Part 2: Graph & Data Structure Selection Problems (~5.5 hrs)

Instructions

For each problem, identify the appropriate data structure for solving the problem. For graph problems, also identify the appropriate graph algorithm. **State the data structure (and algorithm, if applicable) in a comment at the top of your file**. As a reminder, the options are:

1. Graph
    a. Breadth-first search
    b. Depth-first search
    c. Generic traversal (either BFS or DFS works equally well; if one is preferable in terms of Big O either in some or all cases, select that option instead)
    d. Topological sort
2. Stack
3. Queue
4. Deque
5. Heap
6. Priority Queue
7. Tree
8. Hashmap
9. Hashset

Then, write a function to solve the problem and write test cases to check your function. When run, the file you submit should execute your function on your test cases (e.g., through a main method, if applicable in your language). **State the time and space complexity of your solution in a comment at the top of your file**.

Time how long you spend on each problem. You should actively **work on each problem for a MAXIMUM of 40 minutes**. Once 40 minutes has elapsed, submit whatever you have, regardless of whether you are finished. Please indicate in a comment at the bottom of your file how long you spent on the problem. It is important that you are honest about how long each problem took you as it will help your mentor help you!

Question 4: NumberOfIslands

Given a binary matrix in which 1s represent land and 0s represent water. Return the number of islands (contiguous 1s surrounded by 0s or the edge of the matrix).

Examples:

```
Input:
```

1	0	1	1	1
1	1	0	1	1
0	1	0	0	0
0	0	0	1	0
0	0	0	0	0

```
Output: 3 (shown in different colors above)

Input:
```

1	0	0
0	0	0

```
Output: 1 (shown in different colors above)
```

Question 5: FirstKBinaryNumbers

Given a number, k, return an array of the first k binary numbers, represented as strings.

Examples:

```
Input: 5
Output: ["0", "1", "10", "11", "100"]

Input: 10
Output: ["0", "1", "10", "11", "100", "101", "110", "111", "1000", "1001"]
```

Question 6: RoadNetworks

In some states, it is not possible to drive between any two towns because they are not connected to the same road network. Given a list of towns and a list of pairs representing roads between towns, return the number of road networks. (For example, a state in which all towns are connected by roads has 1 road network, and a state in which none of the towns are connected by roads has 0 road networks.)

Examples:

```
Input: ["Skagway", "Juneau", "Gustavus", "Homer", "Port Alsworth", "Glacier Bay",
"Fairbanks", "McCarthy", "Copper Center", "Healy"],
[("Anchorage", "Homer"), ("Glacier Bay", "Gustavus"), ("Copper Center", "McCarthy"),
("Anchorage", "Copper Center"), ("Copper Center", "Fairbanks"), ("Healy", "Fairbanks"),
("Healy", "Anchorage")]

Output: 2 (Networks are Homer-Glacier Bay and Anchorage-Fairbanks-McCarthy-Copper
Center-Homer-Healy)
```

```
Input: ["Kona", "Hilo", "Volcano", "Lahaina", "Hana", "Haiku", "Kahului", "Princeville",
"Lihue", "Waimea"], [("Kona", "Volcano"), ("Volcano", "Hilo") ("Lahaina", "Hana"),
("Kahului", "Haiku"), ("Hana", "Haiku"), ("Kahului", Lahaina"), ("Princeville", "Lihue"),
("Lihue", "Waimea")]

Output: 2 (Networks are Kona-Hilo-Volcano, Haiku-Kahului-Lahaina-Hana, and
Lihue-Waimea-Princeville)
```

Question 7: ReverseWords

Given a string, return the string with the order of the space-separated words reversed

Examples:

```
Input: "Uber Career Prep"
Output: "Prep Career Uber"

Input: "Emma lives in Brooklyn, New York."
Output: "York. New Brooklyn, in lives Emma"
```

Question 8: AlternatingPath

Given an origin and a destination in a directed graph in which edges can be blue or red, determine the length of the shortest path from the origin to the destination in which the edges traversed alternate in color. Return -1 if no such path exists.

Examples:

```
[(A, B, "blue"), (A, C, "red"), (B, D, "blue"), (B, E, "blue"), (C, B, "red"), (D, C,
"blue"), (A, D, "red"), (D, E, "red"), (E, C, "red")]

Input: origin = A, destination = E
Output: 4 (path: A→D (red), D→C (blue), C→B (red), B→E (blue))

Input: origin = E, destination = D
Output: -1 (only path is: E→C (red), C→B (red), B→D (blue))
```

Question 9: MergeKSortedArrays

Given an array of k sorted arrays, merge the k arrays into a single sorted array.

Examples:

```
Input: 2, [[1, 2, 3, 4, 5], [1, 3, 5, 7, 9]]
Output: [1, 1, 2, 3, 3, 4, 5, 5, 7, 9]

Input: 3, [[1, 4, 7, 9], [2, 6, 7, 10, 11, 13, 15], [3, 8, 12, 13, 16]]
Output: [1, 2, 3, 4, 6, 7, 7, 8, 9, 10, 11, 12, 13, 13, 15, 16]
```

Question 10: PrerequisiteCourses

Given a list of courses that a student needs to take to complete their major and a map of courses to their prerequisites, return a valid order for them to take their courses assuming they only take one course for their major at once.

Examples:

```
Input: ["Intro to Programming", "Data Structures", "Advanced Algorithms", "Operating Systems", "Databases"], { "Data Structures": ["Intro to Programming"], "Advanced Algorithms": ["Data Structures"], "Operating Systems": ["Advanced Algorithms"], "Databases": ["Advanced Algorithms"] }
Output: ["Intro to Programming", "Data Structures", "Advanced Algorithms", "Operating Systems", "Databases"] or
["Intro to Programming", "Data Structures", "Advanced Algorithms", "Databases", "Operating Systems"]

Input: ["Intro to Writing", "Contemporary Literature", "Ancient Literature", "Comparative Literature", "Plays & Screenplays"], { "Contemporary Literature": ["Intro to Writing"], "Ancient Literature": ["Intro to Writing"], "Comparative Literature": ["Ancient Literature", "Contemporary Literature"], "Plays & Screenplays": ["Intro to Writing"] }
Output: ["Intro to Writing", "Plays & Screenplays", "Contemporary Literature", "Ancient Literature", "Comparative Literature"] or
["Intro to Writing", "Contemporary Literature", "Plays & Screenplays", "Ancient Literature", "Comparative Literature"] or
["Intro to Writing", "Contemporary Literature", "Ancient Literature", "Plays & Screenplays", "Comparative Literature"] or
["Intro to Writing", "Ancient Literature", "Contemporary Literature",  "Plays & Screenplays", "Comparative Literature"] or
["Intro to Writing", "Ancient Literature",  "Plays & Screenplays",  "Contemporary Literature", "Comparative Literature"] or
["Intro to Writing", "Plays & Screenplays", "Ancient Literature",  "Contemporary Literature", "Comparative Literature"] or
["Intro to Writing", "Ancient Literature",  "Contemporary Literature", "Comparative Literature", "Plays & Screenplays"] or
["Intro to Writing", "Contemporary Literature",  "Ancient Literature", "Comparative Literature", "Plays & Screenplays"]
```

Question 11: VacationDestinations

Given an origin city, a maximum travel time k, and pairs of destinations that can be reached directly from each other with corresponding travel times in hours, return the number of destinations within k hours of the origin. Assume that having a stopover in a city adds an hour of travel time.

Examples:

```
Input: [("Boston", "New York", 4), ("New York", "Philadelphia.", 2), ("Boston", "Newport",
1.5), ("Washington, D.C.", "Harper's Ferry", 1), ("Boston", "Portland", 2.5),
("Philadelphia", "Washington, D.C.", 2.5)]

Origin = "New York", k=5
Output: 2 (["Boston", "Philadelphia"])

Origin = "New York", k=7
Output: 2 (["Boston", "Philadelphia", "Washington, D.C", "Newport"])

Origin = "New York", k=8
Output: 2 (["Boston", "Philadelphia", "Washington, D.C", "Newport", "Harper's Ferry",
"Portland"])
```