CSCE 3110

Project Assignment

Jared Singleton
Yousif Al Hamadani

The design of this project focused on two main algorithms that were necessary in order to accomplish the desired outputs. The main algorithm used was Dijkstra's algorithm which provided us with the shortest path between nodes in a graph. The second algorithm was the permutation algorithm that provided us with every possible path to take between a specific number of nodes.

When working on the implementation of these two algorithms we encountered a few design issues with the WA.cpp file. The first issue was the "map data structure" that was used to create the "City_file" that stored every city along with a unique integer assigned to each city. The function that was given would work perfectly for flight graphs up to 139 cities. But when a flight graph had 140 cities, it would reset the unique integer for the last city (Asuncion, Paraguay) to 0. This would cause the program to save all cities that Asuncion, and Paraguay could fly to be saved under the city with the unique integer 0. This issue only happened when there were 140 cities. We tested it with 139 or fewer cities and the function worked perfectly. We eventually solved this issue with an "if" statement after the function that would set the integer value for Asuncion to 139 if the number of cities equaled 140. Once this major issue was resolved, we were able to move forward with incorporating the Dijkstra algorithm into the program.

```
void createCityFile(int n){
    ifstream cityFile;
    string line;
    cityFile.open("city.name", ios::in);
    for (int i = 0; i < n; i++){
        getline(cityFile, line);
        line.erase(std::prev(line.end()));
        city[line] = i;
        cities[i] = line;
    }
    /*
        loop does not input correct number value
        for the last city "Asuncion, Paraguay"
        so if n = 140, then the if statement
        inputs the correct value for the city
    */
    if(n == 140)
        city["Asuncion, Paraguay"] = 139;

    cityFile.close();
}
```

Figure 1*:* Function used to create the city file had an issue when there were 140 cities. An "if" statement was used to fix this issue

The Dijkstra algorithm consists of two different varients. The original algorithm found the shortest path between two given nodes, but the more common variant fixes a single soure node and finds the shortest paths from the source node to all other nodes in the graph. Out program uses the more common variant because the data that it produces helped in one way or another to solve all four questions for this project. There three main data structure arrays that are used by the Dijkstra algorithm to search, test, and save each individual path. A "distance array" is used to save the total distance form the source node to all other nodes. The "visited array" is used to ensure that a node is not visited more than once. Lastly is the "parent array" that saves the unique integer of the parent node to the child node. As shown in *Figure 2*, these data structures allow for storing and querying partial solutions sorted by distance from the source node.

```
bool Graph::Dijkstra(int src, int distance[], bool visited[], int parent[]){
    for(int i = 0; i < n; i++)
        distance[i] = INFINITY; // set every cities distance to an initial value of 999
    distance[src] = 0;// set city-A distance from itself to 0

    // the 1st 'nearest' value will always be city-A and it then creates an imaginary queue
    // by using minimumDist() to grab the next city in the queue and then calculate
    // the distance from city-A to the current cities children
    for(int i = 0; i < n; ++i) {

        int nearest = minimumDist(distance, visited);
        visited[nearest] = true;
        //cout << "\tdistance[j]:\t" << distance[nearest] << endl;
        for(int j = 0; j < n; ++j){

            if(!visited[j] && adj[nearest][j] && distance[j] > distance[nearest] + adj[nearest][j]) {
                distance[j] = distance[nearest] + adj[nearest][j];
                //cout << j << "\tdistance[j]:\t" << distance[j] << endl;
                parent[j] = nearest;
            }
        }
    }
    return possibleRoutes(src);
}
```

Figure 2*:* The Dijkstra algorithm is very well known for its ability to find the shortest path from a source node to all other given nodes. This variant has a time complexity of Big-Oh(V^2) where |V| is the number of nodes.

Our variant of the Dijkstra algorithm has a time complexity of Big-Oh(V^2), where |V| is the number of nodes. This is not the fastest known single source shortest path algorithm, but it is the one that we understand the best and could incorporate into our program.

We also incorporated a second function into the algorithm to return a Boolean value on whether the source node can fly to at least one other city. This resolved a design issue where we would have questions that required specific nodes to be visited and we needed to know if those nodes could travel to at least one other node.

The permutation algorithm was primarily used for questions 2 and 3 which involved having specific nodes that must be visited. It is not a super complicated function since its only purpose was to take an array of nodes and create every possible permutation of those nodes. The time complexity of this function is Big-Oh(V!) where |V| is the number of must visit

```
void permutation(string **temp2D, string arr[], int n, int l){
    // starting value of l = 0
    if(l == n-1) {
        for(int i = 0; i < n; i++) {
            temp2D[count][i] = arr[i];
        }
        // global variable to help track
        // which row to save the city name
        ++count;
    }
    for(int i = l; i < n; i++) {
        swap(arr, i, l);
        permutation (temp2D, arr, n, l+1);
        swap(arr, i, l);
    }
}
```

Figure 3: The permutation function allowed us to take an array of nodes that must be visited and create every possible permutation of those nodes.

nodes and is governed by the equation: T(n) = n * T(n-1) + O(1).

A big design issue with the permutation function is the high time complexity. With question 2 we incorporated a restriction that only allowed the user to select up to 3 different must visit cites. This kept the run time down, as well as the number of times the user would have to select a node to be used in the program. When working on question 3, we noticed that every node with more than 7 adjacent nodes would throw the following error: *terminate called after throwing an instance of 'std::bad_alloc'.* This error occurs due to memory allocation failure and is happening because we are trying to save every possible path onto a 2D dynamic array. Our solution for this is to save just one path onto an array and compare this path with every new path that is created and keep the one that is fastest. This drops the memory allocation from |V!|, where |V| is the number of permutations, down to 2.

We used a variety of Data Structures throughout our program to accomplish specific tasks in order to answer the 4 questions required in this project. The main Data Structures and their uses can be seen in *Table 1*.

Data Structures used in the project	
City map	Saved a unique integer to every city in the graph.
Distance array	Saved the distance from the source city to every other given city in the graph.
Parent array	Save the parent city to every child city in the graph. This created a path back to the source city.
Visited array	Kept track of which cities had been visited or not visited.
Temp2D dynamic array	Saved all possible permutations for a specific list of cities to visit.
Arr2D dynamic array	Saved the path for every possible permutation.
Cities array	Saved every city in the graph.

Table 1: A list of Data Structures used in our program along with why we chose to use them.

When looking at the time complexity of our algorithms in table 2, we can see that the scalability implementation of our program would only work with a large-scale geographic area when we focused on questions 1, 2, and 4. Looking at question 3 we can see that the issue with the permutations is already causing an issue. So, the program would not function correctly in a large-scale geographic area. During our research, we found that there is a way to optimize

the Dijkstra algorithm to include a list of must visit nodes into the search parameters. But we ran out of time when trying to incorporate this into our program. But, when it comes to finding the shortest path to visit a large number of nodes, then the following algorithms would have been better to use: Held-Karp, Brand-and-Bound, or Cutting-plane Method. Unfortunately, with our project deadline, we did not have the time to try and optimize our code with one of these methods. So, we decided to stick with the Dijkstra "Brute Force Method" and calculate permutations in order to find the shortest path for question 3.

Time Complexity of our Algorithms	
Question 1	Big-Oh(1)
Question 2	Big-Oh(n*n!)
Question 3	Big-Oh(n*n!)
Question 4	Big-Oh(1)
Dijkstra Algorithm	Big-Oh(n^2)
Permutation Algorithm	Big-Oh(n!)

Table 2: The time complexity for all 4 questions the project answers and the two main algorithms used in our program.

In Conclusion, we can confidently say that for small geographical areas our program will work perfectly when finding the shortest path for all 4 questions. When the geographical area becomes larger, then the program will work perfectly for questions 1, 2, and 4. But, question 3 runs into an issue with the workload size and we did not have enough time to fix this design issue.

Screenshots of test running results

```
Script started on 2022-08-07 10:25:33-0500
]0;js1769@cse01: ~/CSCE3110/project]0;js1769@cse01:~/CSCE3110/project$ g++ WA.cpp
]0;js1769@cse01: ~/CSCE3110/project]0;js1769@cse01:~/CSCE3110/project$ ./a.out
Select how many cities you want in the graph
 1) 20  cities
 2) 50  cities
 3) 100 cities
 4) 140 cities
input: 1

The graph generated can be represented by the following adjacent list:
--------------------------------------------------------------------------------
Moscow, Russia
   0 -> 3 4 8 12 16 18 19
Seoul, South Korea
   1 -> 0 3 9 10
Tokyo, Japan
   2 -> 7 12
Hong Kong, SAR
   3 -> 2 6 9 15 19
London, United Kingdom
   4 -> 19
Osaka, Japan
   5 -> 7 10 11 13 15 17 18
Geneva, Switzerland
   6 -> 19
Copenhagen, Denmark
   7 -> 3 14 19
Zurich, Switzerland
   8 -> 3 6 11
Oslo, Norway
   9 -> 0 2 5 11 12 17 18
New York City, United States
   10 -> 5 7 14 16 19
St. Petersburg, Russia
   11 -> 0 1 2 5 8 13
Milan, Italy
   12 -> 0 2 3 4 5 7 10 11 14
Beijing, People's Republic of China
   13 -> 18
Istanbul, Turkey
   14 -> 0 3 6 9 10 19
Paris, France
   15 -> 1 4
Singapore, Singapore
   16 -> 9
Dublin, Ireland
   17 -> 0 2 3 11
Sydney, Australia
   18 -> 3
Shanghai, People's Republic of China
   19 -> 3 5 7 8
```

```
--------------------------------------------------------------------------------
Shanghai, People's Republic of Chinasrc = 19

city  Distance    Path
--------------------------------------------------------------------------------
0  3      0  <- 9  <- 3  <- 19
1  3      1  <- 11  <- 5  <- 19
2  2      2  <- 3  <- 19
3  1      3  <- 19
4  3      4  <- 15  <- 3  <- 19
5  1      5  <- 19
6  2      6  <- 3  <- 19
7  1      7  <- 19
8  1      8  <- 19
9  2      9  <- 3  <- 19
10 2      10  <- 5  <- 19
11 2      11  <- 5  <- 19
12 3      12  <- 2  <- 3  <- 19
13 2      13  <- 5  <- 19
14 2      14  <- 7  <- 19
15 2      15  <- 3  <- 19
16 3      16  <- 10  <- 5  <- 19
17 2      17  <- 5  <- 19
18 2      18  <- 5  <- 19
19 0      19
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Select which question you want to do
 1) Question 1
 2) Question 2
 3) Question 3
 4) Question 4
 5) Quit
Input: 1
```

```
88  ------------------------------------------------------------------------
89  Pick your starting city
90    1) Moscow, Russia
91    2) Seoul, South Korea
92    3) Tokyo, Japan
93    4) Hong Kong, SAR
94    5) London, United Kingdom
95    6) Osaka, Japan
96    7) Geneva, Switzerland
97    8) Copenhagen, Denmark
98    9) Zurich, Switzerland
99    10) Oslo, Norway
100   11) New York City, United States
101   12) St. Petersburg, Russia
102   13) Milan, Italy
103   14) Beijing, People's Republic of China
104   15) Istanbul, Turkey
105   16) Paris, France
106   17) Singapore, Singapore
107   18) Dublin, Ireland
108   19) Sydney, Australia
109   20) Shanghai, People's Republic of China
110 Input: 2
111
112 Pick your ending city
113   1) Moscow, Russia
114   2) Seoul, South Korea
115   3) Tokyo, Japan
116   4) Hong Kong, SAR
117   5) London, United Kingdom
118   6) Osaka, Japan
119   7) Geneva, Switzerland
120   8) Copenhagen, Denmark
121   9) Zurich, Switzerland
122   10) Oslo, Norway
123   11) New York City, United States
124   12) St. Petersburg, Russia
125   13) Milan, Italy
126   14) Beijing, People's Republic of China
127   15) Istanbul, Turkey
128   16) Paris, France
129   17) Singapore, Singapore
130   18) Dublin, Ireland
131   19) Sydney, Australia
132   20) Shanghai, People's Republic of China
133 Input: 14
134
135 Input the maximum amount of connecting flights: 5
136 ------------------------------------------------------------------------
137
```

```
Shortest route from "Seoul, South Korea" to "Beijing, People's Republic of China".
In under 5 connections

Seoul, South Korea
 -> Oslo, Norway
 -> Osaka, Japan
 -> Beijing, People's Republic of China

Total connections: 3
--------------------------------------------------------------------------------

Input paused, press 'Enter'

--------------------------------------------------------------------------------
Select which question you want to do
 1) Question 1
 2) Question 2
 3) Question 3
 4) Question 4
 5) Quit
Input: 2
--------------------------------------------------------------------------------
Pick your starting city
  1) Moscow, Russia
  2) Seoul, South Korea
  3) Tokyo, Japan
  4) Hong Kong, SAR
  5) London, United Kingdom
  6) Osaka, Japan
  7) Geneva, Switzerland
  8) Copenhagen, Denmark
  9) Zurich, Switzerland
  10) Oslo, Norway
  11) New York City, United States
  12) St. Petersburg, Russia
  13) Milan, Italy
  14) Beijing, People's Republic of China
  15) Istanbul, Turkey
  16) Paris, France
  17) Singapore, Singapore
  18) Dublin, Ireland
  19) Sydney, Australia
  20) Shanghai, People's Republic of China
Input: 2
```

```
182
183   Pick your ending city
184     1) Moscow, Russia
185     2) Seoul, South Korea
186     3) Tokyo, Japan
187     4) Hong Kong, SAR
188     5) London, United Kingdom
189     6) Osaka, Japan
190     7) Geneva, Switzerland
191     8) Copenhagen, Denmark
192     9) Zurich, Switzerland
193     10) Oslo, Norway
194     11) New York City, United States
195     12) St. Petersburg, Russia
196     13) Milan, Italy
197     14) Beijing, People's Republic of China
198     15) Istanbul, Turkey
199     16) Paris, France
200     17) Singapore, Singapore
201     18) Dublin, Ireland
202     19) Sydney, Australia
203     20) Shanghai, People's Republic of China
204   Input: 14
205
206   How many Must-Visit-Cities do you want (1, 2, or 3): 3
207
208   Pick your Must-Visit-City #1
209     1) Moscow, Russia
210     2) Seoul, South Korea
211     3) Tokyo, Japan
212     4) Hong Kong, SAR
213     5) London, United Kingdom
214     6) Osaka, Japan
215     7) Geneva, Switzerland
216     8) Copenhagen, Denmark
217     9) Zurich, Switzerland
218     10) Oslo, Norway
219     11) New York City, United States
220     12) St. Petersburg, Russia
221     13) Milan, Italy
222     14) Beijing, People's Republic of China
223     15) Istanbul, Turkey
224     16) Paris, France
225     17) Singapore, Singapore
226     18) Dublin, Ireland
227     19) Sydney, Australia
228     20) Shanghai, People's Republic of China
229   Input: 1
```

```
230
231   Pick your Must-Visit-City #2
232     1) Moscow, Russia
233     2) Seoul, South Korea
234     3) Tokyo, Japan
235     4) Hong Kong, SAR
236     5) London, United Kingdom
237     6) Osaka, Japan
238     7) Geneva, Switzerland
239     8) Copenhagen, Denmark
240     9) Zurich, Switzerland
241     10) Oslo, Norway
242     11) New York City, United States
243     12) St. Petersburg, Russia
244     13) Milan, Italy
245     14) Beijing, People's Republic of China
246     15) Istanbul, Turkey
247     16) Paris, France
248     17) Singapore, Singapore
249     18) Dublin, Ireland
250     19) Sydney, Australia
251     20) Shanghai, People's Republic of China
252   Input: 5
253
254   Pick your Must-Visit-City #3
255     1) Moscow, Russia
256     2) Seoul, South Korea
257     3) Tokyo, Japan
258     4) Hong Kong, SAR
259     5) London, United Kingdom
260     6) Osaka, Japan
261     7) Geneva, Switzerland
262     8) Copenhagen, Denmark
263     9) Zurich, Switzerland
264     10) Oslo, Norway
265     11) New York City, United States
266     12) St. Petersburg, Russia
267     13) Milan, Italy
268     14) Beijing, People's Republic of China
269     15) Istanbul, Turkey
270     16) Paris, France
271     17) Singapore, Singapore
272     18) Dublin, Ireland
273     19) Sydney, Australia
274     20) Shanghai, People's Republic of China
275   Input: 6
276   --------------------------------------------------------------------------
```

```
277
278  Shortest route from "Seoul, South Korea" to "Beijing, People's Republic of China".
279  Must visit cities:
280   -> Moscow, Russia
281   -> London, United Kingdom
282   -> Osaka, Japan
283  ------------------------------------------------------------------------
284  Seoul, South Korea
285   -> Moscow, Russia
286   -> London, United Kingdom
287   -> Shanghai, People's Republic of China
288   -> Osaka, Japan
289   -> Beijing, People's Republic of China
290
291  Smallest number of connection: 5
292  ------------------------------------------------------------------------
293
294  Input paused, press 'Enter'
295
296  ------------------------------------------------------------------------
297  Select which question you want to do
298   1) Question 1
299   2) Question 2
300   3) Question 3
301   4) Question 4
302   5) Quit
303  Input: 3
304  ------------------------------------------------------------------------
305  Pick your starting city
306    1) Moscow, Russia
307    2) Seoul, South Korea
308    3) Tokyo, Japan
309    4) Hong Kong, SAR
310    5) London, United Kingdom
311    6) Osaka, Japan
312    7) Geneva, Switzerland
313    8) Copenhagen, Denmark
314    9) Zurich, Switzerland
315    10) Oslo, Norway
316    11) New York City, United States
317    12) St. Petersburg, Russia
318    13) Milan, Italy
319    14) Beijing, People's Republic of China
320    15) Istanbul, Turkey
321    16) Paris, France
322    17) Singapore, Singapore
323    18) Dublin, Ireland
324    19) Sydney, Australia
325    20) Shanghai, People's Republic of China
326  Input: 2
327
328  ------------------------------------------------------------------------
```

```
329
330  Adjacent cities to "Seoul, South Korea":
331   -> Moscow, Russia
332   -> Hong Kong, SAR
333   -> Oslo, Norway
334   -> New York City, United States
335  --------------------------------------------------------------------------
336  Shortest route from "Seoul, South Korea" to all adjacent cities and back:
337
338  Seoul, South Korea
339   -> New York City, United States
340   -> Istanbul, Turkey
341   -> Moscow, Russia
342   -> Hong Kong, SAR
343   -> Oslo, Norway
344   -> St. Petersburg, Russia
345   -> Seoul, South Korea
346
347  Smallest number of connection: 7
348  --------------------------------------------------------------------------
349
350  Input paused, press 'Enter'
351
352  --------------------------------------------------------------------------
353  Select which question you want to do
354   1) Question 1
355   2) Question 2
356   3) Question 3
357   4) Question 4
358   5) Quit
359  Input: 4
360  --------------------------------------------------------------------------
361  Pick city for Friend A
362    1) Moscow, Russia
363    2) Seoul, South Korea
364    3) Tokyo, Japan
365    4) Hong Kong, SAR
366    5) London, United Kingdom
367    6) Osaka, Japan
368    7) Geneva, Switzerland
369    8) Copenhagen, Denmark
370    9) Zurich, Switzerland
371    10) Oslo, Norway
372    11) New York City, United States
373    12) St. Petersburg, Russia
374    13) Milan, Italy
375    14) Beijing, People's Republic of China
376    15) Istanbul, Turkey
377    16) Paris, France
378    17) Singapore, Singapore
379    18) Dublin, Ireland
380    19) Sydney, Australia
381    20) Shanghai, People's Republic of China
382  Input: 2
```

```
383
384   Pick city for Friend B
385     1) Moscow, Russia
386     2) Seoul, South Korea
387     3) Tokyo, Japan
388     4) Hong Kong, SAR
389     5) London, United Kingdom
390     6) Osaka, Japan
391     7) Geneva, Switzerland
392     8) Copenhagen, Denmark
393     9) Zurich, Switzerland
394     10) Oslo, Norway
395     11) New York City, United States
396     12) St. Petersburg, Russia
397     13) Milan, Italy
398     14) Beijing, People's Republic of China
399     15) Istanbul, Turkey
400     16) Paris, France
401     17) Singapore, Singapore
402     18) Dublin, Ireland
403     19) Sydney, Australia
404     20) Shanghai, People's Republic of China
405   Input: 14
406
407   Pick city for Friend C
408     1) Moscow, Russia
409     2) Seoul, South Korea
410     3) Tokyo, Japan
411     4) Hong Kong, SAR
412     5) London, United Kingdom
413     6) Osaka, Japan
414     7) Geneva, Switzerland
415     8) Copenhagen, Denmark
416     9) Zurich, Switzerland
417     10) Oslo, Norway
418     11) New York City, United States
419     12) St. Petersburg, Russia
420     13) Milan, Italy
421     14) Beijing, People's Republic of China
422     15) Istanbul, Turkey
423     16) Paris, France
424     17) Singapore, Singapore
425     18) Dublin, Ireland
426     19) Sydney, Australia
427     20) Shanghai, People's Republic of China
428   Input: 20
429
```

```
--------------------------------------------------------------------------------
You three should meet at "Moscow, Russia"

Route for person A
------------------
Seoul, South Korea
 -> Moscow, Russia

Route for person B
------------------
Beijing, People's Republic of China
 -> Sydney, Australia
 -> Hong Kong, SAR
 -> Oslo, Norway
 -> Moscow, Russia

Route for person C
------------------
Shanghai, People's Republic of China
 -> Hong Kong, SAR
 -> Oslo, Norway
 -> Moscow, Russia

Total  number of connection: 8
--------------------------------------------------------------------------------

Input paused, press 'Enter'

--------------------------------------------------------------------------------
Select which question you want to do
 1) Question 1
 2) Question 2
 3) Question 3
 4) Question 4
 5) Quit
Input: 5
]0;js1769@cse01: ~/CSCE3110/project]0;js1769@cse01:~/CSCE3110/project$ exit
exit

Script done on 2022-08-07 10:26:09-0500
```