Kaila Manangan, Eric Hernandez, and Ji Guo

Professor Slott

Computer Science 311

30 November 2022

ABET

Program Design

What algorithm did you choose for implementing the shortest path problem? Why is this algorithm applicable here?

- We chose a hash table because we felt as if it would be easier to access connected elements through the hash table system.

What classes do you use or add? Why do you choose to use those classes?

- We used the hash table and graph class. We used the hash table to store all the cities by city code and graph and Dijkstra's to calculate the shortest distance between two destinations.

System Implementation

What algorithm did you choose for implementing the shortest path problem? Why is this algorithm applicable here?

- We used the Dijkstra's program to calculate the shortest distance, and it is applicable because it compares and calculates the distances from the starting city to its neighbors until it reaches the end.

Describe the details of your implementations.

- We used a hash table to store all of the cities, their neighbor destinations, population, and other information. We also used it to search for specific cities and have their information printed.
- We then used a Dijkstra's Shortest path algorithm with a graph to calculate the shortest distance. We made a locator, curDist, and locator array to calculate the shortest distance. We also made an array of destinations to have those printed out to display the shortest path.

Did you run into problems in your implementation? How did you overcome those problems?

- We ran into the issue of trying to print out the destinations instead of distances. However, we simply made a new dynamic array and reused the index variables we created to print them out.

Results

Did your results match the output in the "sample_results.txt" file?

- Yes

What are your answers to the following questions:

a. The shortest distance and path from FI to GG
    i. The shortest distance from IRWIN to GRPVE is 24
    ii. through the route: IRWIN->PARKER->GRPVE
b. The shortest distance and path from PD to PM
    i. The shortest distance from PARKER to POMONA is 133
    ii. through the route: PARKER->BOSSTOWN->TORRANCE->POMONA
c. The shortest distance and path from PM to PD
    i. The shortest distance from POMONA to PARKER is 357
    ii. through the route: POMONA->EDWIN->ANAHEIM->VICTORVILLE->CHINO->GRPVE->IRWIN->PARKER
d. The Shortest distance and path from SB to PR
    i. The shortest distance from BERNADINO to RIVERA is 152
    ii. through the route: BERNADINO->ISABELLA->BREA->CHINO->RIVERA

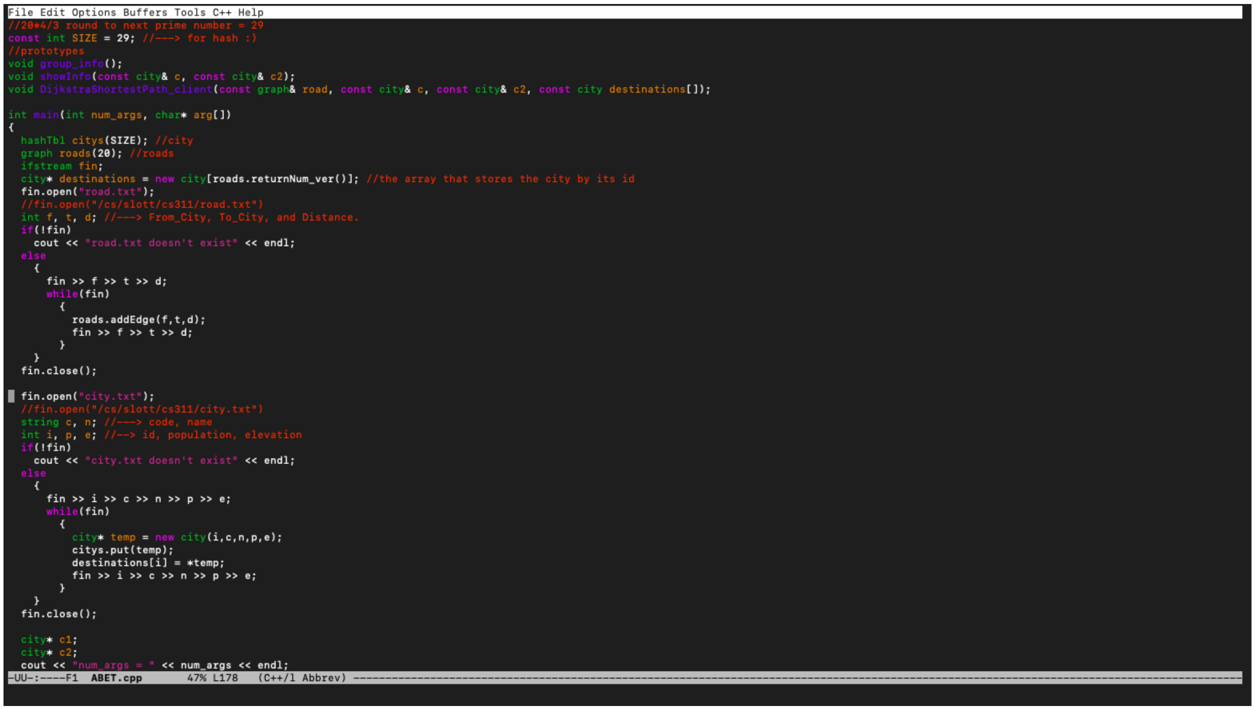
Conclusion

Did your results match the output of the test program? Did you run into problems in your implementation? How did you overcome those problems?

- The results did match the output of the test program. We did run into errors while printing destinations instead of distances, however, by adding a few of our own additional arrays we were able to complete it.

Appendix

You have to provide screenshots of the representative code of your program. You can give 3-4 main function screenshots. The code should not be more than 2 pages.

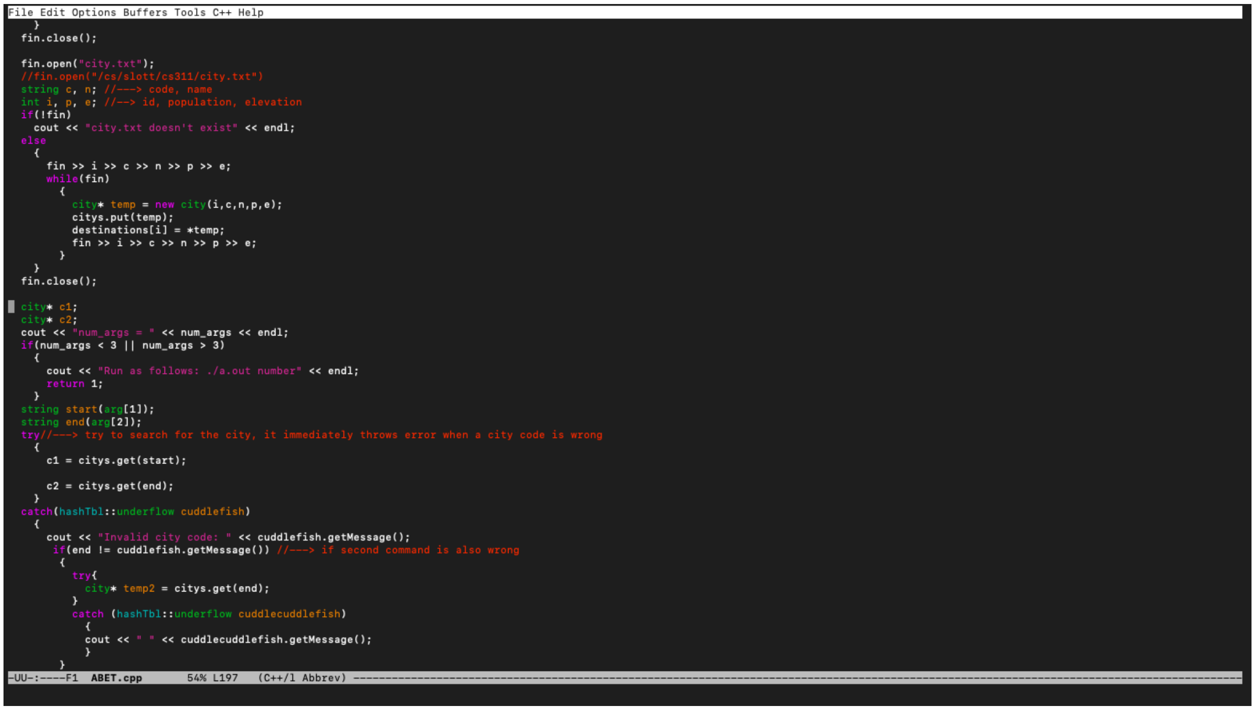


```
//20*4/3 round to next prime number = 29
const int SIZE = 29; //---> for hash :)
//prototypes
void group_info();
void showInfo(const city& c, const city& c2);
void DijkstraShortestPath_client(const graph& road, const city& c, const city& c2, const city destinations[]);

int main(int num_args, char* arg[])
{
  hashTbl citys(SIZE); //city
  graph roads(20); //roads
  ifstream fin;
  city* destinations = new city[roads.returnNum_ver()]; //the array that stores the city by its id
  fin.open("road.txt");
  //fin.open("/cs/slott/cs311/road.txt")
  int f, t, d; //---> From_City, To_City, and Distance.
  if(!fin)
    cout << "road.txt doesn't exist" << endl;
  else
    {
      fin >> f >> t >> d;
      while(fin)
        {
          roads.addEdge(f,t,d);
          fin >> f >> t >> d;
        }
    }
  fin.close();

  fin.open("city.txt");
  //fin.open("/cs/slott/cs311/city.txt")
  string c, n; //---> code, name
  int i, p, e; //--> id, population, elevation
  if(!fin)
    cout << "city.txt doesn't exist" << endl;
  else
    {
      fin >> i >> c >> n >> p >> e;
      while(fin)
        {
          city* temp = new city(i,c,n,p,e);
          citys.put(temp);
          destinations[i] = *temp;
          fin >> i >> c >> n >> p >> e;
        }
    }
  fin.close();

  city* c1;
  city* c2;
  cout << "num_args = " << num_args << endl;
-UU-:----F1  ABET.cpp       47% L178   (C++/l Abbrev) ------------------------------------------------------------------------------------------------
```

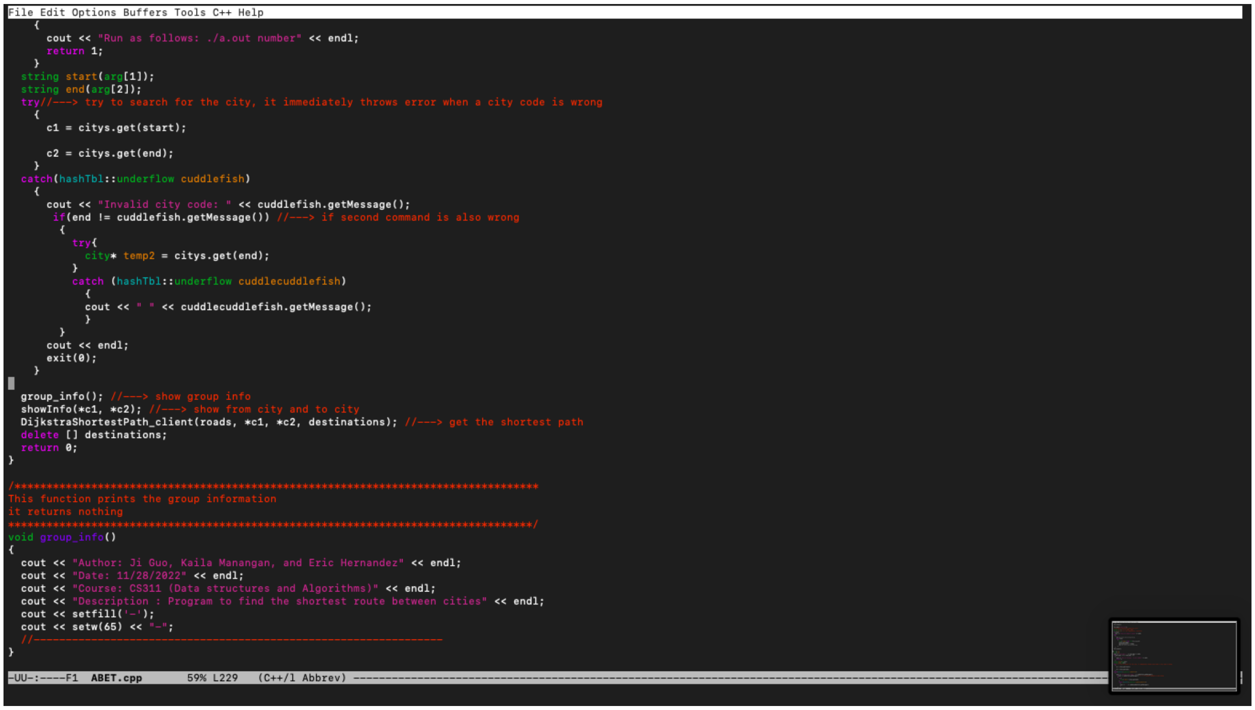


```
    }
  fin.close();

  fin.open("city.txt");
  //fin.open("/cs/slott/cs311/city.txt")
  string c, n; //---> code, name
  int i, p, e; //--> id, population, elevation
  if(!fin)
    cout << "city.txt doesn't exist" << endl;
  else
    {
      fin >> i >> c >> n >> p >> e;
      while(fin)
        {
          city* temp = new city(i,c,n,p,e);
          citys.put(temp);
          destinations[i] = *temp;
          fin >> i >> c >> n >> p >> e;
        }
    }
  fin.close();

  city* c1;
  city* c2;
  cout << "num_args = " << num_args << endl;
  if(num_args < 3 || num_args > 3)
    {
      cout << "Run as follows: ./a.out number" << endl;
      return 1;
    }
  string start(arg[1]);
  string end(arg[2]);
  try//---> try to search for the city, it immediately throws error when a city code is wrong
    {
      c1 = citys.get(start);

      c2 = citys.get(end);
    }
  catch(hashTbl::underflow cuddlefish)
    {
      cout << "Invalid city code: " << cuddlefish.getMessage();
       if(end != cuddlefish.getMessage()) //---> if second command is also wrong
        {
          try{
            city* temp2 = citys.get(end);
          }
          catch (hashTbl::underflow cuddlecuddlefish)
            {
            cout << " " << cuddlecuddlefish.getMessage();
            }
        }
-UU-:----F1  ABET.cpp       54% L197    (C++/l Abbrev) ----------------------------------------
```



```
    {
      cout << "Run as follows: ./a.out number" << endl;
      return 1;
    }
  string start(arg[1]);
  string end(arg[2]);
  try//---> try to search for the city, it immediately throws error when a city code is wrong
    {
      c1 = citys.get(start);

      c2 = citys.get(end);
    }
  catch(hashTbl::underflow cuddlefish)
    {
      cout << "Invalid city code: " << cuddlefish.getMessage();
       if(end != cuddlefish.getMessage()) //---> if second command is also wrong
        {
          try{
            city* temp2 = citys.get(end);
          }
          catch (hashTbl::underflow cuddlecuddlefish)
            {
            cout << " " << cuddlecuddlefish.getMessage();
            }
        }
      cout << endl;
      exit(0);
    }

  group_info(); //---> show group info
  showInfo(*c1, *c2); //---> show from city and to city
  DijkstraShortestPath_client(roads, *c1, *c2, destinations); //---> get the shortest path
  delete [] destinations;
  return 0;
}

/**********************************************************************************
This function prints the group information
it returns nothing
**********************************************************************************/
void group_info()
{
  cout << "Author: Ji Guo, Kaila Manangan, and Eric Hernandez" << endl;
  cout << "Date: 11/28/2022" << endl;
  cout << "Course: CS311 (Data structures and Algorithms)" << endl;
  cout << "Description : Program to find the shortest route between cities" << endl;
  cout << setfill('-');
  cout << setw(65) << "-";
  //---------------------------------------------------------------
}

-UU-:----F1  ABET.cpp       59% L229    (C++/l Abbrev) ----------------------------------------
```