Erin Crenshaw

Sweety Grewal

David Lee

Ngoc Nguyen

**Finding the Most Efficient School Bus Route**

Almost every weekday, school parking lots are lined with school buses dropping off or picking up students during the morning and afternoon hours. As the school day comes to an end, students eagerly pour out of their classes and many scatter on to different school buses ready to take them home. Sometimes only a few students enter a school bus before it gets ready to leave and drop off the students at their respective bus stops. It seems as if the empty seats could have been occupied by students from another half-empty school bus to eliminate operating more buses than necessary. Could the use of less buses decrease the cost transportation and increase school bus efficiency? With recent budget cuts caused by economic woes, it is more important now for schools to spend money efficiently and effectively on necessities such as transportation for students. Our group decided to create a program that tries to find the most efficient bus route(s) for a school based on different input provided by the user. The user enters in the name of each bus stop, the cost to travel to neighboring bus stops from the current bus stop, and the maximum number of students picked up at each stop. From this information, the program returns the route that is the most cost efficient based on the total cost calculated from traveling from one bus stop to the next. Since buses have a maximum capacity for the amount of students that can be picked up, the program counts the number of students picked up from each bus stop and if the number of students loaded on a bus exceeds the maximum capacity, a new bus is added to the route.

We gathered data from a local school to get an idea of how much operating school buses can cost per day for a given school. The cost depends on the size of the school; the larger the number of students attending a school, the higher the number of buses that get sent out to pick up and drop off students. We gathered data from Alderwood Middle School which currently has over 700 students enrolled. From the information we received, we found that it costs Alderwood Middle School a little over $3,000 per day to operate school buses that pick up and drop off students. The yearly cost of operating the school buses for the middle school is over $500,000. The Edmonds school district faces expenses equating close to two million dollars a year to operate all of its school buses. This figure makes a big dent in the yearly budget of the school district which is why it is important to find efficient routes and an efficient number of school buses operated in order to keep the costs down.

The program that we have written is made to find the most efficient route(s) depending on the number of bus stops around the school and the number of students that need to be picked up by the school buses. The user needs to enter three different types of information for the program to work. The user needs to first assign a name to the bus stop. This is entered as a string and the user also needs to include the location of where the buses are originating from (for the morning route, this origin may be the bus depot) and also where the bus’s final destination is, which would be the school for the morning route. The next input that the user needs to provide is the name of the neighboring bus stops and how much it costs to travel from the current bus stop to the neighboring bus stops. The last piece of information that the user provides is the maximum number of students being picked up from each stop. Since, the number of students at each bus stop can vary from day to day due to absence, we ask for the maximum amount of students at the bus stop to prevent the possibility of exceeding the maximum capacity of the bus. After the requested input is fed in to the program, the program generates the results of the most efficient bus route and the number of buses that should be sent out to pick up all the students.

After the program receives the input from the user, it attempts to find all the possible routes to get from the bus depot to the school. For each route, the program calculates the cost of travel and the total number of students picked up. If the total number of students exceeds the default value of the maximum capacity of the bus, another bus is added to the route. The program gathers results which includes having more than one bus sent out to pick up students. The program is designed to gather data until the number of buses operating is equal to the total number of different routes possible. This prevents duplicate results and prevents the number of buses being sent out to be a large number. When another bus is added to pick up students due to the original bus reaching maximum capacity, the total cost of operation is doubled because we assume that the additional bus will follow the same route as the original bus (which is now at full capacity). When the program is finding the possible routes, it does not necessarily include all of the bus stops. Therefore, the program will go through all the possible routes that have been found and choose the ones that include all of the bus stops. From these possible routes, the program will print the solution that has the lowest total cost. The output will include the path of the route, total cost, total number of students picked up, and the number of buses required.

The total cost that the program calculates is only based on the users input of the cost of fuel to get from one bus stop to the other. We do not factor in the cost of labor for each bus or the cost of maintenance and repairs for the buses. Therefore, the route that the program returns as the most efficient may actually cost more than other routes. For example, if the program finds that a route with two school buses is more cost efficient than a route with only one school bus, the user may be given incorrect information. This is because the program does not include the labor costs of each additional bus. The rise in the total cost due to cost of labor may exceed the cost of the other route that has less buses running.

The input that is required from the user is the name of the stop, number of students at the stop, and a dictionary of neighbors with how much it costs to get to that neighboring node. For example, if buses will be leaving a bus depot, picking up students from six different bus stops, and then dropping them off at school, the input provided by the user would look like the following:

input:
'depot', {'1':10, '2':15}, 0
'1', {'depot':10, '2':5, '3':10, '4':11}, 6
'2', {'depot':15, '1':5, '4':4, '5':7}, 8
'3', {'1':10, 'school':10}, 7
'4', {'1':11, '2':4, '5':5, '6':8}, 14
'5', {'2':7, '4':5, '6':9}, 2
'6', {'4':8, '5':9, 'school':12}, 2
'school', {'3':10, '6':12}, 9

The first part of the input is the name of the node, which is taken as a string. The part inside the curly braces gives information about the neighboring nodes and the how much it costs to get from the current node to the neighboring node. For example, for the first input, the name of the node is “depot” which corresponds to where the buses will be leaving from. Inside the curly braces, the ‘1’ corresponds to the nearest node to the bus depot and the “10” corresponds to how much it costs for the bus to get from the bus depot to node 1. The ‘2’ corresponds to node 2 which is also accessible directly from the bus depot and it costs 15 dollars to travel from the bus depot to node 2. The last number in the input corresponds to the maximum number of students picked up at the bus stop. The user enters in this information for all of the bus stops. The program then finds the most efficient route and the cost of the bus route found. If the number of students picked up by the bus exceeds the maximum capacity of the bus, another bus will be added to pick up the students from the stop(s) the full bus was not able to go to. The output of the program is similar to the following:

[['depot', '2', '5', '6', '4', '1', '3', 'school'], 70, 46, 1]

The user is given the most efficient route in terms of total cost of traveling from node to node. The bracketed list corresponds to the route the bus takes, the first integer is the total cost of the route, the second integer corresponds to the number of students picked up by the bus, and the last integer is the total number of buses needed. Visually, this is what the current scenario looks like the following with the most efficient route displayed with the red arrows:

10

10

10

3

1

**Your text here**

**Your text here**

Bus Depot

11

7

5

12

15

9

8

4

5

2

4

6

5

School

Initially, we had hoped to create a program that would be able to print a graphical representation of the most efficient route(s) that the program found. However, we were unable to implement this into our program because the unavailability of resources in Python to do so and the lack of knowledge as to how to do so in Sage. Since most of us are beginning programmers, there were many limitations on what we were able to create in Python. Also, we had initially hoped to print a chart for each bus stop that could potentially be given to each bus driver so he/she would know which stops they would be traveling to and how many students they would be picking up from each stop. We wanted to calculate the total cost of operation by including the cost of labor, maintenance, traffic, etc. Due to time constraints and the inability to gather data in a timely manner from a third party, we were unfortunately unable to include this part in our program as well. What made our project difficult was not being able to create a realistic solution in which the streets the bus stops are located on are used to create a real time map of the route the bus would take.

There isn’t any mathematical background that is needed to understand the program. A feature that we did add to the program was having the ability to load and save from a text file which contains the data of all the bus stops. Instead of having the user enter individual input for each school bus stop, the user is now able to save the data from all the bus stops (nodes) and have the program read the data itself from the text file. This makes it less tedious for the user, since he/she will not have to re-enter the same data every time they want to simulate the program.

There are other factors that determine the most efficient school bus route(s) that we did not take into consideration. The cost of diesel is very important in the total cost of the bus routes. Also, the maximum capacity of a school bus will determine how many school buses must be operated to pick up all the students from the bus stops. We used a default value of 72 students as the maximum capacity of the school bus. We used this value because after doing research, we found that the large school buses are able to carry 72 students on average. However, the value for the maximum capacity will vary depending on what size buses are available for the schools and how many buses of the different sizes are available for each school. We unfortunately did not consider this in our program, but it is something that can be added to make the program more realistic.

Other information that needs to be considered in order to find the most efficient bus route(s) includes how fast the bus is traveling. If the bus travels at a slower speed, it will take the bus longer to get from one stop to the next and it will also increase the cost of labor, if the driver is paid on an hourly basis. Also, we need to consider delays due to the time it takes for the students to load onto or off of the bus and traffic delays. The cost for the bus routes that is returned by the program only takes into consideration the cost of the morning route. The cost should be multiplied by two if the same bus will be going to the same stops to drop off the students after school. We also should consider how long the bus takes to get to each route. Even though having one bus go to all of the bus stops for a certain school may be the most cost effective solution, it may be an inconvenience if the total trip takes more than a half an hour for example.

This program was created in hopes to help schools find an efficient bus schedule in order to minimize costs and maximize efficiency of resources available (for example, the number of buses operated). There are ways in which our program could be improved and made more useful for the user; however it does provide the user with enough information to get an idea of what the most efficient school bus solution will be. Due to time constraints and not having enough programming experience, we were unable to make the program print more useful data, like we had initially planned for. We hope that schools are able to use a program similar to ours to evaluate their school bus routes to make sure the most efficient school bus routes are used and the cost of transportation is minimized.