# COMP2119C Introduction to data structures and algorithms
# Assignment 3 (Programming)

Due: March 15, 2023

**Note: Submit your works for Part 1 ONLY.**

## Part 1 (Maze)

A hero is trapped in a maze and he wants to escape from it. The maze can be represented as an $m \times n$ matrix of cells. Each cell in the map is referred to by a pair of coordinates that represent the cell's "*location*". Specifically, the location of the upper left corner cell of the map is (0,0), and the location of the lower right corner cell of the map is $(m-1, n-1)$. Each cell in the maze can assume one of two values, namely, 0 or 1. A 0-cell is a cell of empty space where the hero can move to. A 1-cell a cell of solid wall where the hero cannot occupy.

The Hero's movements are restricted by two rules: (1) The hero can move only within the map. (2) The hero can move up, down, left or right. Each valid move takes 1 time unit. Also, the starting and exiting cells are both empty.

Write programs to solve the following problems:

(a) [15%]

Given the locations (coordinates) of the starting and exiting cells, and the maze (as represented by a matrix), compute the minimum amount of time it takes the hero to escape the maze. If the hero cannot reach the exiting cell, your program should return -1.

- Example 1:
  The index of upper-left cell is cell $(0, 0)$ and the index of bottom-right cell is cell $(3, 3)$

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |

  **Input:** M=4 N=4
  graph=[[0,1,1,0],[0,0,1,0],[0,0,0,0],[0,1,1,0]]
  start=[0,0] exit=[3, 3]
  **Output:** 6

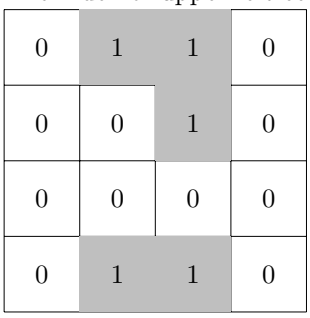
(b) [30%]

The hero has learned a powerful skill - `Flash`. He can choose one direction from {up, down, left, right} and teleport to a target cell in the chosen direction. When using `Flash`, the hero can pass through any wall cells (and of course any empty cells). However, the target cell must be an empty cell. (Otherwise, the hero will be trapped inside the concrete wall.) The hero hasn't mastered this skill perfectly yet, and

so he can only `Flash` once with his (limited) superpower. The maximum distance the hero can travel with `Flash` is $F$ cells. Regardless of the distance traveled using `Flash`, the time taken by `Flash`-ing is 1 unit.

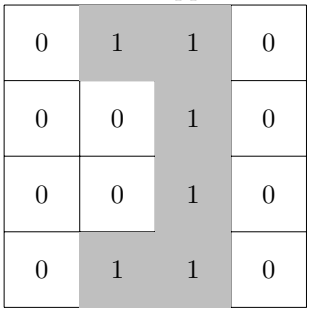
Given a starting cell, an exiting cell, $F$, and a maze, please compute the minimum amount of time for the hero to escape. If the hero cannot reach the exit, return -1.

- Example 1:
  The index of upper-left cell is cell $(0, 0)$ and the index of bottom-right cell is cell $(3, 3)$



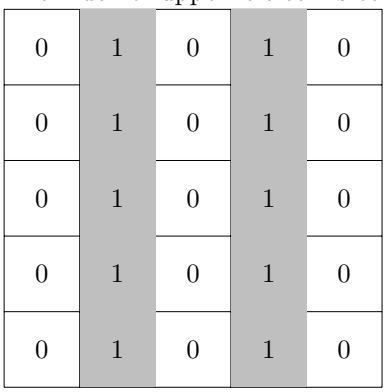

  **Input:** M=4 N=4 F=2
  graph=[[0,1,1,0],[0,0,1,0],[0,0,0,0],[0,1,1,0]]
  start=[0,0] exit=[3, 3]
  **Output:** 5

- Example 2:
  The index of upper-left cell is cell $(0, 0)$ and the index of bottom-right cell is cell $(3, 3)$




  **Input:** M=4 N=4 F=2
  graph=[[0,1,1,0],[0,0,1,0],[0,0,1,0],[0,1,1,0]]
  start=[0,0] exit=[3, 3]
  **Output:** 5
  **Explanation:** The hero can walk to (1,1), Flash to (1,3), and walk to (3, 3).

- Example 3:
  The index of upper-left cell is cell $(0, 0)$ and the index of bottom-right cell is cell $(4, 4)$



**Input:** M=6 N=6 F=4
graph=[[0,1,0,1,0],[0,1,0,1,0],[0,1,0,1,0],[0,1,0,1,0],[0,1,0,1,0]]
start=[0,0] exit=[4, 4]
**Output:** 5
**Explanation:** The hero can Flash to (0,4), and walk to (4, 4).

- Example 4:
The index of upper-left cell is cell $(0, 0)$ and the index of bottom-right cell is cell $(5, 5)$

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |

**Input:** M=6 N=6 F=4
graph=[[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0]]
start=[0,0] exit=[5, 5]
**Output:** -1

(C) [55%]

The hero becomes proficient at `Flash` now. He can use `Flash` multiple times as long as the total distance traveled with `Flash` does not exceed $F$. Regardless of the distance traveled, the time used by each `Flash` is 1 unit.

Given the hero's start, the destination, maximum distance traveled by `Flash`, and the maze, compute the minimum amount of time for him to escape. If the hero cannot reach the destination, return -1.

- Example 1:
The index of upper-left cell is cell $(0, 0)$ and the index of bottom-right cell is cell $(5, 5)$

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |

**Input:** M=6 N=6 F=4
graph=[[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0],[0,1,0,0,1,0]]
start=[0,0] exit=[5, 5]

**Output:** 8
**Explanation:** The hero can Flash to (0,2), walk to (0, 3), Flash to (0, 5) and walk to (5, 5).

**Note:**

1. You should complete the assignment in Python 3. You are allowed to use the 'collections' package, 'heapq' package, 'set', and 'list' directly in your programs.

2. You should complete certain files provided in the **A3** folder (see below). You can only modify the functions **mazeQ1a**, **mazeQ1b** and **mazeQ1c**.

```
A3
├── COMP2119_Graph.pdf ..... The introduction of this assignment is written in this document
└── Part1
    ├── A3_P1_1a.py ..... The mazeQ1a function in this file is for Q1(a) in part 1
    ├── A3_P1_1b.py ..... The mazeQ1b function in this file is for Q1(b) in part 1
    ├── A3_P1_1c.py ..... The mazeQ1c function in this file is for Q1(c) in part 1
    ├── utils.py ..... This file should not be modified
    └── test_data_partial.bin ..... This file should not be modified
```

3. Some test cases are provided but you are encouraged to design your own test cases. The auto-grading result will be shown on your terminal once you run '**python A3_P1_1a.py**', '**python A3_P1_1b.py**'. Your score in the assignment will be evaluated by other (unrevealed) test cases, which are not used by the auto-grader. You should organize your submitted files in the following way. Please replace the **UID** with your university number and make sure your programs can run normally before you zip the folder **UID** as a **UID.zip** file.

```
UID
└── Part1
    ├── A3_P1_1a.py ..... This is your code file for Q1(a)in part 1
    ├── A3_P1_1b.py ..... This is your code file for Q1(b)in part 1
    ├── A3_P1_1c.py ..... This is your code file for Q1(c)in part 1
    └── utils.py ..... This file should not be modified
```

4. In the **mazeQ1a** function, there will be three parameters, *graph*, *start* and *end*. In the **mazeQ1b** and **mazeQ1c** function, there will be four parameters, *graph*, *start* and *end* and $F$ .The structure of the parameter *graph* is List[List[int]] and the parameter *start* and *end* is List[int] and the parameter $F$ is int. The output variable of **mazeQ1a**, **mazeQ1b** and **mazeQ1c** is int.

5. You can test your implementation by `python utils.py`, which will compare your results with the standard ones on certain test cases stored in `test_data_partial.bin`. These test cases may or may not be the same as those used to grade your assignment, so the result is merely for your reference and debugging.

# Part 2

**Q 1.** Islands travel

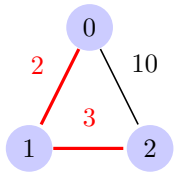
There are $n$ islands connected by $m$ ships. Each ship connects the starting island $u$ and the destination island $v$ for a $w$ cost.

(1) Given $n$ islands and $m$ routes of the ferry, a rabbit wants to travel from the *src* island to the *dst* island. Each route of the ferry is two-way (i.e. the edge is undirected) and there is at most one route of the ferry between every two islands

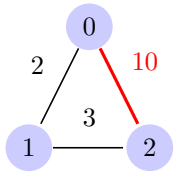
Help the rabbit find the least cost route with up to $k$ stops. If it is impossible for the rabbit to reach *dst* island under the condition, please return -1.

**Think**: what if the route is directed and/or there can be multiple routes between every two islands?

1 Input:
$n = 3$, $ferry\_routes = [[0, 1, 2], [1, 2, 3], [0, 2, 10]]$
$src = 0$, $dst = 2$, $k = 1$
Output:
5
Explanation: The best route with the least cost and up to 1 stop are to take a ship from 0 to 1, then take a ship from 1 to 2. This route will cost $5 = 2 + 3$.

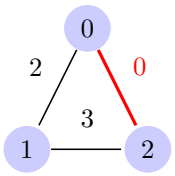



2 Input:
$n = 3$, $ferry\_routes = [[0, 1, 2], [1, 2, 3], [0, 2, 10]]$
$src = 0$, $dst = 2$, $k = 0$
Output:
10
Explanation: The route with the least cost and up to 0 stop is to take a ship from 0 to 2 directly. This route will cost 10.




(2) The rabbit is lucky to win a lottery. It can get a free ticket from a selected list of routes (specified in f_edges). Please help the rabbit the find the route from *src* to *dst* with the minimum cost.

1 Input:
$n = 3$, $ferry\_routes = [[0, 1, 2], [1, 2, 3], [0, 2, 10]]$
$free\_ferry\_routes = [[0, 1], [0, 2]]$
$src = 0$, $dst = 2$
Output:
0
Explanation: The rabbit chooses to use the free ticket on the route $0 \to 2$ and reach *dst* directly. If it chooses to use the free ticket on the route $0 \to 1$, it will cost 3 otherwise.




**Q 2.** Maze with a Stronger Hero

The hero in Part 1 has leveled up and acquires one more skill - `Destruction`. When using `Destruction`, he can eliminate the wall in a cell (i.e. turning that cell from 1 to 0). The hero can use `Destruction` at most $D$ times. Each `Destruction` takes 1 unit of time.

Given the hero's starting cell, the exiting cell, the maximum distance traveled by `Flash`, the maximum number of `Destruction`, and the maze, compute the minimum amount of time for the hero to escape. If the hero cannot reach the destination, return -1. This question is just for fun and you are encouraged to write programs and test them yourself. (To be honest, I am not sure if there's an efficient algorithm for this.)

**Note:**

1. In the folder **Part2**, You can only modify the code within the **islandTravelQ2a** function in **A3_P2_1.py** and **islandTravelQ2b A3_P2_2.py**. In the **islandTravelQ2a** function, there will be five parameters, $n$, *ferry_routes*, *src*, *dst* and $k$. Their types are `int`, `List[Tuple[int, int, int]]`, `int`, `int`, `int` respectively. The type of output of the function is also `int`.

2. In the **islandTravelQ2a** function, there will be five parameters, $n$, *ferry_routes*, *free_ferry_routes*, *src*, *dst* and . Their types are `int`, `List[Tuple[int, int, int]]`, `List[Tuple[int, int]]`, `int`, `int` respectively. The type of output of the function is also `int`.

3. You can test your implementation by `python utils.py`, which will compare your results with the standard ones on certain test cases stored in `test_data_partial.bin`.