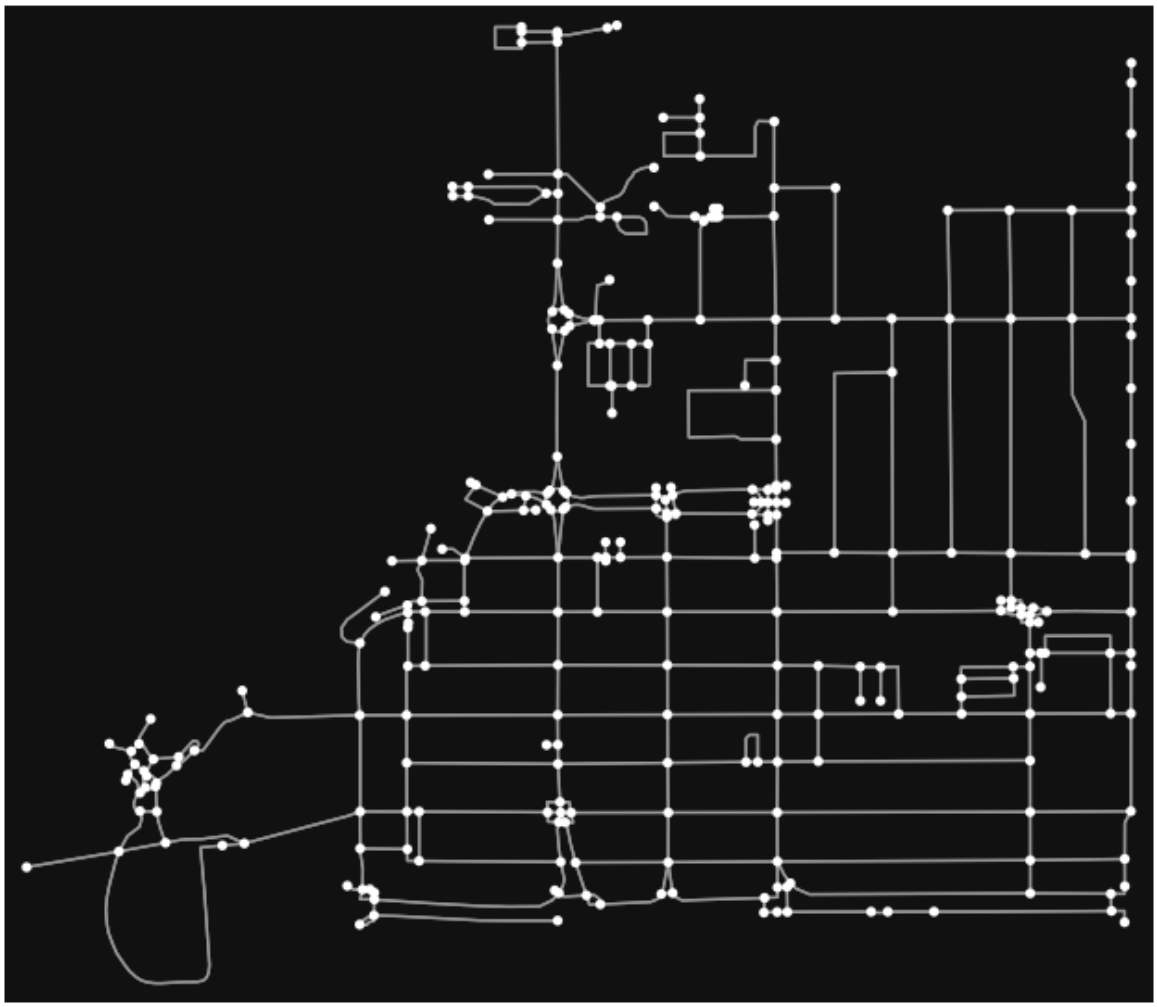
In [5]:
```
import osmnx as ox
import networkx as nx
import plotly.graph_objects as go
import numpy as np
```

In [2]:
```
H = ox.graph_from_place("Kelowna, Canada", network_type="drive")
fig, ax = ox.plot_graph(H)
```

In [6]:
```
# Defining the map boundaries
#north, east, south, west = 49.9049238, -119.4543568, 49.8461362, -119.
north, east, south, west = 49.8916962, -119.4884523, 49.8834416, -119.5
# Downloading the map as a graph object
kelowna_G = ox.graph_from_bbox(north, south, east, west, network_type =
# Plotting the map graph
ox.plot_graph(kelowna_G)
```



Out[6]: (<Figure size 800x800 with 1 Axes>, <AxesSubplot:>)

In [7]:
```
kelowna_G.number_of_nodes(),kelowna_G.number_of_edges()
```

Out[7]: (302, 850)

In [14]:
```
edges = ox.graph_to_gdfs(kelowna_G, nodes=False, edges=True)
# edges
# edges_series = edges['length'] #gives you a pandas series with edge l
edges[['osmid','length']] #gives you a pandas dataframe with osmids of
```

Out[14]:

			osmid	length
u	**v**	**key**		
355377664	**2742445757**	**0**	31754193	60.051
	355377951	**0**	100577419	73.357
	2367768318	**0**	[128070577, 969983956]	87.515
355377709	**4757605586**	**0**	846929817	54.433
	8044344509	**0**	919464719	54.577
...	**...**	**...**	...	...
9884241022	**8072927121**	**0**	1078096002	8.293
	4003592525	**0**	1078096003	12.265
10295505569	**1173428791**	**0**	31754226	183.617
	355378187	**0**	31754226	54.454
	1173428778	**0**	676314907	240.817

850 rows × 2 columns

In [15]:
```
length_weight = list(edges['length'])
len(length_weight)
```

Out[15]: 850

In [16]:
```
edges_list = list(kelowna_G.edges())

nodes_list= []
for edge in edges_list:
    x,y = edge[0],edge[1]
    nodes_list.append(x)
    nodes_list.append(y)

len(nodes_list)
```

Out[16]: 1700

In [17]:
```
nodesNumber = kelowna_G.number_of_nodes()

nodes_set = set(nodes_list)
reverse_nodes_dict = dict(enumerate(nodes_set))
nodes_dict = dict(zip(nodes_set,range(0,nodesNumber)))
# reverse_nodes_dict
nodes_dict
```

Out[17]:
```
{355377664: 0,
 8065990145: 1,
 8065990146: 2,
 8065990147: 3,
 8065990148: 4,
 8065990149: 5,
 355378182: 6,
 8065990150: 7,
 2727714822: 8,
 2727714825: 9,
 8065990151: 10,
 355378187: 11,
 8976144907: 12,
 8976144908: 13,
 2727714832: 14,
 2727714834: 15,
 2727714837: 16,
 356386839: 17,
 2727714840: 18,
```

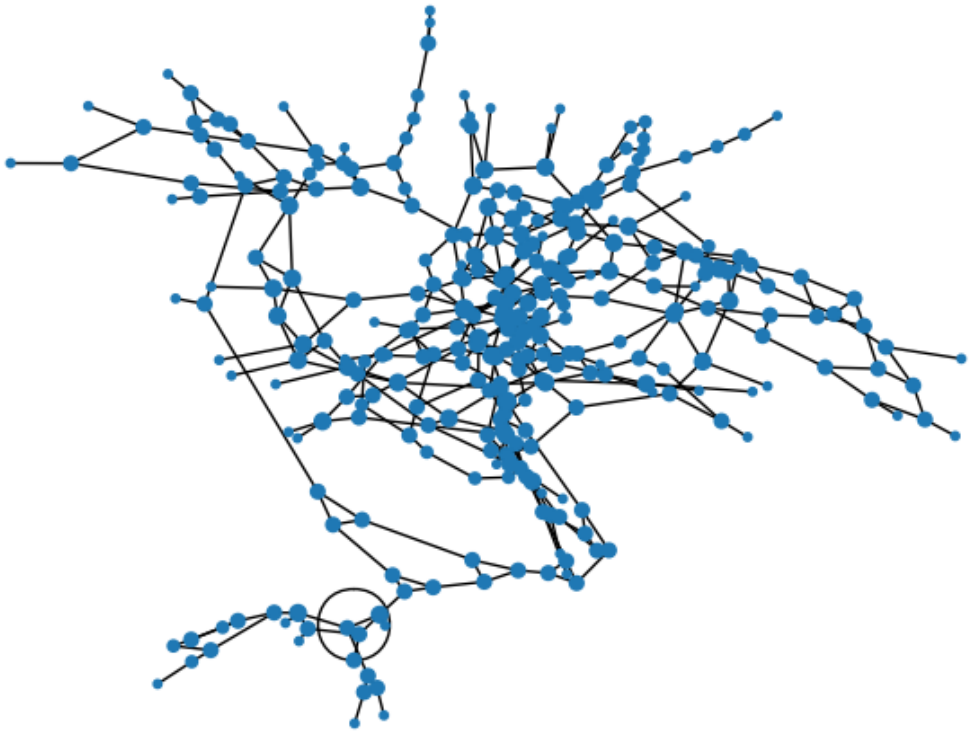
In [18]:
```
new_edges_list = []
for edge in edges_list:
    x,y= edge[0],edge[1]
    newX,newY= nodes_dict[x],nodes_dict[y]
    new_edge = (newX, newY)
    new_edges_list.append(new_edge)
new_edges_list
```

Out[18]:
```
[(0, 131),
 (0, 201),
 (0, 183),
 (37, 151),
 (37, 130),
 (37, 6),
 (37, 280),
 (39, 143),
 (39, 285),
 (39, 118),
 (195, 214),
 (195, 301),
 (195, 64),
 (197, 301),
 (197, 226),
 (197, 31),
 (197, 150),
 (201, 166),
 (201, 0),
```

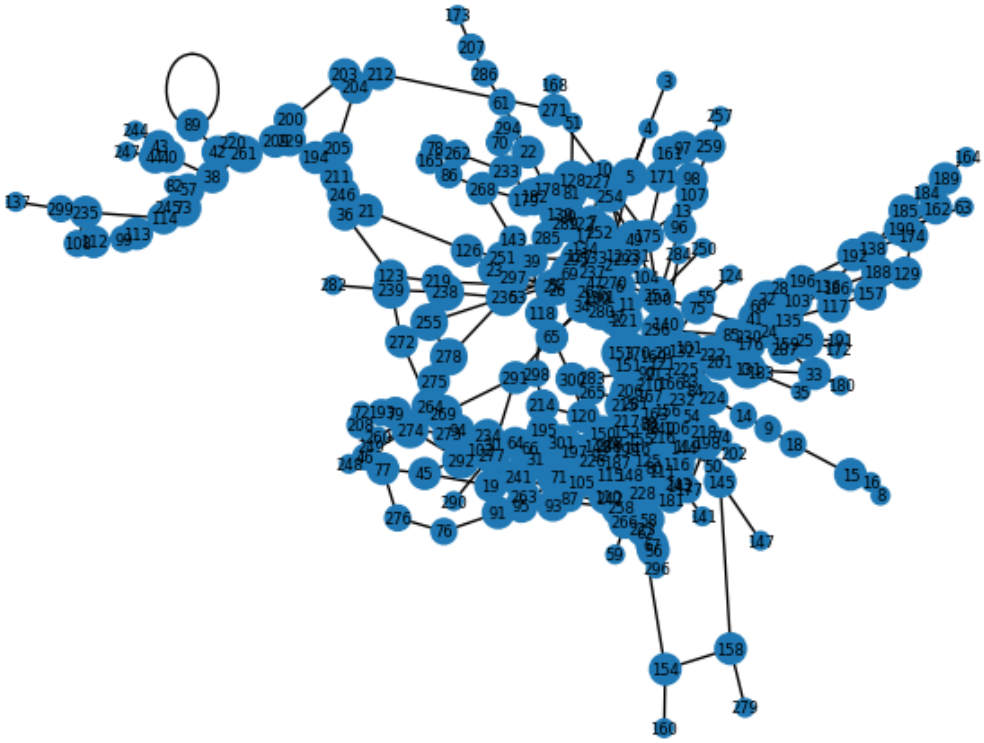
In [19]:
```
weight_length = list(edges['length'])
new_edges_list_weight = [list(i) for i in new_edges_list]
for x in range(0,len(new_edges_list_weight)):
    new_edges_list_weight[x].append(length_weight[x])
new_edges_list_with_weight = [tuple(ele) for ele in new_edges_list_weig
new_edges_list_with_weight
```

Out[19]:
```
[(0, 131, 60.051),
 (0, 201, 73.357),
 (0, 183, 87.51499999999999),
 (37, 151, 54.433),
 (37, 130, 54.577),
 (37, 6, 117.62499999999999),
 (37, 280, 111.378),
 (39, 143, 54.497),
 (39, 285, 85.385),
 (39, 118, 41.89),
 (195, 214, 99.275),
 (195, 301, 102.17099999999999),
 (195, 64, 49.937),
 (197, 301, 256.796),
 (197, 226, 110.572),
 (197, 31, 49.727),
 (197, 150, 51.474000000000004),
 (201, 166, 47.516),
 (201, 0, 73.357),
```

In [20]:
```
G = nx.Graph()
G.add_weighted_edges_from(new_edges_list_with_weight)
d = dict(G.degree)
d.keys()
# nx.draw(G, with_labels=True, node_size=d)
# nx.draw(G)
# nx.draw(G, with_labels=True, nodelist=list(d.keys()), node_size=[v *
nx.draw(G, nodelist=list(d.keys()), node_size=[v * 10 for v in d.values
```



In [21]:
```
nx.draw(G, with_labels=True,font_size=6,nodelist=list(d.keys()), node_s
```



In [22]:
```
from search_algs import dijkstra
```

In [23]:
```
# define origin and desination locations
# origin_point = (49.9049238, -119.4543568)
# destination_point = (49.8461362, -119.5047125)
origin_point = (49.883911, -119.5018168)
destination_point = (49.8871646, -119.4945047)
# get the nearest nodes to the locations
origin_node = ox.distance.nearest_nodes(kelowna_G, origin_point[1], ori
destination_node = ox.distance.nearest_nodes(kelowna_G, destination_poi
# printing the closest node id to origin and destination points
print(origin_node, destination_node)
print(nodes_dict[origin_node], nodes_dict[destination_node])
```

```
1182749233 1162896325
41 278
```

In [24]:
```
labels = {}
for node in G.nodes():
    if node==nodes_dict[origin_node]:
        #set the node name as the key and the label as its value
        labels[node] = 's'
    if node==nodes_dict[destination_node]:
        labels[node] = 'e'
#set the argument 'with labels' to False so you have unlabeled graph
nx.draw(G, labels=labels, font_size=8, font_color='r', node_size=35)
```

In [25]:
```
start_node = nodes_dict[origin_node]
end_node = nodes_dict[destination_node]
trace = dijkstra(G, start_node, end_node)
trace
```

Out[25]:
```
([41,
  103,
  186,
  157,
  117,
  135,
  176,
  201,
  0,
  131,
  33,
  287,
  230,
  24,
  270,
  237,
  295,
  251,
  297,
  23,
  26,
  27,
  219,
  238,
  255,
  278],
 800.5190000000001)
```

In [26]:
```
from plot_path import plot_path
```

In [27]:
```
steps = trace[0]
path = [reverse_nodes_dict[r] for r in steps]
long = []
lat = []
for i in path:
    point = kelowna_G.nodes[i]
    long.append(point['x'])
    lat.append(point['y'])
```

In [28]:
```
plot_path(lat, long, origin_point, destination_point)
```

In [ ]:
```

```