Computer Science 2XC3 – Final Project

Instructions:

This project will include a final report and your code. Your final report will have the following.

- Title page
- Table of Content
- Table of Figures
- An executive summary highlighting some of the main takeaways of your experiments/analysis
- An appendix explaining to the TA how to navigate your code.

For each experiment, include a clear section in your lab report which pertains to that experiment. The report should look professional and readable.

- Complete all work using python programming language.
- Please read all problem descriptions carefully before beginning the work.
- Many problems require some discussion to resolve.
- Feel free to seek the help of TA's if you need clarifications on the task.
- Please do not hard code any results.
- Your submission must include a zip folder with all *.py files and a *.pdf file on Avenue.
- This project has 7 parts.

Grading Rubric: This project is total 120 points.

Part	Submission Type	Points
Part 1: Team Charter	Group	20
Part 2: Single source shortest path algorithms	Group	20
Part 3: All-pair shortest path algorithm	Group	10
Part 4: A* algorithm	Group	20
Part 5: Compare Shortest Path Algorithms	Group	30
Part 6: Organize your code as per UML diagram	Group	10
Part 7: Peer Evaluation	Individual	10

Part 1: Create a Team Charter

A team charter is a project document that aligns all team member towards how to accomplish their goal. Discuss within the team, a) the expectations from each member on being a **team player**, as well as an **individual contributor**, b) culture and norms on handling unexpected issues such as health and personal crisis, c) protocols for providing timely and constructive feedback on each other's performance to ensure team success. Answer the below questions in your team charter.

1. What will be your group's communication protocol? Discuss about the tools/technology that you will use (e.g. email, text, Teams, etc.) and a reasonable timeline of expecting a response? For instance, when there is no submission deadline, replying to an email withing 1-2 business days might be reasonable. However, this delay will hamper team progress when your work is due tonight. Please discuss these expectations and your short response below.
2. Describe below, what would be a reasonable penalty if team members *repeatedly* fail to adhere to the above communication agreement. This penalty is not to single out individuals and penalize them but to ensure that every individual is aware of the consequences of not communicating timely with their team. The penalty you suggest below will be imposed on the entire team in circumstances when individuals fail to communicate with each other.
3. Describe below the technology you will use to *collaborate* with each other (e.g Colab, VS code extensions) and *maintain version control* (e.g Git, SVN).
4. Describe below how would you resolve team disputes. The TA team and I will be available to resolve disputes that are beyond your control. However, the team members should make an honest attempt to respectfully resolve the issue amongst themselves before emailing me or the TA's. Please describe below what that attempt would look like.
5. Please state any other material you wish to include in your team contract.

Part 2 : Single source shortest path algorithms

Part 2.1: In this part you will implement variation of Dijkstra's algorithm. It is a popular shortest path algorithm where the current known shortest path to each node is updated once new path is identified. This updating is called *relaxing* and in a graph with n nodes it can occur at most $n-1$ times. In this part implement a function dijkstra (graph, source, k) which takes the graph and source as an input and where each node can be relaxed on only k times where, $0 < k < N-1$. This function returns a distance and path dictionary which maps a node (which is an integer) to the distance and the path (sequence of nodes).

Part 2.2: Consider the same restriction as previous and implement a variation of Bellman Ford's algorithm. This means implement a function bellman_ford(graph, source, k) which take the graph and source as an input and finds the path where each node can be relaxed only k times, where, $0 < k < N-1$. This function also returns a distance and path dictionary which maps a node (which is an integer) to the distance and the path (sequence of nodes).

Part 2.3: Design an experiment to analyze the performance of functions written in Part 1.1 and 1.2. You should consider factors like graph size, graph. density and value of k, that impact the algorithm performance in terms of its accuracy, time and space complexity.

Part 3: All-pair shortest path algorithm

Dijkstra's and Bellman Ford's are single source shortest path algorithms. However, many times we are faced with problems that require us to solve shortest path between all pairs. This means that the algorithm needs to find the shortest path from every possible source to every possible destination. For every pair of vertices u and v, we want to compute shortest path $distance(u, v)$ and the second-to-last vertex on the shortest path $previous(u, v)$. How would you design an all-pair shortest path algorithm for both positive edge weights and negative edge weights? Implement a function that can address this. Dijkstra has complexity $\Theta(E + VlogV)$, or $\Theta\ (V2)$ if the graph is dense and Bellman-Ford has complexity $\Theta\ (VE)$, or $\Theta(V3)$ if the graph is dense. Knowing this, **what would you conclude the complexity of your two algorithms to be for dense graphs**? Explain your conclusion in your report. You do not need to verify this empirically.

Part 4: A* algorithm

In this part, you will analyze and experiment with a modification of Dijkstra's algorithm called the A* (we will cover this algorithm in next lecture, but you are free to do your own research if you want to get started on it). The algorithm essentially, is an "informed" search algorithm or "best-first search", and is helpful to find best path between two given nodes. Best path can be defined by shortest path, best time, or least cost. The most important feature of A* is a heuristic function that can control it's behavior.

Part 3.1: Write a function A_Star (graph, source, destination, heuristic) which takes in a directed weighted graph, a sources node, a destination node, and a heuristic "function". Assume h is a dictionary which takes in a node (an integer), and returns a float. Your method should return a 2-tuple where the first element is a predecessor dictionary, and the second element is the shortest path the algorithm determines from source to destination. **This implementation should be using priority queue.** Feel free to use the heapq library for this task.

Part 3.2: In your report explain the following:

- What issues with Dijkstra's algorithm is A* trying to address?
- How would you empirically test Dijkstra's vs A*? Describe the experiment in detail.
- If you generated an arbitrary heuristic function (like randomly generating weights), how would Dijkstra's algorithm compare to A*?
- What applications would you use A* instead of Dijkstra's?

Part 5: Compare Shortest Path Algorithms

In this part, you will compare the performance of Dijkstra's and A* algorithm. While generating random graphs can give some insights about how algorithms might be performing, not all algorithms can be assessed using randomly generated graphs, especially for A* algorithm where heuristic function is important. In this part you will compare the performance of the two algorithms on a real-world data set. Enclosed are a set of data files that contain data on London Subway system. The data describes the subway network with about 300 stations, and the lines represent the connections between them. Represent each station as a node in a graph, and the edge between stations should exists if two stations are connected. To find weights of different edges, you can use latitude and longitude for each station to find the distance travelled between the two stations This distance can serve as the weight for a given edge. Finally, to compute the heuristic function, you can use the physical direct distance (NOT the driving distance) between the source and a given station. Therefore, you can create a hashmap or a function, which serves as a heuristic function for A*, takes the input as a given station and returns the distance between source and the given station.

Once you have generated the weighted graph and the heuristic function, use it as an input to both A* and Dijkstra's algorithm to compare their performance. It might be useful to check all pairs shortest paths, and compute the time taken by each algorithm for all combination of stations. Using the experiment design, answer the following questions:

- When does A* outperform Dijkstra? When are they comparable? Explain your observation why you might be seeing these results.
- What do you observe about stations which are 1) on the same lines, 2) on the adjacent lines, and 3) on the line which require several transfers?
- Using the "line" information provided in the dataset, compute how many lines the shortest path uses in your results/discussion?

Figure 1: London Subway Map

Part 6: Organize your code as per UML diagram

Organize you code as per the below Unified Modelling Language (UML) diagram in Figure 2. Furthermore, consider the points listed below and discuss these points in a section labelled Part 4 in your report (where appropriate).

- Instead of re-writing A* algorithm for this part, treat the class from UML as an “adapter”.
- Discuss what design principles and patterns are being used in the diagram.
- The UML is limited in the sense that graph nodes are represented by the integers. How would you alter the UML diagram to accommodate various needs such as nodes being represented as Strings or carrying more information than their names.? Explain how you would change the design in Figure 2 to be robust to these potential changes.

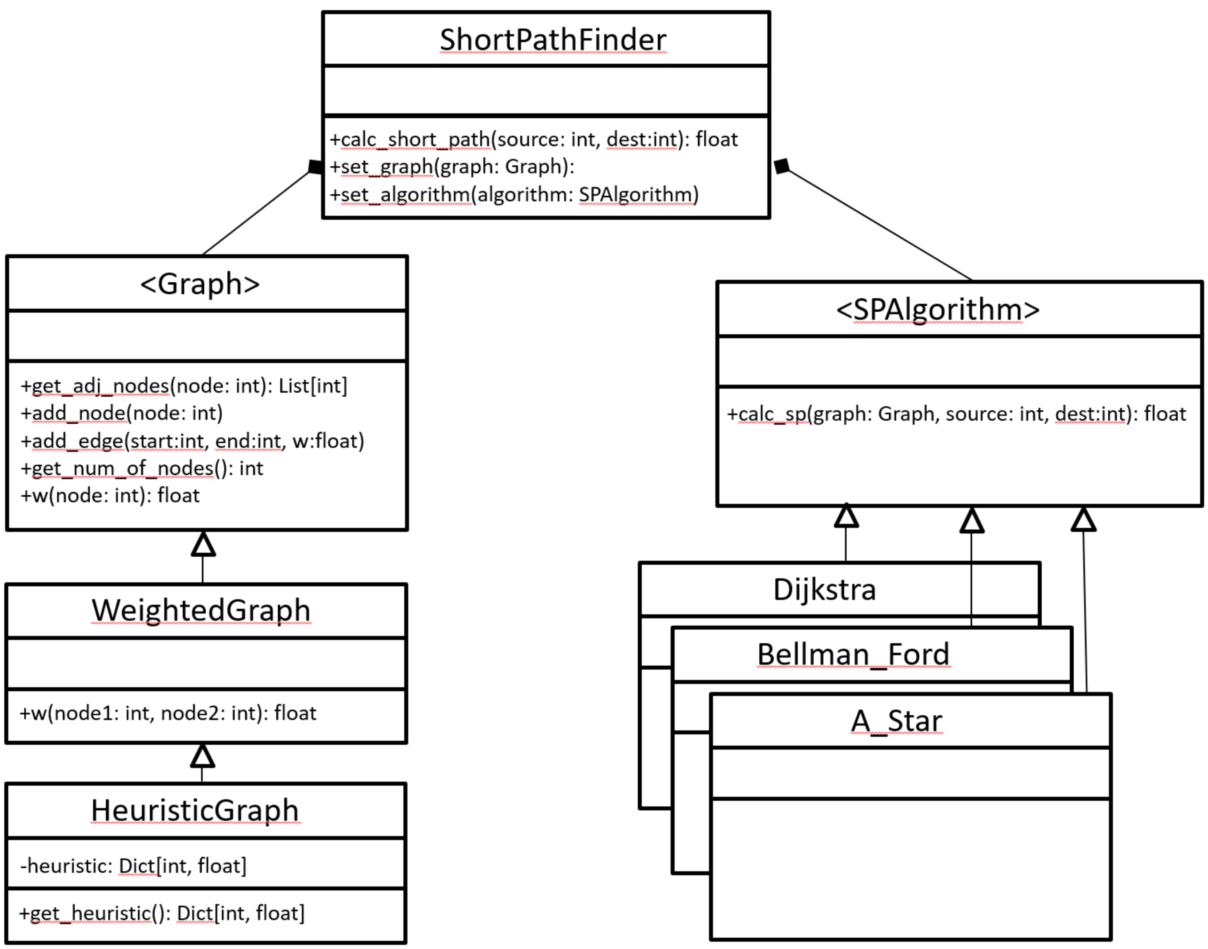

Figure 2: UML Diagram

Part 7: Peer Evaluation:

Please fill out the peer evaluation form and submit it individually. A student will receive their Part 7 grade, only *after* they submit the peer evaluation for **all** members. Please find the peer evaluation form uploaded on Avenue.