

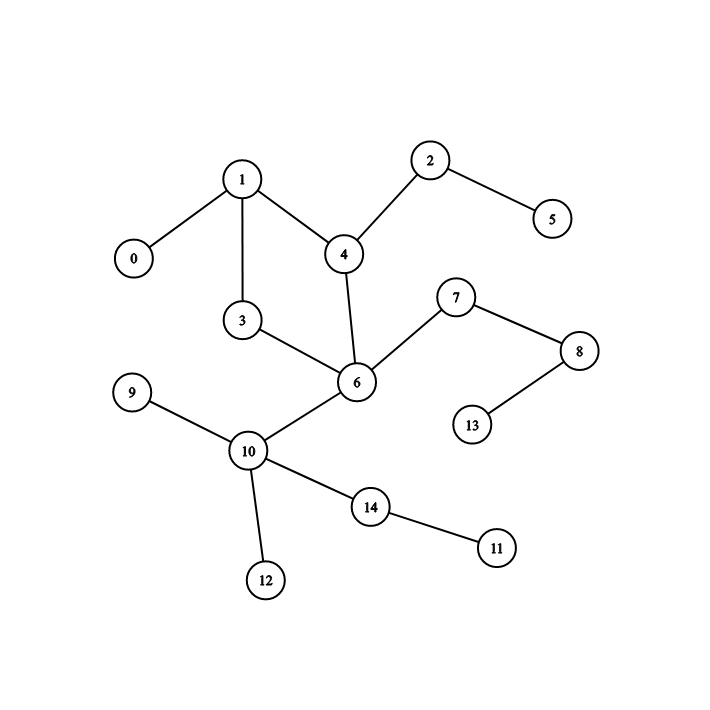


Figure 2 - Labyrinth

## Setting Exit Node

As in the given task, supervisor should specify exit point for labyrinth. Due to this I added endPoint for labyrinth.

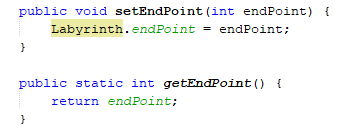


Figure 3 – End Point

## Creating Wizards

In this part I created a class called Wizards. This class implements Runnable class to be able to use threads. A wizard can have a name, starting position, speed, adjacency list, time and place.

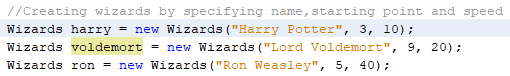


Figure 4 –Creating Wizards

When the supervisor gives the speed, it’s triggering the thread’s sleep method.



Figure 5 – Sleep Method

If we compare the speeds as given above due to sleep method:

Harry => (1000/10)\*10=1000

Voldemort => (1000/20)\*10=500

Ron => (1000/40)\*10=250

Due to these calculations Harry will finish walking in 1 second per corridor, Voldemort 0.5 seconds and Ron 0.25 second per corridor. To be able to determine which wizard won I used AtomicInteger. Thank to AtomicInteger, it helps to determine which thread ended first.

## An example from the application

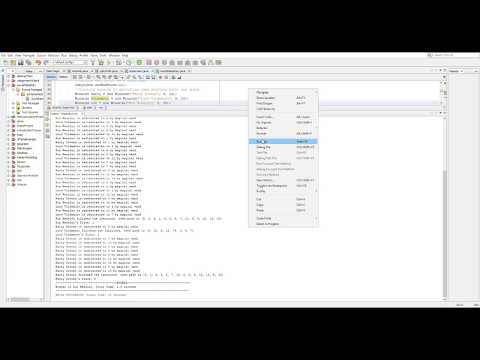
[](https://www.youtube.com/watch?v=YdFgmiKK28A)

Figure 6 – <https://youtu.be/YdFgmiKK28A>

As you can see on the videp Ron has won the tournament. In total the time was 3.5 seconds. Because the speed for Ron was 40 and he visited 14 nodes. It makes 14\*0.25=3.5 seconds.

**Lord Voldemort’s Path**

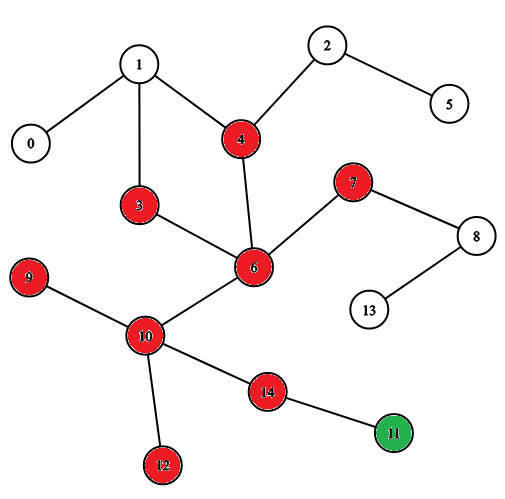
****

Figure 7 – Voldemort’s Visited Nodes

**Harry Potter’s Path**

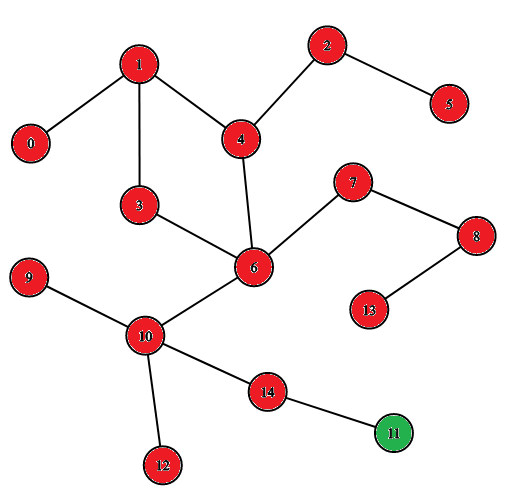
****

Figure 8- Harry Potter’s Visited Node

**Ron Weasley’s Path**

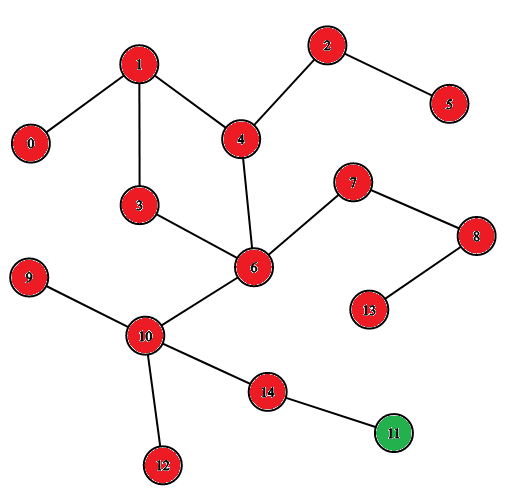
****

Figure 9 –Ron Weasley’s Visited Nodes

# chapter 2 – AUNT’S NAMEDAY

Your beloved aunt Petunia is throwing her namesday party, to which as usual, she invites all the family. There are however some animosities in the family. Aunt's idea is to have two separate tables for quest during the party, so that no two disliking one another pair of people will sit at the same table. Given the list of invited guests and a list of your aunt's suggestions on "who doesn't like whom", your task, as your aunt's dear little pumpkin sausage computer genius, is to set up a "sitting scheme". Using non-recursive DFS is required.

For this part I created a graph which has bidirectional edges.

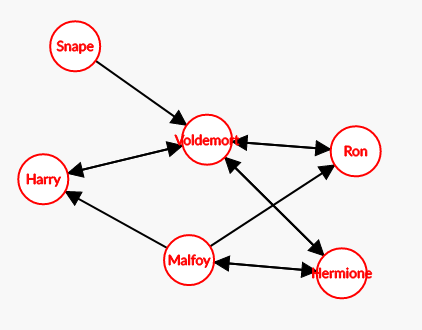


Figure 10 –Dislike Network

The main operation here is when the node(person) is visited, the program controls the tables. If a disliked person is sitting already, program checks other table. It makes this steps for all nodes and then it shows the final look as below:

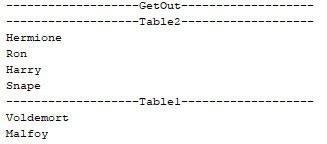


Figure 11 – Table Result

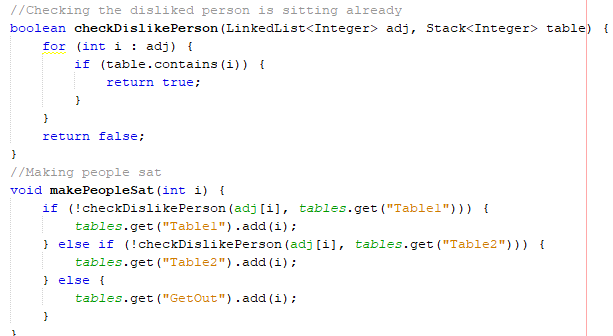


Figure 12 – Table Operation

The block of code above is making people sat and checking dislike network. If one of disliked people is sitting already, it goes for second table so on.