Final Project Presentation

Dharmil Jadav, Param Mehta, Kavin Krishnasami, Ayan Deka

Project Goals

- Acquire Data from an online site, specifically with airports and flights
- Create a BFS traversal traversing through all the connecting flights and their paths between two airports.
- Implement Dijkstra’s Algorithm to find the shortest path between two airports, while also using a variable factor to account for airports that may come in the middle.
- Implement Kruskal’s algorithm to find the shortest cycle and path between all the airports

Source of Data

- Got this data from OpenFlights, in which we put download the data file from their website and read from this file to build graph
- Airports.dat, which has an airport ID, name, and longitude and latitude
- Routes.dat, in which we used the route destination ID and the route source ID

Creating the Graph

- Basic Graph implementation, in which it is both directed and weighted
- Vertices are implemented as the airports that we read from the airport.dat file
- Edges are implemented through the routes.dat file that we read from,
- Edges are also directed and weighted to fulfill the basic implementation

BFS

- Idea behind this is to search wide before deep into the graph
- Visits the neighbors of a node first before visiting the neighbors of the neighbors
- Can be used to find shortest path, but not in this case because graph is weighted
- First created a visited map, with the vertex string as the key and a bool value for its value, and initialize all these values as false
- Create a source node and a queue in which we push the source node
- Create a while loop and loop until the queue is empty
  - Pop that node
  - Add to overall list
  - Visit all of that node’s neighbor, and if they are not visited, then mark visit as true and push into queue
  - Print the overall list

Dijkstras

- Accepts a graph, start airport, and end airport and returns the minimum connecting distance between them
- Initialize map of distance and path, and initialize priority queue to iterate through
- Iterate through the current nodes neighbors and add each nodes distance from the source
- Print out the final distance in the distance map

Kruskals

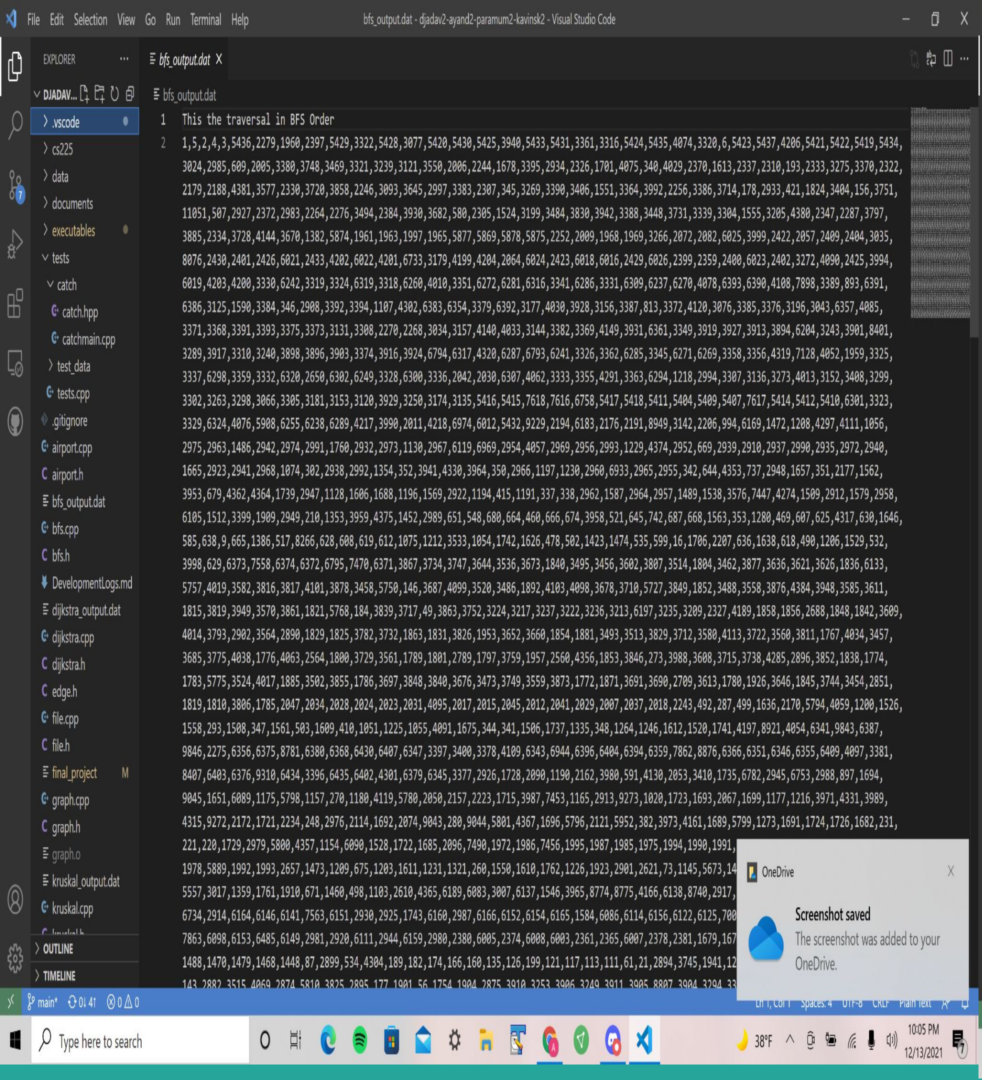
- Kruskals is a type of minimum spanning tree algorithm that inserts the edges from lowest weight to highest weight
- Also prevents a full cycle from being inserted with the edges
- First, get the edges of the graph and sort them
- Associated each edge with a number
- Initialized a disjoint set for each vertex
- If the vertices for each edge are in different sets, then add the weight to the overall result and union these sets together

Testing

- Checked if graph was made correctly first
  - Checked if bad routes were ignored, if the airport constructor was correct implemented, and if distance was calculated correctly
- For BFS, Dijkstras, and Kruskals, we made basic test cases with a graph with generated edges
  - BFS has test cases of the size and the order of the first few objects in the traversal
  - Dijsktras has test cases of calculating the shortest distance between two different nodes in a basic generated graph
  - Kruskals has test cases with the basic graph and tests if it follows the the correct order

Results/Conclusion

- Saved the traversal and algorithm data to their respective .dat files
- With Dijkstras, we were able to find the shortest path between any two given cities, which was interesting to implement when using real data
- Additionally, Kruskals gave us the shortest way to go to every airport.
- While not applicable in real life, it was still cool to see this and how the graph plays out with data
- Struggled in the beginning with building the project with nothing to build off on unlike previous mps, but was still fun to do when finally done with it

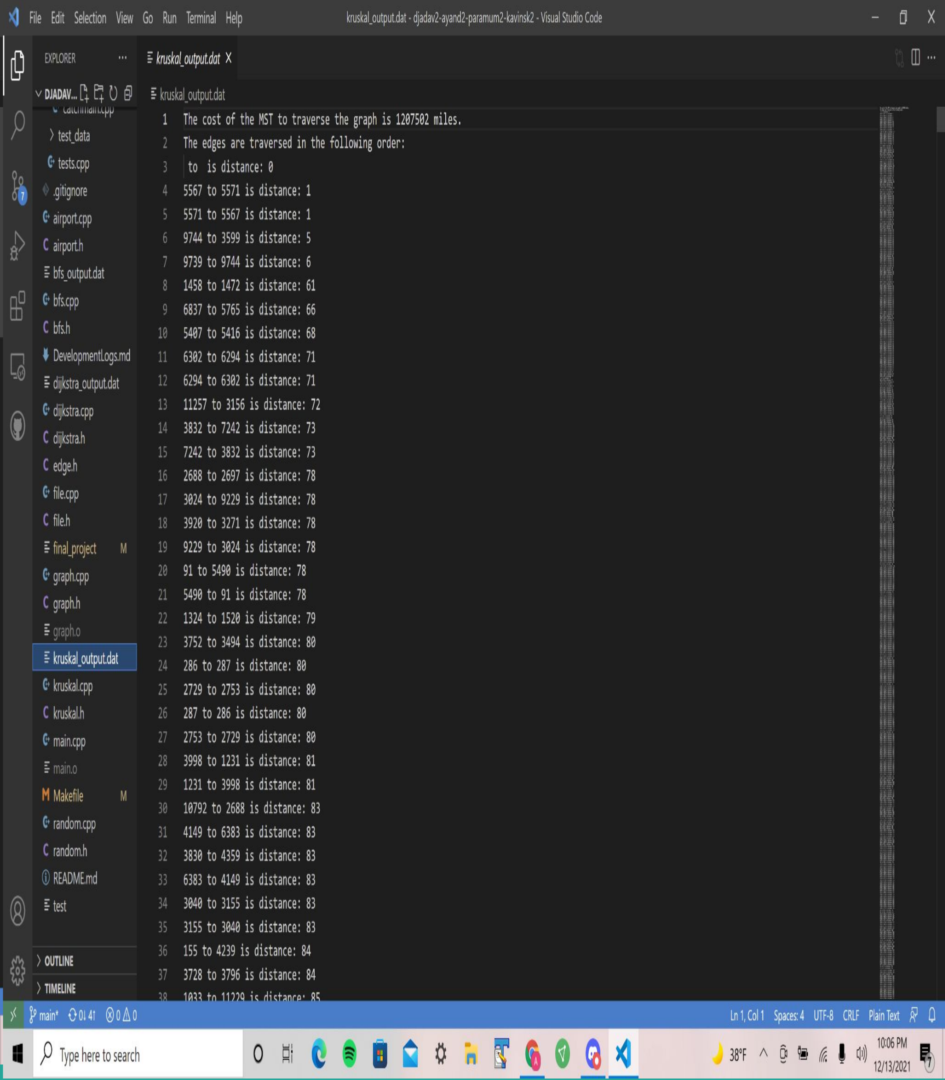


```
This the traversal in BFS Order
1,5,2,4,3,5436,2279,1960,2397,5429,3322,5428,3077,5420,5430,5425,3940,5433,5431,3361,3316,5424,5435,4074,3320,6,5423,5437,4206,5421,5422,5419,5434,
3024,2985,609,2005,3380,3748,3469,3321,3239,3121,3550,2006,2244,1678,3395,2934,2326,1701,4075,340,4029,2370,1613,2337,2310,193,2333,3275,3370,2322,
2179,2188,4381,3577,2330,3720,3858,2246,3093,3645,2997,3383,2307,345,3269,3390,3406,1551,3364,3992,2256,3386,3714,178,2933,421,1824,3404,156,3751,
11051,507,2927,2372,2983,2264,2276,3494,2384,3930,3682,580,2305,1524,3199,3484,3830,3942,3388,3448,3731,3339,3304,1555,3205,4380,2347,2287,3797,
3885,2334,3728,4144,3670,1382,5874,1961,1963,1997,1965,5877,5869,5878,5875,2252,2009,1968,1969,3266,2072,2082,6025,3999,2422,2057,2409,2404,3035,
8076,2430,2401,2426,6021,2433,4202,6022,4201,6733,3179,4199,4204,2064,6024,2423,6018,6016,2429,6026,2399,2359,2400,6023,2402,3272,4090,2425,3994,
6019,4203,4200,3330,6242,3319,3324,6319,3318,6260,4010,3351,6272,6281,6316,3341,6286,3331,6309,6237,6270,4078,6393,6390,4108,7898,3389,893,6391,
6386,3125,1590,3384,346,2908,3392,3394,1107,4302,6383,6354,3379,6392,3177,4030,3928,3156,3387,813,3372,4120,3076,3385,3376,3196,3043,6357,4085,
3371,3368,3391,3393,3375,3373,3131,3308,2270,2268,3034,3157,4140,4033,3144,3382,3369,4149,3931,6361,3349,3919,3927,3913,3894,6204,3243,3901,8401,
3289,3917,3310,3240,3898,3896,3903,3374,3916,3924,6794,6317,4320,6287,6793,6241,3326,3362,6285,3345,6271,6269,3358,3356,4319,7128,4052,1959,3325,
3337,6298,3359,3332,6320,2650,6302,6249,3328,6300,3336,2042,2030,6307,4062,3333,3355,4291,3363,6294,1218,2994,3307,3136,3273,4013,3152,3408,3299,
3302,3263,3298,3066,3305,3181,3153,3120,3929,3250,3174,3135,5416,5415,7618,7616,6758,5417,5418,5411,5404,5409,5407,7617,5414,5412,5410,6301,3323,
3329,6324,4076,5908,6255,6238,6289,4217,3990,2011,4218,6974,6012,5432,9229,2194,6183,2176,2191,8949,3142,2206,994,6169,1472,1208,4297,4111,1056,
2975,2963,1486,2942,2974,2991,1760,2932,2973,1130,2967,6119,6969,2954,4057,2969,2956,2993,1229,4374,2952,669,2939,2910,2937,2990,2935,2972,2940,
1665,2923,2941,2968,1074,302,2938,2992,1354,352,3941,4330,3964,350,2966,1197,1230,2960,6933,2965,2955,342,644,4353,737,2948,1657,351,2177,1562,
3953,679,4362,4364,1739,2947,1128,1606,1688,1196,1569,2922,1194,415,1191,337,338,2962,1587,2964,2957,1489,1538,3576,7447,4274,1509,2912,1579,2958,
6105,1512,3399,1909,2949,210,1353,3959,4375,1452,2989,651,548,680,664,460,666,674,3958,521,645,742,687,668,1563,353,1280,469,607,625,4317,630,1646,
585,638,9,665,1386,517,8266,628,608,619,612,1075,1212,3533,1054,1742,1626,478,502,1423,1474,535,599,16,1706,2207,636,1638,618,490,1206,1529,532,
3998,629,6373,7558,6374,6372,6795,7470,6371,3867,3734,3747,3644,3536,3673,1840,3495,3456,3602,3807,3514,1804,3462,3877,3636,3621,3626,1836,6133,
5757,4019,3582,3816,3817,4101,3878,3458,5750,146,3687,4099,3520,3486,1892,4103,4098,3678,3710,5727,3849,1852,3488,3558,3876,4384,3948,3585,3611,
1815,3819,3949,3570,3861,1821,5768,184,3839,3717,49,3863,3752,3224,3217,3237,3222,3236,3213,6197,3235,3209,2327,4189,1858,1856,2688,1848,1842,3609,
4014,3793,2902,3564,2890,1829,1825,3782,3732,1863,1831,3826,1953,3652,3660,1854,1881,3493,3513,3829,3712,3580,4113,3722,3560,3811,1767,4034,3457,
3685,3775,4038,1776,4063,2564,1800,3729,3561,1789,1801,2789,1797,3759,1957,2560,4356,1853,3846,273,3988,3608,3715,3738,4285,2896,3852,1838,1774,
1783,5775,3524,4017,1885,3502,3855,1786,3697,3848,3840,3676,3473,3749,3559,3873,1772,1871,3691,3690,2709,3613,1780,1926,3646,1845,3744,3454,2851,
1819,1810,3806,1785,2047,2034,2028,2024,2023,2031,4095,2017,2015,2045,2012,2041,2029,2007,2037,2018,2243,492,287,499,1636,2170,5794,4059,1200,1526,
1558,293,1508,347,1561,503,1609,410,1051,1225,1055,4091,1675,344,341,1506,1737,1335,348,1264,1246,1612,1520,1741,4197,8921,4054,6341,9843,6387,
9846,2275,6356,6375,8781,6380,6368,6430,6407,6347,3397,3400,3378,4109,6343,6944,6396,6404,6394,6359,7862,8876,6366,6351,6346,6355,6409,4097,3381,
8407,6403,6376,9310,6434,3396,6435,6402,4301,6379,6345,3377,2926,1728,2090,1190,2162,3980,591,4130,2053,3410,1735,6782,2945,6753,2988,897,1694,
9045,1651,6089,1175,5798,1157,270,1180,4119,5780,2050,2157,2223,1715,3987,7453,1165,2913,9273,1020,1723,1693,2067,1699,1177,1216,3971,4331,3989,
4315,9272,2172,1721,2234,248,2976,2114,1692,2074,9043,280,9044,5801,4367,1696,5796,2121,5952,382,3973,4161,1689,5799,1273,1691,1724,1726,1682,231,
221,220,1729,2979,5800,4357,1154,6090,1528,1722,1685,2096,7490,1972,1986,7456,1995,1987,1985,1975,1994,1990,1991,
1978,5889,1992,1993,2657,1473,1209,675,1203,1611,1231,1321,260,1550,1610,1762,1226,1923,2901,2621,73,1145,5673,14
5557,3017,1359,1761,1910,671,1460,498,1103,2610,4365,6189,6083,3007,6137,1546,3965,8774,8775,4166,6138,8740,2917,
6734,2914,6164,6146,6141,7563,6151,2930,2925,1743,6160,2987,6166,6152,6154,6165,1584,6086,6114,6156,6122,6125,700
7863,6098,6153,6485,6149,2981,2920,6111,2944,6159,2980,2380,6005,2374,6008,6003,2361,2365,6007,2378,2381,1679,167
1488,1470,1479,1468,1448,87,2899,534,4304,189,182,174,166,160,135,126,199,121,117,113,111,61,21,2894,3745,1941,12
143,2882,3515,4069,2874,5810,3825,2895,177,1901,56,1754,1904,2875,3910,3253,3906,3249,3911,3905,8807,3904,3294,33
```

OneDrive

Screenshot saved

The screenshot was added to your OneDrive.



```
The cost of the MST to traverse the graph is 1207502 miles.
The edges are traversed in the following order:
 to  is distance: 0
5567 to 5571 is distance: 1
5571 to 5567 is distance: 1
9744 to 3599 is distance: 5
9739 to 9744 is distance: 6
1458 to 1472 is distance: 61
6837 to 5765 is distance: 66
5407 to 5416 is distance: 68
6302 to 6294 is distance: 71
6294 to 6302 is distance: 71
11257 to 3156 is distance: 72
3832 to 7242 is distance: 73
7242 to 3832 is distance: 73
2688 to 2697 is distance: 78
3024 to 9229 is distance: 78
3920 to 3271 is distance: 78
9229 to 3024 is distance: 78
91 to 5490 is distance: 78
5490 to 91 is distance: 78
1324 to 1520 is distance: 79
3752 to 3494 is distance: 80
286 to 287 is distance: 80
2729 to 2753 is distance: 80
287 to 286 is distance: 80
2753 to 2729 is distance: 80
3998 to 1231 is distance: 81
1231 to 3998 is distance: 81
10792 to 2688 is distance: 83
4149 to 6383 is distance: 83
3830 to 4359 is distance: 83
6383 to 4149 is distance: 83
3040 to 3155 is distance: 83
3155 to 3040 is distance: 83
155 to 4239 is distance: 84
3728 to 3796 is distance: 84
1033 to 11229 is distance: 85
```

Future Development

- For now, we just added the base features to the project
- Includes the traversal and two algorithms for finding the shortest path and shortest time to pass through all of the airports
- This are just the base features, and we can add extra features to the project
- For example, a variable factor that takes into account which airport is closer to the starting airport if we have one connecting flight