Final Presentation

Jiyang Xu, Jason Li, Jian Song, Alex Li

Goals

- We are hoping to produce a general search tool to find the shortest path between departure and destination airports. When entering two locations, we will use the weighted graph to measure and compare the distance of different airplane routes between them.

- We are hoping to produce a general tool to suggest several travel locations based on a given airport. When entering one locations, we will use the strongest connected components to give suggestions on cities near it that could be visited.

Development : Graph Construction

- Processing data:
  - Read data sets routes.dat and airports.dat
  - Populate a map that contains both routes and airports information
  - Populate a map that stores all the airports id with airports names

- Constructing a graph:
  - Set all edges by using the map obtained from data sets
  - Set all vertices by using the airports obtained above

- Testing our graph:
  - Test graph construction and read files in test cases

Development : BFS

- Two BFS traversals :
  - BFS : traverse the whole graph
  - BFS by destination : traverse the graph based on a given source airport id and a given destination airport id
- Implementation:
  - Data storage : vector, queue
  - Helper function adjacent : get the adjacent airports based on the map obtained above
  - Input : a graph (and source airport id & destination airport id)
  - Output : a vector of strings of traversal results
- Testing:
  - Both BFS traverse the whole graph and traverse by destination is tested by test cases

Development : Dijkstra’s Algorithm

- Two Dijkstra’s functions :
  - Find shortest path based on given source airport id and given destination airport id
  - Find shortest path based on given source airport name and given destination airport name
- Implementation :
  - Use the outline from lectures
  - Helper function to get the distance from the source airport to its adjacent airports
  - Input : a source airport id/name and a destination airport id/name
  - Output : a vector represent the shortest path
- Testing :
  - Two test cases test these two functions

Development : Kosaraju Algorithm

- Dijkstra’s algorithms :
  - Use DFS two times, and transpose the whole graph
- Implementation :
  - Helper functions :
  - 1) DFS : obtain the vertices from a given vertex
  - 2) Fill order : Fills Stack with vertices, the top element of stack has the maximum finishing
  - 3) Transpose : reverse the graph
  - Input : a source airport id
  - Output : a vector of the strongest connected components of the given airport
- Testing :
  - Test case on the main Kosaraju algorithm function

Conclusion

- Achieve all our goals by showing a shortest path between two airports and recommended travel cities as results

- Further goal is to improve these search tools by showing these results to users in a more direct and usable way

- Final and full-scale input dataset we used are data sets airports and routes
- This project reinforced concepts on graph, several graph algorithms and overall coding skills