Joint segment

Joint segment

```
def Joint_segment(road_network, Segment_list):#road_network는 self
    Joint_node = Segment_list[0].nodes()
    for seg in Segment_list[1:]:
        start_overlap = np.where(Joint_node == seg.nodes()[0])[0]
# if segments are overlap
        if len(start_overlap)>0:
            Joint_node = np.r_[Joint_node[:start_overlap[0]], seg.nodes()]
# if segments are not overlap
        else:
            shortest_path = nx.shortest_path(road_network, Joint_node[-1], seg.nodes()[0], 'length')#length를 weight로
            shortest_path_array = np.zeros([len(shortest_path)])
            for i in range(len(shortest_path)): shortest_path_array[i]=shortest_path[i]
            Joint_node = np.r_[Joint_node, shortest_path_array[1:], seg.nodes()[1:]]
# node array -> segment form
    edge_in = (Joint_node[0],Joint_node[1],0,road_network.get_edge_data(Joint_node[0],Joint_node[1],0))
    Jointsegment = segment(edge_in)
    for edge_count in range(len(Joint_node)-2):
        edge_in = (Joint_node[edge_count+1],Joint_node[edge_count+2],0,road_network.get_edge_data(Joint_node[edge_count+1],Joint_node[edge_count+2],0))
        Jointsegment = Jointsegment.expand(edge_in)
    return Jointsegment
```

20200702

Joint segment

no overlap segments jointing

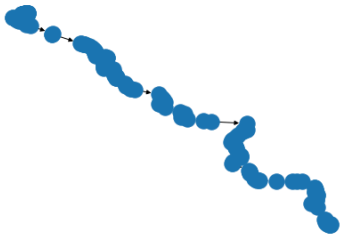
```
In [315]:
1 %%timeit
2 test = Joint_segment(Seoul, [seg1,seg4])
```

```
1.33 s ± 9.44 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
In [313]:
1 test.path
```

```
Out[313]: array([(102960, 0), (103166, 0), (103386, 0), (103602, 0), (103798, 0),
       (103990, 0), (104180, 0), (104414, 0), (104614, 0), (104858, 0),
       (104972, 0), (105093, 0), (105289, 0), (105521, 0), (105695, 0),
       (105877, 0), (106039, 0), (106193, 0), (106039, 0), (105877, 0),
       (105695, 0), (105521, 0), (105289, 0), (105093, 0), (104972, 0),
       (104858, 0), (104614, 0), (104414, 0), (104180, 0), (103990, 0),
       (103798, 0), (103602, 0), (103386, 0), (103166, 0), (102960, 0),
       (101903, 0), (101656, 0), ( 99770, 0), ( 96094, 0), ( 95600, 0),
       ( 93642, 0), ( 93508, 0), ( 93250, 0), ( 93249, 0), ( 91490, 0),
       ( 89164, 0), ( 87891, 0), ( 80812, 0), ( 77978, 0), ( 67758, 0),
       ( 67428, 0), ( 67408, 0), ( 67410, 0), ( 63001, 0), ( 63002, 0),
       (296431, 0), ( 62637, 0), ( 62545, 0), ( 61751, 0), ( 60835, 0),
       ( 60836, 0), ( 59505, 0), ( 58959, 0), ( 57835, 0), ( 57537, 0),
       ( 57511, 0), ( 57436, 0), ( 56742, 0), ( 56368, 0), ( 56300, 0),
       ( 55918, 0), ( 55894, 0), ( 55895, 0), ( 55758, 0), ( 55742, 0),
       ( 55205, 0), ( 55206, 0), ( 55025, 0), ( 54991, 0), ( 54397, 0),
       ( 54143, 0), ( 54101, 0), ( 54035, 0), ( 53835, 0), ( 53075, 0),
       ( 52175, 0), ( 51241, 0), ( 50772, 0), ( 48979, 0), ( 48980, 0),
       ( 48499, 0), ( 48359, 0), ( 47481, 0), ( 47086, 0), ( 46834, 0),
       ( 46823, 0), ( 46341, 0), ( 45981, 0), ( 45567, 0), ( 45529, 0),
       ( 41732, 0), ( 40463, 0), ( 40464, 0), ( 38222, 0), ( 38114, 0),
       ( 38113, 0), ( 33513, 0), ( 31125, 0), ( 30993, 0), ( 29691, 0),
       ( 27810, 0), ( 27124, 0), ( 26698, 0), ( 25814, 0), ( 22602, 0),
       ( 19566, 0), ( 18860, 0), ( 16774, 0), ( 16536, 0), ( 16232, 0),
       ( 16158, 0), ( 15776, 0), ( 14947, 0), ( 14913, 0), ( 14866, 0),
       ( 14868, 0), ( 14503, 0), ( 14406, 0), ( 14359, 0), ( 14361, 0),
       ( 13766, 0), ( 13351, 0), ( 13353, 0), ( 13024, 0), ( 12930, 0),
       ( 12929, 0), ( 12348, 0), ( 12265, 0), ( 12267, 0), ( 12028, 0),
       ( 11914, 0), ( 11193, 0), ( 11195, 0), ( 10830, 0), ( 10654, 0),
       (  9999, 0), (  9864, 0), (  9670, 0), (  9143, 0), (  9145, 0),
       (  9026, 0), (  8962, 0), (  8961, 0), (  8840, 0), (  8839, 0),
       (  8332, 0), (  8269, 0), (  8106, 0), (  8011, 0), (  8008, 0),
       (  7914, 0), (  7913, 0), (  7903, 0), (  7894, 0), (  7893, 0),
       (  7512, 0), (  7511, 0), (  7114, 0), (  7097, 0), (  7099, 0),
       (  6762, 0), (  6640, 0), (  6618, 0), (  6617, 0), (  6586, 0),
       (  6585, 0), (  6222, 0), (  6221, 0), (  5840, 0), (  5589, 0),
       (  5470, 0), (  5342, 0), (  4542, 0), (  4500, 0), (  4342, 0),
       (  4268, 0), (  4206, 0), (  4133, 0), (  4076, 0), (  4058, 0),
       (  4040, 0), (  4028, 0), (  4000, 0), (  3956, 0), (  3968, 0),
       (  3956, 0)], dtype=[('node', '<i4'), ('id', '<i4')])
```

```
In [314]:
1 test.plot(pos(Seoul))
```



약 200개정도 새로운 경로를 찾아갈 때 1초정도 걸림

overlap segments jointing

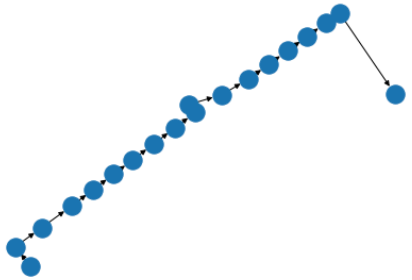
```
In [316]:
1 %%timeit
2 test = Joint_segment(Seoul, [seg1, seg2])
```

```
307 µs ± 8.81 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

```
In [310]:
1 test.path
```

```
Out[310]: array([(102960, 0), (103166, 0), (103386, 0), (103602, 0), (103798, 0),
       (103990, 0), (104180, 0), (104414, 0), (104614, 0), (104858, 0),
       (104972, 0), (105093, 0), (105289, 0), (105521, 0), (105695, 0),
       (105877, 0), (106039, 0), (106193, 0), (105119, 0)],
      dtype=[('node', '<i4'), ('id', '<i4')])
```

```
In [311]:
1 test.plot(pos(Seoul))
```



경로가 겹치는 경우 수백us걸림

20200702

Joint segment

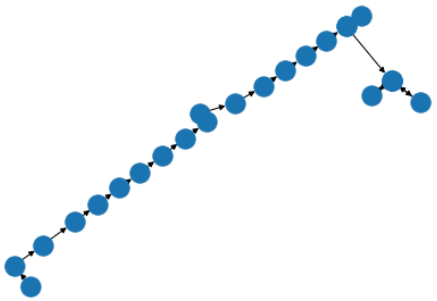
```
In [ ]:
%%time
test = Joint_segment(Seoul, [seg1,seg4])

CPU times: user 366 µs, sys: 980 µs, total: 1.35 ms
Wall time: 1.05 ms
```

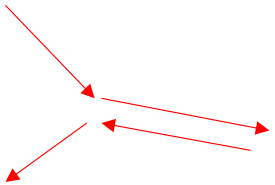
```
In [ ]:
seg1.path

Out[ ]:
array([(102960, 0), (103166, 0), (103386, 0), (103602, 0), (103798, 0),
       (103990, 0), (104180, 0), (104414, 0), (104614, 0), (104858, 0),
       (104972, 0), (105093, 0), (105289, 0), (105521, 0), (105695, 0),
       (105877, 0), (106039, 0), (106193, 0)],
      dtype=[('node', '<i4'), ('id', '<i4')])
```

```
In [ ]:
seg4.path

Out[ ]:
array([(105119, 0), (105367, 0), (105193, 0), (105367, 0)],
      dtype=[('node', '<i4'), ('id', '<i4')])
```

```
In [ ]:
test.path

Out[ ]:
array([(102960, 0), (103166, 0), (103386, 0), (103602, 0), (103798, 0),
       (103990, 0), (104180, 0), (104414, 0), (104614, 0), (104858, 0),
       (104972, 0), (105093, 0), (105289, 0), (105521, 0), (105695, 0),
       (105877, 0), (106039, 0), (106193, 0), (106039, 0), (105367, 0),
       (105119, 0), (105367, 0), (105193, 0), (105367, 0)],
      dtype=[('node', '<i4'), ('id', '<i4')])
```

```
In [ ]:
test.plot(pos(Seoul))
```





최단거리로 잇는 방법에도 각도 계산을 해주어야하나..?

20200702