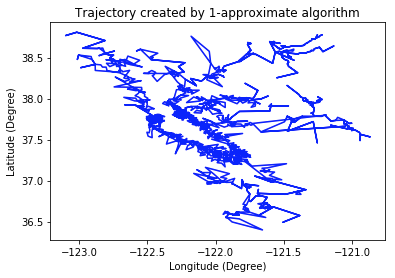
**Question 9 (continued)**

The 1-approximate algorithm’s main steps are duplicate edges, find the Eulerian path, shortcutting. All the edges are duplicated, and then find the Eulerian path, to get the shortest Eulerian spanning tree, try to extract the embedding path (to remove duplicate vertices in the Eulerian path). For example, 1-3-2-3-4-5-4-3-1, will first become 1-3-2- -4-5-4-3-1, if edge (2,4) exists in the graph (it is real edge), if not find the minimum path between these two nodes.

This is based on the results from Question 8, from Question 8 we can see that 92.6% satisfy the triangle inequality, which means about there is about 92.6% this triangle relation in triangle (2,3,4) satisfies. We could just assume the triangle inequality satisfies during the shortcutting. Through shortcutting we could find the shortest Eulerian path.

Because the Eulerian path is duplicated from the **duplicated** minimum spanning tree, the worst cost of the Eulerian path is the **twice** of the cost of the minimum spanning tree , after the shortcutting the cost will be lower than that so , therefore, the calculated upper bound about **1.74** makes sense. And because it is a 1-approximate method.

**Question 10**

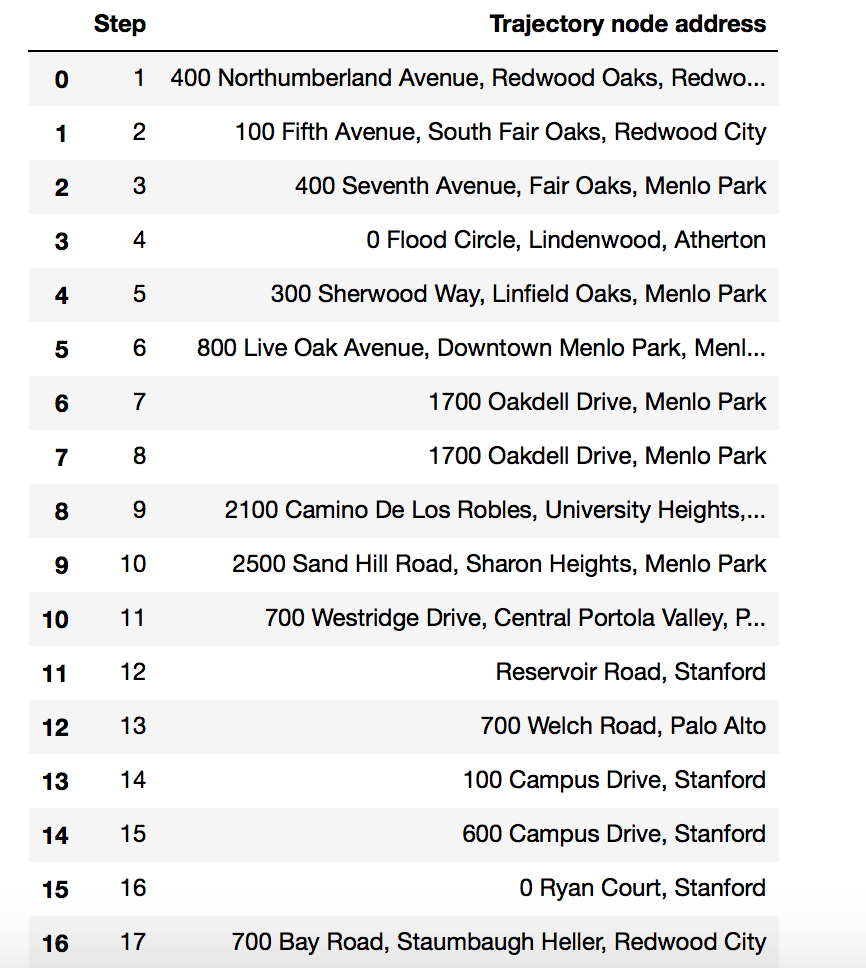


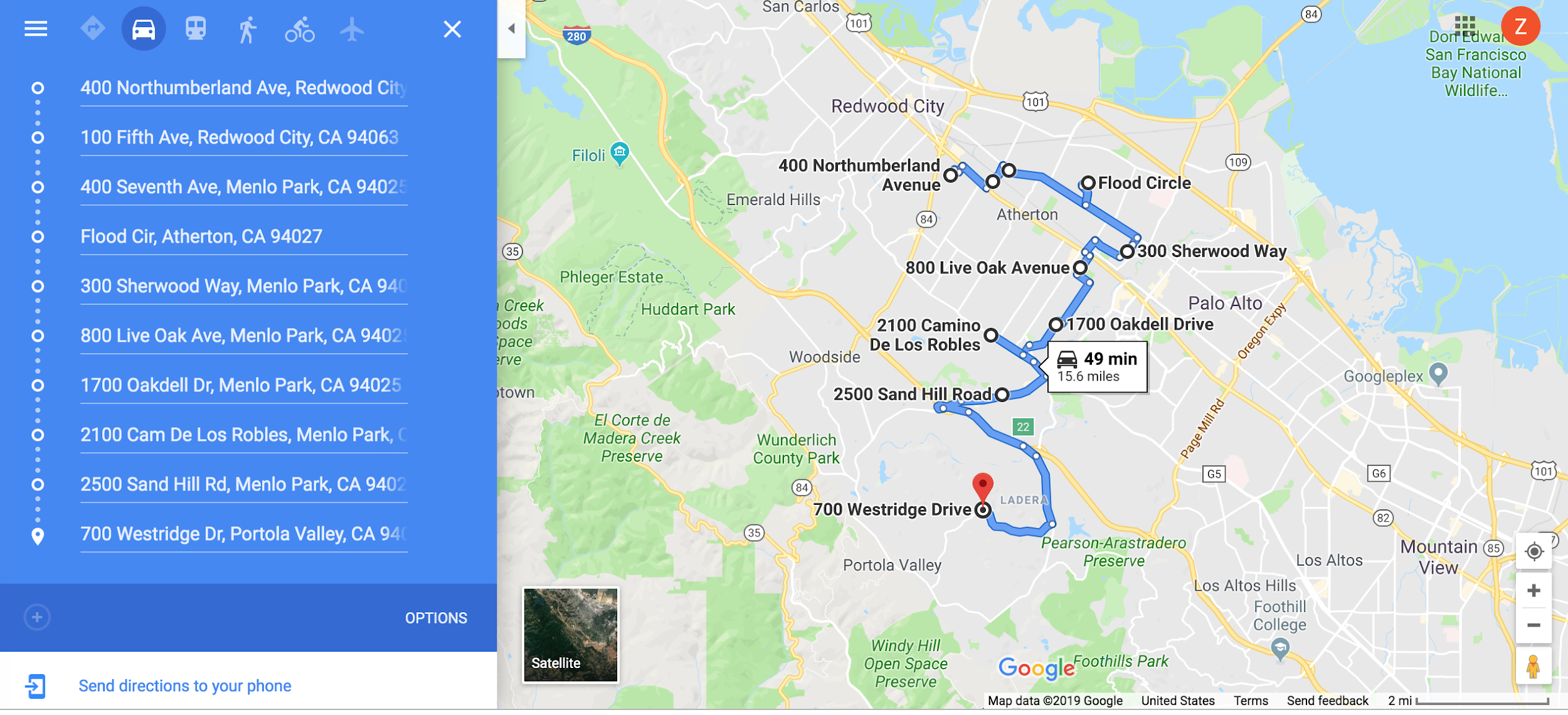
| 1-Approximate Algorithm | Minimum Spanning Tree |
| --- | --- |
|  |  |

The figure above compare the trajectory plot of Santa has to travel generated by the 1-approximate algorithm and the minimum spanning tree in igraph package. They look very similar, which means that the 1-approximate algorithm works very well.

The minimum spanning tree generated by the igraph package uses the Prim’s algorithm, which is a kind of greedy algorithm. That is “We start from one vertex and keep adding edges with the lowest weight until we reach our goal”.

The dataframe below is the initial part of the address sequence produced by 1-approximate algorithm. And the following is the first 10 street addresses in the sequence showed onto the real map. The results make sense, because the adjacent nodes in the sequence are close to each other.





**Question 11**

| Road mesh | Zoom-in road mesh |
| --- | --- |
|  |  |

The figures above are the road mesh created by Delaunay triangulation. After Delaunay triangulation all the nodes are connected, and the Delaunay algorithm maximize the minimum angles of all the triangles. For each triangle its circumcircle contains no other nodes, and for each edge a circle exists through its endpoints contains no other nodes. The Euclidean minimum spanning tree (the Euclidean minimum spanning tree is the minimum spanning tree based on the Euclidean distance of the edge in space instead of the weight of the edge) is a subset of the Delaunay triangulation of the same set of nodes.

**Question 12**

, is cars/(road.hour), is cars/mile, is mile/hour

Safety distance:

Distance between two cars:

Density:

Flow:

Each road has 2 lanes, the flow is:

|  |
| --- |

To calculate the flow of each road the edge of the graph, the edge distance is calculated by the Euclidean distance between the coordinates and then convert the degree to mile, using average 69 mile per degree. The time is the mean travel time of the edge. So Then use the equation derived above to calculate the flow of each edge. The average flow of the graph is about **2996 cars/(road.hour).** Below is part of the flow values of the edges:

| 3281 2706 3157 3043 3312 2969 3040 3327 2858 3243 3266 2494  3024 3042 3107 3030 3202 3229 2846 3160 2447 3031 3234 2765  3277 3163 2646 3169 3270 3262 2951 3069 3193 2951 2887 2920  3137 3179 3234 3074 3148 3147 2692 3210 3317 2950 3169 3296  2949 2913 3163 2717 3254 3302 2499 3172 2889 2744 3082 3231  3236 3005 2771 3066 2757 2806 3141 2980 3316 3238 3151 3183  3286 2816 3054 3294 3171 3242 2753 2671 3240 3081 2827 2554  2581 3299 3226 2922 3098 3131 3119 2497 2957 3256 3156 2822  2677 3272 2878 2674 2785 3309 3126 3031 3099 3137 3160 3018  2902 2969 3286 3159 3148 2941 2700 3165 3096 2827 2500 3276  2914 3117 2927 3206 3078 3169 3208 2791 3017 3186 3230 3301  2167 2892 2788 3184 3158 2521 2918 3140 2985 3196 3254 3223  2860 2600 2749 3148 2199 2617 2418 3189 3055 2924 2946 3195  3330 2886 3016 2754 3321 3127 3067 3052 3107 3107 3085 3034  2715 2754 2715 2757 2851 3112 2674 2699 2641 3088 2770 2658  3242 3024 3150 3406 3189 3101 3167 2990 3313 3075 2707 2621  2440 3263 2981 3060 2751 3100 3144 2898 3046 3100 3191 2869  3036 2919 3250 3046 3257 3281 2974 2364 2327 2750 3152 3155  2772 3023 2936 2660 3093 3310 2733 2765 3095 3076 3145 2881  … ... |
| --- |

**Question 13**

Use the maxflow algorithm Ford-Fulkerson algorithm to calculate the maxflow of the graph based on the flow of each edge calculated in Question 12. The maxflow value is **470908 cars/(road.hour)**. It means that in the ideal case (no traffic jam), the maximum total number of cars that can commute on the road travelling from “100 Campus Drive, Stanford” to “700 Meder Street, Santa Cruz” is 470908 per hour.

The number of edge-disjoint paths equals the number of edges that have to be removed in order to disconnect the two vertices. The number of edge-disjoint paths between the source and the destination is **6**.

**Edge-disjoint paths are the maximum number of paths that don’t have common edges. The number of edge-disjoint paths problem is one of the variants of the maxflow problem.** In the edge-disjoint paths problem, each edge has a capacity of 1. And the maxflow between the source and the target nodes in this graph is k if and only if the number of edge-disjoint paths is k.

The results make sense, because from the zoom-in road mesh of source and the destination we can see that there are 6 neighbor nodes directly connected to Stanford node, but there are only 5 neighbor nodes directly connected to UCSC node. So the maximum number of paths that don’t have common edges should be a number that is <= 6.

| Stanford neighbourhood (-122.18, 37.43)  ID in Geo data = 2607 | UCSC neighbourhood (-122.06, 36.97)  ID in Geo data = 1968 |
| --- | --- |
|  |  |

**Question 14**

| Pruned graph | Original Delaunay graph |
| --- | --- |
|  |  |

Delaunay triangulation might generate some edges that are not in the graph. To prune the graph:

(1) Check all the edges of all the triangles generated by Delaunay to see if the edge is in the edgelist of the graph;

(2) If the edge is in the edgelist of the graph, then check if the edge mean\_travel\_time is lower than the set threshold 600;

(3) Remove the edge if it is not in the edgelist of the graph, or if the edge mean\_travel\_time is lower than the set threshold.

For the pruned graph now, all the edges are “real road” that is they are in the edgelist of the graph, also all of their mean\_travel\_time is within 600 seconds (10 min).

As for the bridges, from the pruned graph, we can see that all the bridges except for the “San Mateo Bridge” are treated as fake bridge and removed, because it is the longest bridge of the five bridges listed in the question leading to longer mean travel time.

**Question 15**

We repeat Question 13 with the pruned graph produced in Question 14, that is (1) calculate the flow of each edge of the pruned graph, (2) get the maxflow and the number of edge-disjoint paths between Stanford to UCSC.

The maxflow value of the pruned graph is **21363 cars/(road.hour)**. It means that in the ideal case (no traffic jam), the maximum total number of cars that can commute on the road travelling from “100 Campus Drive, Stanford” to “700 Meder Street, Santa Cruz” is 21363 per hour. And the number of edge-disjoint path remains **6**.

The maxflow value of the pruned graph is lower because the threshold is relatively low, leading to a lot of road are treated as “fake”, according to the Ford-Fulkerson algorithm for the calculation of the maxflow, since there are less road that could be used to travel from Stanford to UCSC, the number of cars that can commute between these two nodes will decrease.

From the zoom-in road mesh of source and the destination we can see that there are 6 neighbor nodes directly connected to Stanford node, but there are only 4 neighbor nodes directly connected to UCSC node. The decreased number of neighbor nodes is due to the edge trimming process. Edge-disjoint paths are the maximum number of paths that don’t have common edges. The results make sense, because from the zoom-in road mesh of source and the destination we can see that there are 6 neighbor nodes directly connected to Stanford node, but there are only 4 neighbor nodes directly connected to UCSC node. So the maximum number of paths that don’t have common edges should be a number that is <= 6.

| Stanford neighbourhood (-122.18, 37.43)  ID in Geo data = 2607 | UCSC neighbourhood (-122.06, 36.97)  ID in Geo data = 1968 |
| --- | --- |
|  |  |