

UNIVERSITY OF LODZ

ADVANCED ALGORITHM—Assignment 2

OGUZ ARAC E-5633

2020

# INDEX

[INTRODUCTION 3](#_Toc43399109)

[TRIWIZARD TOURNAMENT 4](#_Toc43399110)

[1- The TASK 4](#_Toc43399111)

[1.1- Solution 4](#_Toc43399112)

[2- MAZE 4](#_Toc43399113)

[2.1- Connection 4](#_Toc43399114)

[2.2- Finish 5](#_Toc43399115)

[3-WIZARD 6](#_Toc43399116)

[4- OUTPUT 7](#_Toc43399117)

[AUNT'S NAMESDAY 8](#_Toc43399118)

[1- TASK 8](#_Toc43399119)

[1.1- Solution 8](#_Toc43399120)

[2- DISLIKE CONNECTION 8](#_Toc43399121)

[3- OUTPUT 9](#_Toc43399122)

# INTRODUCTION

This report includes two tasks which is triwizard tournament and Aunt’s Namesdays. For these two tasks, I used C# programing language to make. Triwizard Tournament task includes a BFS algorithm. Aunt’s Namesdays includes non-recursive DFS. Triwizard Tournament is solving the maze with BFS. Aunt’s Namesdays is a seating plan for the people who don’t like each other.

# TRIWIZARD TOURNAMENT

## 1- The TASK

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 wizards 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.

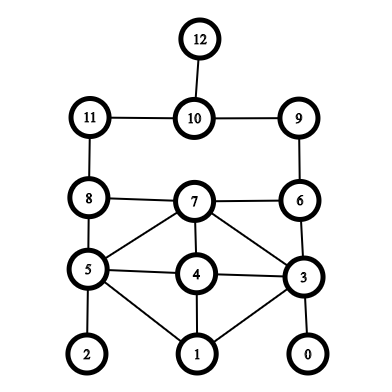
### 1.1- Solution

To helping the wizard to find maze finish point, magical wand benefits from BFS algorithm. To determine the winner, wizard’s class keeps the wizard’s speeds and divided it to path which is followed by wizard. As a result, it can calculate the total time which is passing in the maze corridors and can find the winner.

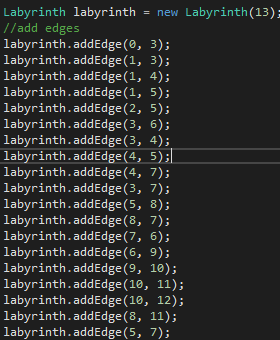
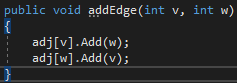
## 2- MAZE

### 2.1- Connection

The image below represents the graph used by the maze.

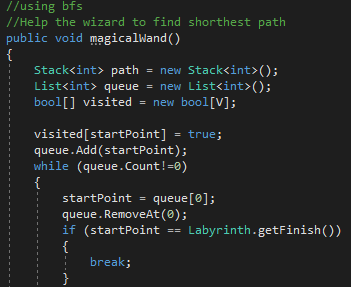


To make this connection, labyrinth class has an addEdge function. This function adds to connection to the list.



### 2.2- Finish

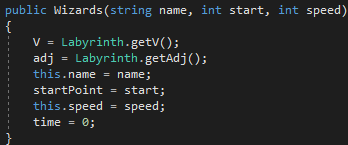
According to the task given, supervisor has to declare a finish node. To make it, it keeps the given node and use it inside the code which is used by wizards class.



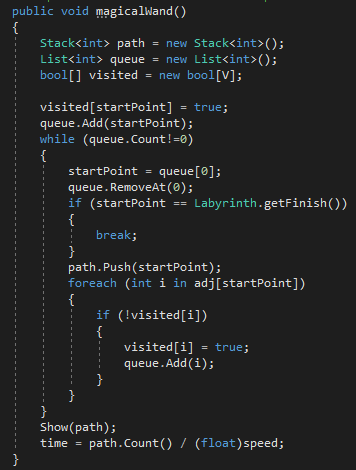
When the start point is equals to the finish point, BFS stops searching to find the finishing node.

## 3-WIZARD

The Wizards class maintains name, start point, and speed to specify the wizard’s attributes.

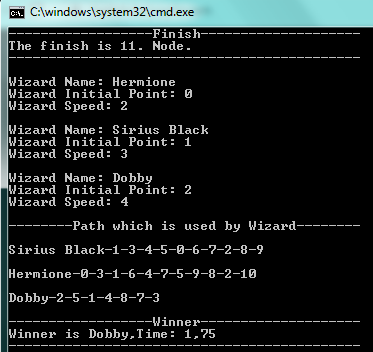


The class also has a magicalWand function which helps the wizard to find the way. The function uses BFS algorithm to search finish point.



## 4- OUTPUT

The Output is as image below.



# AUNT'S NAMESDAY

## 1- TASK

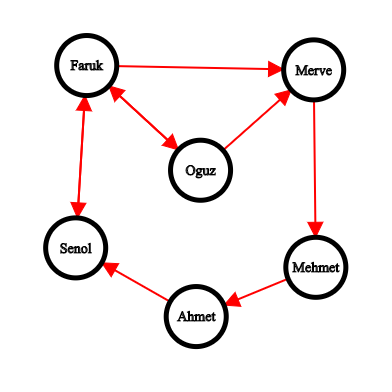
Your beloved aunt Petunia is throwing her names day party, to which as usual, she invites all the family. There are however some animosities in the family. Aunt's idea is to have two separate tables for quest during the party, so that no two disliking one another pair of people will sit at the same table. Given the list of invited guests and a list of your aunt's suggestions on "who does not 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.

### 1.1- Solution

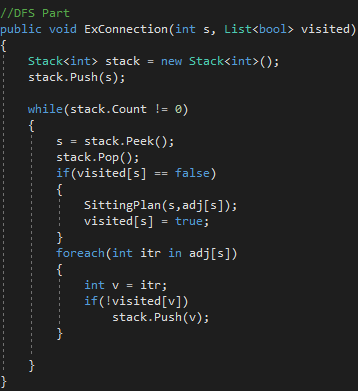
For create a dislike connection and determine who does not like whom, application benefit from DFS algorithm. As a result, it can decide that who will able to sit to the table or not.

## 2- DISLIKE CONNECTION

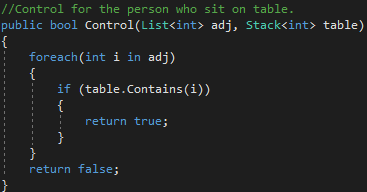
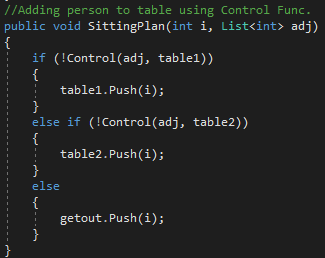
The image below represents that who does not like whom.



DFS algorithm checks the nodes and with using the SittingPlan Function to add person to the table.



The SittinPlan function use Control function which controls the table which person sits.



## 3- OUTPUT

The Result for the dislike connection has to be like image below. The people who dislikes each other cannot sit together.

