Uber Career Prep Homework Assignment 3

Due: Wednesday, June 21, 2023

Graph & Data Structure Selection Problem-Solving Techniques

These questions should be completed without Googling the answer, which in most cases is easily available on the internet. That would defeat the point of the assignment, which is for *you* to practice interviewing skills. You use your own knowledge as it stands. There are two exceptions to this rule. Firstly, if you are unsure of syntax (e.g., method names, constructor arguments, etc.), you may look it up. Secondly, if you do not know a term, you may look it up. In some cases, depending on your academic schedule, you may not yet have covered a data structure yet, in which case we *encourage* you to learn and study the general concepts *before* attempting the problem.

Some questions may seem vague. This is intentional, as interview questions are often vague. Avoid making unnecessary assumptions, including based on example input/output, which is not exhaustive. If you *must* make an assumption, state your assumption in a comment in your code.

Part 1: Data Structure Implementations (~2 hrs)

Instructions

If you are unfamiliar with graph or heaps, please read the relevant articles before attempting the corresponding problems:

- Graphs:
    - Graph representations
    - Topological sorting
    - Kahn's algorithm
    - Directed vs. undirected graphs
    - Cyclic vs. acyclic graphs
    - Weighted vs. unweighted graphs
- Heaps:
    - Definition & types of heap
    - Array representation

You're welcome to consult other internet resources on the definition and use cases of these data structures, but **do not** read any implementations before attempting these problems. (If you've seen or created implementations in the past, of course that's okay, but do these problems without consulting them.)

The core implementations do not have a maximum time; you will need them to understand in order to do the rest of the assignment. If you are stuck, please get help from your mentor.

Question 1: Build an Adjacency List/Set Representation of a Graph

Given an array of pairs of values representing edges in an unweighted graph, create the equivalent adjacency list/set representation (a map from element to a list or set of elements). Pairs represent directed edges: (A, B) means there is an edge from A to B. If the pair (B, A) is also provided then there is an undirected edge between A and B. For simplicity, you may assume that each node of the graph stores an integer rather than a generic data type and that the elements are distinct. Implement a basic DFS and BFS searching for a target value and a topological sort (using either DFS or Kahn's algorithm).

```
// Build graph representation. You can also use an array rather than a set
map<int, set<int>> adjacencySet(array<pair<int, int>> edges);
// Example
Input: [(1, 2), (2, 3), (1, 3), (3, 2), (2, 0)]
Output:
{
    0: []
    1: [2, 3]
    2: [0, 3]
    3: [2]
}

bool bfs(int target, map<int, set<int>> graph);
bool dfs(int target, map<int, set<int>> graph);
array<int> topologicalSort(map<int, set<int>> graph);
```

Question 2: Build a Heap

Write a min heap class according to the following API using an array as the underlying data structure. (A max heap has the same implementation; you simply need to flip the direction of the comparators.) For simplicity, you can assume that the heap holds integers rather than generic comparables.

```
class Heap {
    private:
      array<int> arr; // the underlying array

    public:
      int top(); // returns the min (top) element in the heap
      void insert(int x); // adds int x to the heap in the appropriate position
      void remove(); // removes the min (top) element in the heap
}
```

Question 3: Build a Priority Queue

A priority queue functions like a queue except that elements are removed in order of priority rather than insertion. This is typically implemented as a max heap that stores pairs of elements and numeric weights, with the weights used as the values in the heap. Implement a priority queue according to the following API using a heap as the underlying data structure. For simplicity, you can assume the priority queue stores string elements with integer priorities. Start by copy-pasting your heap implementation from question 2 and making modifications.

```
class PriorityQueue {
    private:
      array<pair<string, int>> arr; // the underlying array

    public:
      int top(); // returns the highest priority (first) element in the PQ
      void insert(string x, int weight); // adds string x to the PQ with priority weight
      void remove(); // removes the highest priority (first) element in the PQ
}
```

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 Gustavus-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"])
```