**Algorithmics Unit 3**

**School Assessed Task 2024 – The Scourge of the Flying Pangolins**

**Scenario**

Flying Pangolins (known as Pangobats) have been seen in Victoria. They are known to spread Itchy Nose Syndrome (INS), a disease which is as bad as it sounds. Your job is to design an algorithm to manage the government’s response when a Pangobat has been sighted. The Pangobat Response Force is made up of:

* a medical team, to provide a vaccine which lasts for a short period of time and reduces hospitalisation.
* a search team, to find and capture the Pangobat to carry out genomic research.
* a sanitation team, who spray the roads to prevent Pangobat droppings from being spread by vehicles.
* any other team you feel need recruiting.

When a Pangobat is reported in a particular location:

1. The response force needs to be deployed from Bendigo to reach the target site as quickly as possible.
2. The medical team must visit every town within a given radius and vaccinate everybody in the shortest possible time.
3. The search team must visit every town in that same radius to see if the Pangobat has settled there.
4. The sanitation team must traverse every road and spray it with Pangobat repellent in the shortest possible time.

You should choose **two** of the above tasks for your algorithm. You can also decide on another approach to the problem, however, it is important that you discuss this with your teacher to make sure that your idea will fulfil the criteria and be at the right level of difficulty.

Additional information:

* the medical team can quickly vaccinate the entire population in a location regardless of population size.
* the search team can recruit an unlimited number of people from the location to carry out the search. Your algorithm needs to identify how many people will be needed, and in which order in time the towns will be reached.
* there is only one medical team and one sanitation team.
* you are provided with travel times, and you can assume that there is no traffic as it is government policy to lock down population movement if a Pangobat has been reported.
* you may consider other motivations when designing your algorithm, for example, vaccinating older people first where possible.

Sample Output

Input: Mildura, 300

Output:

Travel to Mildura via Charlton and Ouyen, taking XX mins to drive XX kilometres.

Towns within 300 km of Mildura are: Mildura, Ouyen, Robinvale, Nhill, Warracknabeal, Donald, Charlton, Kerang.

The medical team should visit towns in this order: Mildura, Ouyen, Nhill, Warracknabeal, Donald, Charlton, Kerang, Robinvale, Mildura. This will take a total of XX mins to cover XX kilometres.

The sanitation team should visit towns in this order: Mildura, Ouyen, Nhill, Warracknabeal, Ouyen, Donald, Ouyen, Charlton, Ouyen, Charlton, Robinvale, Mildura. This will take a total of XX mins to cover XX kilometres.

The search team will require 8 people, and will visit (in time order): Mildura, Robinvale, Ouyen, Nhill, Warracknabeal, Donald, Charlton, Kerang.

**Data Provided**

A CSV containing a list of roads:

From, To, Length, Time (without traffic),

A CSV containing a list of towns:

Population, Median Household Weekly Income, Average Age, GPS Latitude, GPS Longitude

You do not have to use every piece of information, but you should aim to meet the criteria as comprehensively as you can. In addition, you may add your own attributes.

You are also provided with Python code which reads these CSVs and creates a model of edges and vertices representing the network.

**Report**

Your report should contain the following sections:

**1. Model and Specification**

* Clearly specify the problem, including discussion of the relevant real-life information you intend to model.
* State which ADTs you will use to model the problem.
* Provide signatures for the operations for each ADT.

You are provided with Python code which performs the following:

* reads the CSVs
* creates a graph object containing nodes and edges
* creates nodes with attributes name, lat, long, pop, income and colour (a number from 0 to 5)
* creates edges with attributes place1, place2, dist, time, and colour.
* displays the graph object

Depending on how you model the problem, you will need to create additional ADTs. You will also need to make some decisions about how you interpret the problem.

For example:

* if you are designing your algorithm to take age into account, you might decide to store this information in another ADT.
* when storing e.g. the distances for Dijkstra, you’ll need to decide whether to store these in dictionaries or as node properties. v

**2. Design your Algorithm**

* Explain why you chose a particular approach, comparing features of your algorithm to other options.
* Clearly explain the broad overview/approach of each algorithm.

**3. Code your Algorithm**

* Write your code in Python, including comments which you might expect to see written as pseudocode. It should be possible for someone who doesn’t understand Python to read your pseudocode comments and fully understand how your algorithm works.

The criteria are given below. Criteria 2 and 4 are marked in the ‘design your algorithm’ section above. Criterion 2 assesses how well you explain what you are doing. Criterion 4 assesses how well you justify it in comparison to other options.

You must code your project in Python in order to demonstrate that it works.

**Authentication**

* This is an individual project. You can discuss it with others, but the work must be your own.
* Students must be able to explain their work if called upon.
* You must acknowledge any external resources you draw upon.
* You are not being assessed on your Python coding abilities, so you can ask for help and/or use generative AI to create Python code according to your own pseudocode.

**Links to Unit 4**

In Unit 4, you will analyse your algorithm for time complexity, and then use advanced algorithm design features to improve it. Such features include:

* dynamic programming
* backtracking
* hill climbing
* simulated annealing
* A\*

You should avoid using these features in Part 1 of the SAT, and you are advised to stick to designs you have seen in Unit 3. Note also that if your algorithm in Part 1 contains an intractable component, this can make it easier to improve in Parts 2 and 3 (however this doesn’t mean you should make your algorithm deliberately inefficient – e.g. using brute force to find a shortest path between two points, when we know plenty of single source path algorithms, would not help you to achieve top marks.)

**Other tips**

* Do not change the data you are provided with, even if you discover factual inaccuracies.
* Avoid focusing on statistical arguments in your evaluation. For example, we know there are advantages and disadvantages to using a median age, but this is not what this course assesses.
* Keep an eye on the news forum for further advice that will undoubtedly arise as students work on the project.
* Build your program in small pieces, testing as you go. Use “print” statements in your code to make sure the value you are storing are the ones you expect.
* Test your program on small problems that you can easily work out by hand.
* You can use pre-existing packages for things not explicitly covered in the course. Ask if unsure, but for example: you must show all detail of Dijkstra, but you can use list sort functions and permutation tools without showing how they work.
* You do not need to worry about a user interface or pretty graphics, however, it is possible to change the colours of edges and nodes if you want to display your results for ease of checking.

**Timeline**

**Your final model and report is due on 31st May 2024.** It should be submitted via VSV online.

Your teacher can provide ongoing support if you run into difficulties. You can submit a draft once, at any point before 10th May 2024, to receive comprehensive feedback on how you can better meet the criteria.

**Assessment Criteria**

Criterion 1: specify a problem and model its key features. Precisely specify the algorithm problem you intend to solve.

Criterion 2: design an algorithm to solve your problem, considering multiple approaches and combinations of algorithm patterns.

Criterion 3: communicate your algorithm using pseudocode.

Criterion 4: evaluate your model, considering the suitability of your algorithm, how it integrates with your model and whether or not it solves your original problem. (Formal analysis will come later in Unit 4).

You will receive a mark out of ten for each criterion, based on the following rubric.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 1-2 (very low) | 3-4 (low) | 5-6 (medium) | 7-8 (high) | 9-10 (very high) |
| 1 | Specifies an algorithmic problem. | Identifies some algorithmic aspects of the real world problem. | Outlines/explains some algorithmic aspects of the real world problem. | Formulates an algorithmic problem from the real world problem. | Explains how the algorithmic problem is formulated from the real world problem. | Provides a clear and precise specification of the algorithmic problem, formulated from the real world problem |
| 1 | Explains the salient features of the real world problem. | Lists arbitrary features of the real world problem. | Identifies some relevant features of the real world problem, giving reasons. | Describes salient features of the real world problem, giving reasons. | Chooses a suitable set of features to model based on consideration of the features of the real world model. | Chooses a **comprehensive** set of features to model based on consideration of the features of the real world model |
| 1 | Models the problem using ADTs. | Makes a limited attempt, ADTs may not be suitable. | Models some features using suitable ADTs. | Models selected features using a combination of suitable ADTs. Outlines how the problem maps to this model. | Models selected features using a **coherent** combination of suitable ADTs. Describes how the problem maps to this model. **Some signatures** are provided. | Models selected features using a **coherent and fit-for-purpose** combination of suitable ADTs. Describes how the problem maps to this model. **All signatures** for key operations are provided. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 1-2 (very low) | 3-4 (low) | 5-6 (medium) | 7-8 (high) | 9-10 (very high) |
| 2 | Considers suitable algorithmic approaches to the problem | Identifies an algorithm with some relevant to the problem | Outlines a few algorithm design approaches that could form the basis of a solution. | Considers relevant characteristics of several algorithm design approaches. | Compares the suitability of some algorithm design approaches to determine an appropriate approach. | **Thoroughly** compares the suitability of algorithm design approaches to determine the best approach. |
| 2 | Describes the design of an algorithmic solution to the real world problem | Identifies some aspects of an algorithm to solve a real world problem. | Outlines a simple solution to the problem. | Describes a non-trivial algorithm that some some aspects of the specified problem. | **Describes** an algorithm that solves the specified problem using some combination of algorithms or algorithm design patterns. | **Clearly explains** an algorithm that solves the specified problem using some combination of algorithms or algorithm design patterns. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 1-2 (very low) | 3-4 (low) | 5-6 (medium) | 7-8 (high) | 9-10 (very high) |
| 3 | Communicates the algorithmic solution in pseudocode | Limited elements are expressed in pseudocode. Some correct initialisation of variables and data structures. | Elements of the structure are expressed in pseudocode. Correct use of simple iteration and conditional control structures where appropriate. | Algorithm is expressed in pseudocode so that the structure of the design is clear. Includes nested iteration and recursion where appropriate. | Algorithm is expressed in pseudocode that mostly reflects the solution design. Errors are minor and do not affect the overall structure. Some attempt to use functional abstractions. | Algorithm is expressed in pseudocode correctly and precisely. A modular approach is used including ADTs and functional abstractions. |
| 4 | Justifies the solution to the real world problem. | Identifies relevant reasons in support of a solution. | Outlines the rationale for a solution based on limited consideration of pros and cons. | Justifies choice of solution based on comparative advantages over other approaches. | Justifies choice of solution based on arguments relating to its suitability, coherence or fitness for purpose. | Justifies choice of solution by clearly demonstrating its suitability, coherence or fitness for purpose. |