CAB203 PROJECT

20/5/2021

CAB203 Discrete Structures
Benjamin Rogers
N10212868

1. INTRODUCTION

Bob has constructed a family tree of everyone on the spaceship that goes back to the first generation. The tree is in the form of cards that have the person's name, father, mother, and children. An example of the first two cards can be seen below:

Name:	Alice
Father:	Arlo
Mother:	Madeline
Children:	None

Name:	Bob
Father:	Charlie
Mother:	Eve
Children:	None

Bob is wanting to simplify his family tree by finding the most direct relationships between himself and every other person onboard. A child-parent relationship is the most direct and is referred to as a step. Each other relationship is measured by how many child-parent relationships it takes to get from a person back to Bob. In the simplified tree, there will be a * to mark the relationships that are closest to bob. In other words, the problem is to produce a new family tree so that Bob can quickly see the most direct relationship he has with any other person.

2. INSTANCE MODEL

An instance of the problem can be shown as a set of child-parent steps in the tree, and a starting person Bob. Let L be the set of all child-parent steps in the family tree. Two people are adjacent and form a step (u, v) if they have a child-parent relationship. The test data can be modelled by

$$L = \{(Bob, Charlie), (Bob, Eve), (Alice, Arlo), (Alice, Eve), (Eve, Oliver), (Eve, Aurora), (Charlie, Jack), (Charlie, Luna), (Madeline, Jack), (Madeline, Aurora), (Arlo, Oscar) (Arlo, Isla), (Oliver, Hugo), (Oliver, Rose), (Luna, Oscar), (Luna, Isla), (Aurora, Hugo), (Aurora, Rose), (Jack, Oscar), (Jack, Rose)\}$$

The instance model also must take into consideration Bob as the starting position. Therefore, the instance is modelled by (L, s) where s is Bob as the starting position. The complete instance with s can be seen here:

$$(\{(Bob, Charlie), (Bob, Eve), (Alice, Arlo), (Alice, Eve), \dots\}, Bob)$$

3. SOLUTION MODEL

To use the family tree network, we need to identify which child-parent steps to use and in what order to reach every other person in the tree with the minimum steps required. The first child-parent step must have the starting node s from the instance model. To encode the simplified family tree mathematically, the solution will be a set of key-value pairs in tuples that represent a step between two people. This will look like {(Bob, false), (Charlie, Bob), (Eve, Bob) …}. The first tuple with the starting vertex Bob is the root of the graph. From this we will then have a path between each person and can print the updated cards. In the updated cards when the shortest relationship path to Bob needs to be marked, there will be a * next to the name of the person. For

example, if there was a shortest path from Bob -> Eve -> Aurora, then Eve's card would have Aurora's card would have a star next to Eve and on Eve's card there would be a star next to Aurora.

4. PROBLEM MODEL

Given an instance (L, s), we can model the family tree network as $G = (V, E)$. The set of vertices V is the set of people in child-parent pairs in L. This can be shown by:

$$\mathrm{V} = \{\mathrm{u}: (\mathrm{u}, \mathrm{v}) \in \mathrm{L}\} \cup \{\mathrm{v}: (\mathrm{u}, \mathrm{v}) \in \mathrm{L}\}.$$

E in $G = (V, E)$ represents an edge between each pair of people in L. E can be described as the following:

$$\mathrm{E} = \mathrm{L} \cup \{(\mathrm{v}, \mathrm{u}) : (\mathrm{u}, \mathrm{v}) \in \mathrm{L}\}.$$

It was mentioned that everyone in the network is related in some way. Therefore, the graph is connected. The edges in this graph are not weighted. To get from the starting person s to any other person in the tree with the fewest steps, we will need to construct a spanning tree. Therefore, the solution will be a simplified subgraph of the original family tree where any path from starting point s to another person is marked if it is done in the minimum number of steps. In cases where two people have more than one path between them of the same length, only one is marked.

5. SOLUTION METHOD

Breadth first search can be used to find the shortest path between the starting position s (Bob) and each other person in the family tree. This can be done by building a spanning tree that from the original graph with the minimum possible number of edges. Firstly, we must organize the set of people into distances classes. If we set the initial vertex as v, the distance classes can be defined as a set of vertices that are distance j from v. The length of the shortest path between s and a particular node d in the graph, can be seen as $\ell(d, v)$.

$$D_j = \{v \in V : \ell(d, v) = \mathrm{j}\}$$

When traversing the tree, there will need to be some way of keeping track of vertices have not been passed through V_j which is a helper set. V_j will contain a set of vertices that are distance j from starting vertex v. This can be shown mathematically as:

$$V_j = V \setminus (D_0 \cup D_1 \cup D_2 \ldots \cup D_{j-1})$$

To setup the recursion needed, we start by defining the base cases. Before traversing the tree, we must start initializing the minimal spanning tree with a single vertex which is Bob as he is our starting position.

$$V_0 = V$$

$$D_0 = \{d\}$$

$$parent = \{(d, false)\}$$

V_0 represents the vertices in the graph which will be used later in a helper function. D_0 represents the list of all the distance classes. The d in D_0 is where we set the initial vertex to Bob. Lastly, $parent$ will be set of key value pairs that form the spanning tree. We initially set $parent$ to have Bob as its key and false as its value since Bob has no children.

Now the recursive case can be defined for algorithm.

$$V_j = V_{j-1} \setminus D_{j-1}$$

V_{j-1} represents all the vertices that have a distance of at least $j - 1$. Referring to the expression above, the vertices v in D_j are those with a path length of j, between d and v. A path in the tree starts at s and goes through a vertex u in D_{j-1} where (u, v) creates the edge in E.

$$D_j = N_{V_j}(D_{j-1})$$

By doing this we calculate successive sets of D_j that are increasing distance from s and we can soon find a path from Bob to every person in the tree. Now that we have our distance classes, we need to put together the spanning tree. For each of the distance classes, there could be more than one vertex that could be used at each step so there need to be a way to arbitrarily pick one. We can create a function called arbitraryElement(S) that returns a single element of a set S. To check if the recursion is complete, instead of checking if we have found a vertex, we check if $V_j = 0$ or if $V_{j-1} = 0$. If $V_{j-1} = 0$ this means that there are no more neighbors and that the graph is disconnected. If the recursion is not complete and $V_j \neq 0$ and $V_{j-1} \neq 0$, then we loop through D_j and add a key-value pair from the distance classes to the spanning tree $parent$ where $v \in D_j$.

$$parent(v) = arbitraryElement(N_{D_{-1}}(v))$$

The subscript D_{-1} here is the vertex set and the edge set E was mentioned earlier. Once this is repeated until $V_j = 0$ or $V_{j-1} = 0$ then we will have a completed spanning tree from key-value pairs of tuples in $parent$, which will show us the minimum path from the starting vertex Bob to every other person. This can then be used to print out the updated cards where the relationship closest to Bob is marked with a *.