Open Flights Final Presentation

Ryan Day, Jonathan Yuen, Jack Chen, Kevin Zhou

Goals

- Create a graph data structure that holds the airports and the routes between them
- Implement a BFS traversal
- Implement Dijkstra's algorithm
- Implement the Landmark Path algorithm
- Implement a Page Rank algorithm
- Find a way to represent the data visually
- Leading Question : In this project, we want to explore the OpenFlights dataset to gain insight into how humans can be optimally interconnected via air travel. Construct a route based on the shortest path.

Development Process: Graph Structure

- Gathering and parsing data
  - Downloaded airports.dat and routes.dat from openflights
  - CSV Files, could have invalid data
  - Parsed the data character by character and stored it in vectors of strings
- Creating a graph
  - Inserting all airports
  - Inserting all Flights depending on existing airport and Flights
  - Unordered_map to keep track of adjacency airports/edges
- Testing our graph structure
  - Printing out selected airports and their adjacent airports

Development Process : Traversal

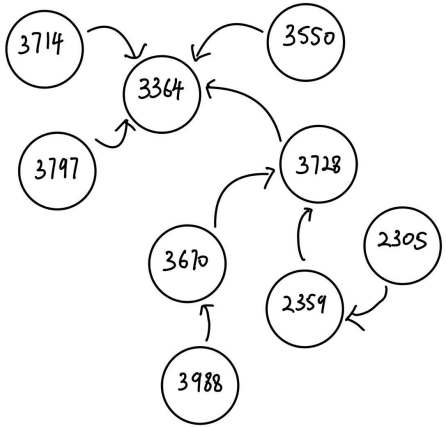
- Overview
  - Three BFS functions using algorithms discussed in lecture
  - Utilized queue
  - Included in Page Rank algorithm
  - Output: Vector of strings of passing airports
- Testing
  - Smaller dataset
  - BFS_moves
  - BFS_dest
- Results
  - Surprisingly fast
  - O(m+n)

```
1  BFS(G):
2    Input: Graph, G
3    Output: A labeling of the edges on
4        G as discovery and cross edges
5
6    foreach (Vertex v : G.vertices()):
7      setLabel(v, UNEXPLORED)
8    foreach (Edge e : G.edges()):
9      setLabel(e, UNEXPLORED)
10   foreach (Vertex v : G.vertices()):
11     if getLabel(v) == UNEXPLORED:
12       BFS(G, v)
```

```
14 BFS(G, v):
15   Queue q
16   setLabel(v, VISITED)
17   q.enqueue(v)
18
19   while !q.empty():
20     v = q.dequeue()
21     foreach (Vertex w : G.adjacent(v)):
22       if getLabel(w) == UNEXPLORED:
23         setLabel(v, w, DISCOVERY)
24         setLabel(w, VISITED)
25         q.enqueue(w)
26       elseif getLabel(v, w) == UNEXPLORED:
27         setLabel(v, w, CROSS)
```

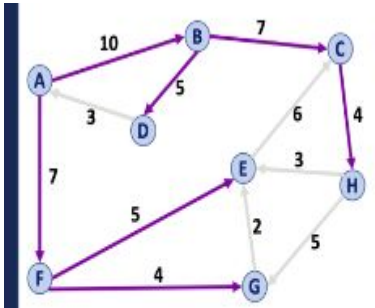
Development Process: Page Rank Algorithm

- Overview
  - The Algorithm:
    - Get initial adjacent matrix by processing a graph object
    - Adjust the matrix with normalization and damping factor
    - Generate a starting vector
    - Basically just doing …* Matrix * Matrix * starting vector
  - Output:
    - Vector that revealing importance of airports
    - Most important airports can be find with a helper function
- Testing
  - Run the algorithm on a small subset of the data
  - Run the algorithm on full set of the data
- Results
  - Confirmed with a few selected data set
  - Running time explodes with normalization



Development Process : Dijkstra's Algorithm

- Overview
  - Used the outline of the algorithm that we covered in lecture
  - Choose a start point and an endpoint to find a path between them
  - Output: a vector of airports along the path; a path length
- Testing
  - Started with paths between neighbors
  - Compared with public flight distances
- Results
  - Confirmed shortest paths
  - O(|E|+|V|log(|V|))




```
6   foreach (Vertex v : G):
7     d[v] = +inf
8     p[v] = NULL
9   d[s] = 0
10
11  PriorityQueue Q // min distance, defined by d[v]
12  Q.buildHeap(G.vertices())
13  Graph T          // "labeled set"
14
15  repeat n times:
16    Vertex u = Q.removeMin()
17    T.add(u)
18    foreach (Vertex v : neighbors of u not in T):
19      if cost(u, v) + d[u] < d[v]:
20        d[v] = cost(u, v) + d[u]
21        p[v] = u
```

Development Process: Landmark Path Algorithm

- Overview
  - Used Dijkstra's to find distance between relevant nodes - now choose multiple landmarks to find path between those
  - Output: a vector of airports along the path; a path length
- Testing
  - Compared with path generated from BFS
- Results
  - Optimal running time of O(|E|+|V|log(|V|))

Overall Results and What is Next

- Successful implementation of our algorithms and traversals
- Promising and interesting results
- Next steps would be an interactive visual aid that shows a user what the shortest flight path or landmark path between two destinations are based on their custom input
- Project reinforced concepts from the semester and helped us learn more about graphs, relevant algorithms, writing tests, etc