ADS - Network analysis module homework

Task (World airports).

Q1.

Find the network connected components, report their number and the sizes.

Q2.

In the largest connected component find and report (print a list of city names and centrality scores) top 10 airports by each centrality metric (degree, betweenneass, closeness, pagerank with $alpha = 0.85$).

Q3.

Find maximum and average network distance of the shortest path (number of edges, regardless of size) between all pairs of the airports from the largest connected component. Visualize the shortest path of the maximum length.

Q4.

Find average network distance among the top 100 airports by pagerank (with $alpha = 0.85$) within the largest connected component

Q5.

Partition the entire network using Combo algorithm, visualize the partition

In [1]:
```
#add necessary libraries
import networkx as nx #library supporting networks
import matplotlib.pyplot as plt #plotting
import pandas as pd
import numpy as np
import scipy.stats as stat
#make sure plots are embedded into the notebook
%pylab inline
import statsmodels.formula.api as smf
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
path = 'https://raw.githubusercontent.com/CUSP-ADS2024/Data/main/'
cities = pd.read_csv(path + 'citiesTable.csv' , index_col=0 )
cities.head()
```

Out[2]:

city name	country name	most active airport	long. most active airport	lat. most active airport	number of routes	number incoming flights	number outcoming flights	number incoming domestic flights	ou d
London	United Kingdom	Heathrow	-0.103	51.795	1984	993	992	57	
Chicago	United States	Chicago Ohare Intl	-87.842	42.631	1406	705	702	526	
Paris	France	Charles De Gaulle	2.916	49.021	1254	626	629	68	
Moscow	Russia	Domododevo	38.510	55.681	1179	589	591	231	
Shanghai	China	Pudong	122.342	31.238	1115	560	556	357	

In [3]:
```
#now read the links
links = pd.read_csv('https://raw.githubusercontent.com/CUSP-ADS2024/Data/main/
links.head()
```

Out[3]:

	departure city	long. departure (decimal)	lat. departure (decimal)	departure country	arrival city	long. departure (decimal).1	lat. departure (decimal).1	arrival country	r
0	Sao Paulo	-46.116	-23.054	Brazil	Rio De Janeiro	-42.740	-22.682	Brazil	
1	Rio De Janeiro	-42.740	-22.682	Brazil	Sao Paulo	-46.116	-23.054	Brazil	
2	Beijing	116.974	40.133	China	Shanghai	122.342	31.238	China	
3	Johannesburg	28.410	-25.566	South Africa	Cape Town	19.002	-33.942	South Africa	
4	Honolulu	-157.871	21.531	United States	Tokyo	140.643	36.274	Japan	

In [4]:
```
#create the graph (assume connections are symmatric directionality)
Flights=nx.Graph()
Flights.add_nodes_from(cities.index)
EN=len(links.index)
edgelist=[(links['departure city'][j],links['arrival city'][j]) for j in range
Flights.add_edges_from(edgelist)
```

Q1.

Find the network connected components, report their number and the sizes.

In [5]:
```
cc=list(nx.connected_components(Flights))
```

In [6]:
```
len(cc)
```

Out[6]: 4

In [7]:
```
cc
```

Out[7]:
```
[{'Grand Rapids',
  'La Rioja',
  'Luoyang',
  'Gdansk',
  'Samana',
  'Appleton',
  'Dakhla',
  'Skiathos',
  'Muskegon',
  'Beslan',
  'Iquitos',
  'Iquique',
  'Caticlan',
  "Coff's Harbour",
  'Chattanooga',
  'Eagle',
  'Nyingchi',
  'New Stuyahok',
  'Haines',
  'Ciudad Obregon',
  'Ishigaki',
  'Sioux Lookout',
  'Kiana',
  'Tottori',
  'Roi Et',
  'The Valley',
  'Newark',
  'Wau',
  'Uberaba',
  'Miyazaki',
  'Deire Zor',
  'Pereira',
  'Armidale',
  'Puerto Carreno',
  'Kabri Dehar',
  'Allentown',
  'Borba',
  'Nantes',
  'Managua',
  'Hanoi',
  'Monbetsu',
  'Bergamo',
  'Lubumashi',
  'Muenster/osnabrueck',
  'Ua Huka',
  'Faleolo',
  'Minacu',
  'Jacksn Hole',
  'Nakhon Si Thammarat',
  'Hong Kong',
  'Dresden',
  'Sibiu',
  'Sao Nocolau Island',
  'Nadzab',
  'Atlantic City',
  'Florencia',
  'Maniitsoq',
  'Mykonos',
  'Fort Mcpherson',
  'Kardla',
```

```
'Abakan',
"N'djamena",
'Juba',
'Philadelphia',
'Sacheon',
'Colville Lake',
'Malindi',
'Kiritimati',
'Broome',
'Gizo',
'Yuncheng',
'Bolzano',
'Basse Terre',
'Afutara',
'Port Said',
'Port Lions',
'Manchester NH',
'Mohe County',
'Kristianstad',
'Nyala',
'Illizi',
'Wuhan',
'Colombia',
'Islay',
'Killeen',
'Hooper Bay',
'Benin',
'Kazan',
'Panama City',
'Kramfors',
'Cuenca',
'Jaipur',
'Jersey',
'Uzhgorod',
'Casper',
'Leon',
'Redding',
'Toluca',
'Naryan-Mar',
'Alamosa',
'Miles City',
'Chittagong',
'Gwalior',
'Lewiston',
'Washington',
'Pointe-a-pitre',
'Menongue',
'Nizhnevartovsk',
'Ambler',
'Datong',
'Nan',
'Hiva-oa',
'Tumaco',
'Minto',
'Rotterdam',
'High Level',
'Norfolk',
'Guangzhou',
'Puvirnituq',
'Kalamazoo',
```

```
'Gaya',
'Lifou',
'Terrace',
'Cebu',
'Kasos',
'Ouzinkie',
'Beograd',
'Cabo Frio',
'Ivujivik',
'San Sebastian',
'Rockhampton',
'Ji-Paraná',
'Modesto',
'Shizuoka',
'Gothenborg',
'Jos',
'Vilankulu',
'Salt Lake City',
'New Orleans',
'Asahikawa',
'Navoi',
'Bombay',
'Alamogordo',
'Montego Bay',
'Yakutat',
'Keewaywin',
'Diyabakir',
'Alpena',
'Marsh Harbor',
'Brunei',
'Vernal',
'Kegaska',
'Phnom-penh',
'Harrisburg',
'Ljubliana',
'Sendai',
'Allakaket',
'LaCrosse',
'Rio De Janeiro',
'Savannakhet',
'Bima',
'Riga',
'Funchal',
'Kish Island',
'Helsinki',
'Kayseri',
'Formosa',
'Williston',
'Lake Charles',
'Tepic',
'Ulsan',
'La Serena',
'Deer Lake',
'Puerto Montt',
'Osh',
'Nantong',
'Ballina Byron Bay',
'Mountain Village',
'Wrangell',
'Schefferville',
```

```
'Aleppo',
'Loreto',
'Helgoland',
'Tiksi',
'Nanki-shirahama',
'Puerto Plata',
'Kashi',
'Manokotak',
'Sittwe',
'Van',
'State College Pennsylvania',
'Jinzhou',
'Cheyenne',
'North Eleuthera',
'Vaxjo',
'Ronne',
'Zhijiang',
'Comox',
'Bamako',
"Governor's Harbor",
'Ndola',
'Salta',
'Sohag',
'Juazeiro Do Norte',
'Faisalabad',
'Masai Mara',
'Lambasa',
'Orenburg',
'Palangkaraya',
'Tiree',
'Beaver',
'Sibu',
'Moa',
'Inhambane',
'Tirana',
'Tallahassee',
'Suceava',
'Dalian',
'Tomsk',
'Tabatinga',
'Jammu',
'Sachs Harbour',
'Buonmethuot',
'King Cove',
'Piedras Negras',
'Shanghai',
'Pensacola',
'Teheran',
'Alghero',
'Saratov',
'Mpumalanga',
'Karratha',
'Branson',
'Xiamen',
'Muscle Shoals',
'Yiwu',
'Meridian',
'Allahabad',
'Pavlodar',
'St. Thomas',
```

```
                'Mar Del Plata',
                'Tabubil',
                'Yacuiba',
                'Tuguegarao',
                'Choiseul Bay',
                'Mare',
                'Phoenix',
                'Norilsk',
                'King Salmon',
                'Dalaman',
                'Red Lake',
                'Simferopol',
                'Puebla',
                'Marktoberdorf',
                'Stokmarknes',
                'Puerto Maldonado',
                'Basco',
                'Columbus Mississippi',
                'Minatitlan',
                'Eek',
                'Manaus',
                'Holy Cross',
                'Pangnirtung',
                'Iles De La Madeleine',
                'Port Moresby',
                'Little Rock',
                'Lewisburg',
                'Harlingen',
                'Montreal',
                'Melilla',
                'Hue',
                'Heraklion',
                'Louisville',
                'Las Palmas',
                'Chitral',
                'Learmonth',
                'Leros',
                'Bremen',
                'Wewak',
                'Bahia Solano',
                'Hahn',
                'Foshan',
                'Chokurdah',
                'Eirunepe',
                'Humberside',
                'East Midlands',
                'Zweibruecken',
                'Murmansk',
                'Teesside',
                'Mitiaro Island',
                'Taipei',
                'Kulyab',
                'Xining',
                'Saravena',
                'Kobuk',
                'Agadir',
                'Bilbao',
                'Nepalgunj',
                'Sao Paulo',
                'Ujung Pandang',
```

```
'Westsound',
'Nizhniy Novgorod',
'Halmstad',
'Catarman',
'Cockburn Town',
'Khartoum',
'Natal',
'Heihe',
'Minot',
'Hotan',
'Osaka',
'Guantanamo',
'Campina Grande',
'Odessa',
'Kelowna',
'Lubango',
'Egilsstadir',
'Pasco',
'Taupo',
'West Tinian',
'Solwesi',
'Ozamis',
'Beauvais',
'Tongoa Island',
'Fuzhou',
'Chub Cay',
'Paderborn',
'Hailar',
'San Tome',
'Rovaniemi',
'Ghardaia',
'Vilnius',
'Tolagnaro',
'Launceston',
'Port Protection',
'Medford',
'Cork',
'Kerikeri',
'Brazzaville',
'Rouyn',
'Ronchi De Legionari',
'Tuxtla Gutierrez',
'Hays',
'Indianapolis',
'Conson',
'Gambell',
'Idaho Falls',
'Temuco',
'Kimmirut',
'Ajaccio',
'Durango',
'Toksook Bay',
'Sandnessjoen',
'Adana',
'Levaldigi',
'Mpacha',
'Kingfisher Lake',
'Brainerd',
'Lawas',
'Tari',
```

```
'Nuquí',
'Santa Clara',
'Tawau',
'Burlington',
'Missoula',
'Brownsville',
'Huai An',
'Nanchong',
'Longview',
'Medellin',
'Belize',
'Haiphong',
'Waala',
'Walaha',
'Copenhagen',
'Busuanga',
'Karluk',
'Baghdogra',
'San Jose Del Cabo',
'Puerto Inírida',
'Tauranga',
'Nanchang',
'Gisborne',
'Darwin',
'Foz Do Iguacu',
'Barcelona',
'Pickle Lake',
'Simara',
'Kasigluk',
'Port Hope Simpson',
'Kinmen',
'Chevery',
'Lethbridge',
'Bettles',
'Bisha',
'Antsiranana',
'Tasiujaq',
'Davao',
'Antalaha',
'Ahwaz',
'Ashkhabad',
'Postville',
'Exeter',
'Morgantown',
'Hobbs',
'Hagerstown',
'Syktyvkar',
'Penang',
'Kandahar',
'Flin Flon',
'Ponce',
'Karaganda',
'Sialkot',
'Corozal',
'Tacheng',
'Tureia',
'Bangalore',
'Amboseli National Park',
'Bar Harbor',
'Tenakee Springs',
```

```
'Naples',
'San Vincente De Caguan',
'Monteria',
'Tenerife',
'Waingapu',
'Ilorin',
'Midland',
'Lhok Seumawe-Sumatra Island',
'Lopez',
'Menorca',
'Narvik',
'Aspen',
'Balmaceda',
'Karpathos',
'Billund',
'Pendleton',
'Sarasota',
'Dryden',
'Pokhara',
'Chihuahua',
'Colonel Hill',
'Caracas',
'Sachigo Lake',
'Svalbard',
'Yellowknife',
'Ruby',
'Torreon',
'Zurich',
'Coppermine',
'Craig Cove',
'Paamiut',
'Riyadh',
'Sambava',
'Kaltag',
'Ardabil',
'Ancona',
'Akron',
'Circle',
'Nice',
'Santo Domingo',
'Yining',
'Devils Lake',
'Cody',
'Olbia',
'Campbell River',
'Salekhard',
'Whangarei',
'Ogoki Post',
'Erechim',
'Show Low',
'Abbotsford',
'Caldas Novas',
'Pangkalan Bun',
'Badajoz',
'Pyongyang',
'Ulanhot',
'Novy Urengoy',
'Dongsheng',
'Wajima',
'Graciosa Island',
```

```
'Sangafa',
'Amilcar Cabral',
'Lugano',
'Adelaide',
'Phuket',
'Dushanbe',
'Kaadedhdhoo',
'Xilinhot',
'Zamboanga',
'Fort Chipewyan',
'Kuala Lumpur',
'Nis',
'Joinville',
'Cap Haitien',
'Parma',
'Zagreb',
'Fort Yukon',
'Sucre',
'Attawapiskat',
'Dipolog',
'Dibrugarh',
'Alta',
'Whyalla',
'Dourados',
'Salamanca',
'Liverpool',
'Los Angeles',
'Granada',
'Touho',
'Akulivik',
'Ho Chi Minh City',
'Lar',
'Punta Arenas',
'Uberlandia',
'Joenkoeping',
'Dawadmi',
'Moroni',
'Tête-à-la-Baleine',
'Petropavlovsk',
'Syracuse',
'Long Akah',
'Kwethluk',
'Kitakyushu',
'Barter Island',
'Kona',
'Melbourne',
'Khasab',
'Roxana',
'Takaroa',
'Nawabshah',
'Alliance',
'Mulu',
'Greenville',
'Gafsa',
'Kaunas',
'Bagotville',
'Quujjuaq',
'Rostov',
'Riverton WY',
'Camaguey',
```

```
'Posadas',
'Rzeszow',
'Nuernberg',
'Mytilini',
'Masset',
'Togiak Village',
'Mexico City',
'Liping',
'Jeddah',
'Changuinola',
'Treasure Cay',
'Aurangabad',
'Criciuma',
'Atiu Island',
'Huambo',
'Hughes',
'Luena',
'Vardø',
'Sand Point',
'Raleigh-durham',
'Jackson',
'Bristol',
'Tacloban',
'White Mountain',
'Nouakschott',
'Grand Forks',
'Scarborough',
'Point Baker',
'Honolulu',
'Rockland',
'Sao Luis',
'Orlando',
'Kichinau Fir/acc/com',
'Yibin',
'Marília',
'Bandar Lampung-Sumatra Island',
'Sion',
'Malacca',
'Karlsruhe/Baden-Baden',
'Yulin',
'Larsen Bay',
'Old Harbor',
'Bergen',
'Lalibella',
'Rutland',
'Farmington',
'Fargo',
'Palm Springs',
'Laghouat',
'Antalya',
'Ganzhou',
'Ilulissat',
'Lençóis',
'Manston',
'Andros Town',
'Pelican',
'Novokuznetsk',
'Arua',
'Invercargill',
'Tuntutuliak',
```

```
'BRISTOL',
'Longyan',
'Kone',
'Kristiansand',
'Colombo',
'Fagernes',
'Mount Hagen',
'McCook',
'Sovetskiy',
'Tarbes',
'Garden City',
'Coral Harbour',
'Podgorica',
'Tsushima',
'Satu Mare',
'Deering',
'Mehamn',
'Akhiok',
'Gassim',
'La Ceiba',
'Kamuela',
'Tegucigalpa',
'Waikabubak-Sumba Island',
'Konya',
'Tengchong',
'Medicine Hat',
'Annaba',
'Narssarssuaq',
'Lamidanda',
'Decatur',
'Kiev',
'Blantyre',
'Makung',
'Owensboro',
'Punta Gorda',
'Sao Vicente Island',
'Kumejima',
'Boise',
'Gaberone',
'Mashhad',
'Catamarca',
'Sheridan',
'Takoradi',
'Treviso',
'Beziers',
'Port Hedland',
'Sivas',
'Alderney',
'Copiapo',
'Cyclades Islands',
'Yerevan',
'Geralds',
'Termez',
'Salerno',
'Shageluk',
'Linz',
'Victoria',
'Barnstable',
'Peshawar',
'Kabul',
```

```
'Hamburg',
'Cagliari',
'Dijon',
'Urgench',
'Okayama',
'Sioux Falls',
'Tampere',
'Elista',
'Dawson',
'Nalchik',
'Brive',
'Edinburgh',
'Pucon',
'Tubuai',
'Waskaganish',
'Long Datih',
'Townsville',
'Golfito',
'Changde',
'Johor Bahru',
'Klagenfurt',
'Reno',
'Szczechin',
'Kilimanjaro',
'Beijing',
'Trujillo',
'Igarka',
'Dhangarhi',
'Benghazi',
'Baku',
'Bandar Mahshahr',
'Fort Leonardwood',
'Lhasa',
'Yuma',
'Harare',
'Yichang',
'Nemiscau',
'Valdez',
'Saint George',
'Lahad Datu',
'Dayton',
"Land's End",
'Maastricht',
'Kulusuk',
'Guarapuava',
'Wagga Wagga',
'Wichita',
'Stony River',
'Rio Negro',
'Sanya',
'Flint',
'Lima',
'Uralsk',
'Acapulco',
'Sandakan',
'Raivavae',
'Hatay',
'La Fortuna/San Carlos',
'Nouadhibou',
'Jujuy',
```

```
                'Novosibirsk',
                'Oranjemund',
                'Sukkur',
                'Vijayawada',
                'Guiyang',
                'Anaa',
                'Pilot Point',
                'Hodeidah',
                'Prague',
                'Pisa',
                'Hilton Head',
                'Umtata',
                'Bangor',
                'Moab',
                'Ogdensburg',
                'Poza Rico',
                'International Falls',
                'North Whale Pass',
                'Bucaramanga',
                'Myrtle Beach',
                'San Rafael',
                'Eday',
                'Ca Mau',
                'La Coruna',
                'Marrakech',
                'Blagoveschensk',
                'Lusaka',
                'Bintulu',
                'Mendoza',
                'Bournemouth',
                'Jessore',
                'Vieques Island',
                'Sundsvall',
                'Barnaul',
                'Tanjung Manis',
                'Pingtung',
                'Delhi',
                'Fort Albany',
                'Georgetown',
                'Pittsburgh (pennsylva)',
                'Patos de Minas',
                'Ji An',
                'Makemo',
                'North Platte',
                'Fort Smith',
                'Adrar',
                'Londrina',
                'Manado',
                'Myitkyina',
                'Johannesburg',
                'St. Louis',
                'Venice',
                'Castres',
                'Tehran',
                'Kwangju',
                'Sandefjord',
                'Chignik Lagoon',
                'Kostanay',
                'Southend',
                'Zaragoza',
```

```
                'Tete',
                'Batman',
                'Yola',
                'Tresco',
                'Zachar Bay',
                'Kiunga',
                'Lodwar',
                'Spokane',
                'Deline',
                'Aniwa',
                'Casablanca',
                'Makale',
                'Bridgetown',
                'Pulau',
                'Perth',
                'Trivandrum',
                'Lorient',
                'Santo',
                'Coari',
                'New Plymouth',
                'Faro',
                'Barra',
                'Nuevo Laredo',
                'Yopal',
                'Inukjuak',
                'Detroit',
                'Puerto Asis',
                'Vigo',
                'Kagau Island',
                'Nunam Iqua',
                'Lanai',
                'Beira',
                'Williams Harbour',
                'Kengtung',
                'Uruguaiana',
                'San Julian',
                'Cherepovets',
                'Bardufoss',
                'Toowoomba',
                'Anadyr',
                'Hailey',
                'Fort Simpson',
                'Leknes',
                'Quelimane',
                'Arica',
                'Birjand',
                'Shiraz',
                'Sept-iles',
                'Helena',
                'Kosrae',
                'Moorea',
                'Ikaria',
                'Arutua',
                'San Juan',
                'Vientiane',
                'Chios',
                'Kyaukpyu',
                'Angmagssalik',
                'Arctic Bay',
                'Yinchuan',
```

```
'Maun',
'Tuluksak',
'Prince Pupert',
'Matsuyama',
'Tromso',
'Malanje',
'Ninbo',
'Alicante',
'Nantucket',
'Mangaia Island',
'Huron',
'Isfahan',
'Contadora Island',
'Tioman',
'Qeqertarsuaq Airport',
'Dundee',
'Ibiza',
'Pasto',
'Canberra',
'St. Paul Island',
'Norsup',
'Rock Sound',
'Dillingham',
'Frankfurt',
'Barrancabermeja',
'Kanazawa',
'Bario',
'Kajaani',
'Jiuzhaigou',
'Malang',
'Bukoba',
'George',
'Tanjung Pandan',
'Manizales',
'Nandayure',
'Garoua',
'Bratislava',
'Kigoma',
'Stuttgart',
'Elfin Cove',
'Agatti Island',
'Iliamna',
'Quzhou',
'Gorontalo',
'Masterton',
'Port-vila',
'Isafjordur',
'Lago Agrio',
'Stockholm',
'Angoon',
'Castlegar',
'Charlottetown',
'Zakynthos',
'Sitia',
'Nosara Beach',
'Yeosu',
'Hassi-messaoud',
'Turaif',
'Kitoi Bay',
'Okinawa',
```

```
'Charleroi',
'Fonte Boa',
'Jiamusi',
'Toliara',
'Klamath Falls',
'Campbeltown',
'Provo',
'Abadan',
'Nagoya',
'Salalah',
'Cordoba',
'Traverse City',
'Romblon',
'Port Williams',
'Concepcion',
'Calabar',
'Guaymas',
'Jijel',
'Ahmedabad',
'Zhob',
'Wick',
'Lonorore',
'Nyuang U',
'Jakarta',
'Sao Tome',
'Bandung',
'Sanaa',
'Perugia',
'Chengdu',
'Ballalae',
'Bacolod',
'Aizwal',
'Vail',
'Timika',
'Bydgoszcz',
'Kalibo',
'Dominica',
'Sacramento',
'Sleetmute',
'Ciudad del Este',
'Lawton',
'Dinard',
'St.-denis',
'Anjouan',
'Diamantina',
'Laoag',
'Matei',
'Villavicencio',
'Samburu South',
'Labuhan Bajo',
'Floro',
'Thunder Bay',
'Taiz',
'Kerkyra/corfu',
'Marquette',
'Jacksonville NC',
'Fernando Do Noronha',
'Hakodate',
'Brevig Mission',
'Central',
```

```
 'Spring Point',
 'Kuwait',
 'Puerto Lempira',
 'Corumba',
 'Whale Cove',
 'Shreveport',
 'Yakutsk',
 'Stella Maris',
 'Dazhou',
 'Algier',
 'Gelendzik',
 'Urumqi',
 'Tiruchirappalli',
 'Westport',
 'Ribeirao Preto',
 'Anelghowhat',
 'Ondangwa',
 'Pelotas',
 'Eskimo Point',
 'Zhuhai',
 'Kingstown',
 'Huntington',
 'Bishkek',
 'Rennes',
 'Hyderabad',
 'Avalon',
 'Manhattan',
 'Stockton',
 'Bujumbura',
 'Aurillac',
 'Baroda',
 'Kotlik',
 'Sao Paulo de Olivenca',
 'Quaqtaq',
 'Kerry',
 'Mackay',
 'Kunsan',
 'Nausori',
 'Saurimo',
 'Oujda',
 ...},
{'Akutan', 'Nikolski', 'Unalaska'},
{'Boulder City', 'Grand Canyon West'},
{'Block Island', 'Leixlip'}]
```

Q2.

In the largest connected component find and report (print a list of city names and centrality scores) top 10 airports by each centrality metric (degree, betweenneass, closeness, pagerank with $alpha = 0.85$).

In [8]:
```
max_cc = max(cc,key=len)
```

In [9]:
```
max_cc_graph = nx.Graph()
max_cc_graph.add_nodes_from(max_cc)
max_cc_graph.add_edges_from(edgelist)
```

In [10]:
```
max_cc_graph
```

Out[10]:
```
<networkx.classes.graph.Graph at 0x780e39eb6290>
```

In [11]:
```
#output top tn centrality scores, given the dictionary d
def topdict(d,tn):
    ind=sorted(d, key=d.get, reverse=True)
    for i in range(0,tn):
       print('{0}|{1} - {2}'.format(i+1, ind[i], d[ind[i]]))
```

In [12]:
```
mc1=dict(nx.degree(max_cc_graph))
ind1=topdict(mc1,10)
```

```
1|London - 311
2|Paris - 267
3|Frankfurt - 232
4|Moscow - 229
5|Amsterdam - 208
6|Atlanta - 208
7|Chicago - 199
8|Beijing - 184
9|Dallas-fort Worth - 181
10|Istanbul - 181
```

In [13]:
```
mc2 = nx.betweenness_centrality(max_cc_graph)
ind2=topdict(mc2,10)
```

```
1|London - 0.09034228164817266
2|Anchorage - 0.08177900060547011
3|Paris - 0.07747744448398254
4|Moscow - 0.06075222333964135
5|Chicago - 0.057862725040192346
6|Frankfurt - 0.05426812317265433
7|Seattle - 0.052472738379445105
8|Dubai - 0.046225264730113666
9|Tokyo - 0.04563994105465253
10|Toronto - 0.04510249041924881
```

In [14]:
```
mc3 = nx.closeness_centrality(max_cc_graph)
ind4=topdict(mc3,10)
```

```
1|London - 0.4223634613466614
2|Frankfurt - 0.4165055203725343
3|Paris - 0.4153998050047715
4|Amsterdam - 0.4084935977492648
5|New York - 0.3995989790885954
6|Toronto - 0.3954005059740856
7|Los Angeles - 0.3950852364676611
8|Chicago - 0.39180500848960853
9|Dubai - 0.391701766063129
10|Newark - 0.38873122442719005
```

In [15]:
```
mc4 = nx.pagerank(max_cc_graph,0.85)
ind4=topdict(mc4,10)
```

```
1|London - 0.006643690212771941
2|Moscow - 0.006394773861081004
3|Paris - 0.00628119041679376
4|Atlanta - 0.005005165152212129
5|Frankfurt - 0.004791527894478361
6|Chicago - 0.004698791227789343
7|Dallas-fort Worth - 0.004682221065566249
8|Denver - 0.004615583755208124
9|Houston - 0.004388076110471842
10|Amsterdam - 0.004302029752645577
```

Q3.

Find maximum and average network distance of the shortest path (number of edges, regardless of size) between all pairs of the airports from the largest connected component. Visualize the shortest path of the maximum length.

From the largest connected component

In [16]:
```
max_cc_graph
```

Out[16]:
```
<networkx.classes.graph.Graph at 0x780e39eb6290>
```

In [17]:
```
CityPos={c:(cities['long. most active airport'][c],cities['lat. most active ai
         for c in cities.index}
```

In [18]:
```
sp = dict(nx.shortest_path_length(max_cc_graph))
```

In [19]:
```
diameter=0; s=0; n=0;
for i in sp.items():
    for j in i[1].items():
        if j[1]>diameter:
            s += j[1]
            n += 1

print('Average distance is:', s/n)
```

```
Average distance is: 3.9398398584324226
```

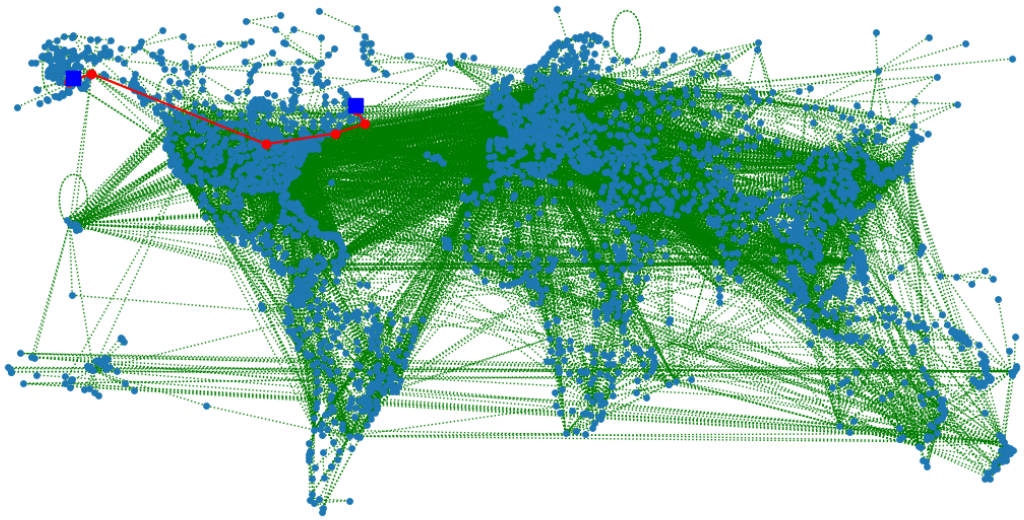
In [20]:
```
diameter=0; i0=0; j0=0;
for i in sp.items():
    for j in i[1].items():
        if j[1]>diameter:
            diameter=j[1]
            i0=i[0]
            j0=j[0]
print('{0} - {1} : {2}'.format(i0,j0,diameter))
```

```
Port Hope Simpson - Koliganek : 12
```

In [21]:
```
def visualize_path(path):
    plt.figure(figsize = (12,6))
    nx.draw(max_cc_graph,pos=CityPos,with_labels=False,arrows=False,node_size=
    x=[CityPos[v][0] for v in path]
    y=[CityPos[v][1] for v in path]
```

```
    plt.plot(x,y,'ro-')
    plt.plot([x[0],x[-1]],[y[0],y[-1]],'bs',markersize=10)
```

In [22]:
```
Lpath=nx.shortest_path(max_cc_graph,i0,j0)
visualize_path(Lpath)
```



From the whole network

In [23]:
```
wsp = dict(nx.shortest_path_length(Flights))
```

In [24]:
```
diameter=0; s=0; n=0;
for i in wsp.items():
    for j in i[1].items():
        if j[1]>diameter:
            s += j[1]
            n += 1

print('Average distance is:', s/n)
```

```
Average distance is: 3.9398398584324226
```

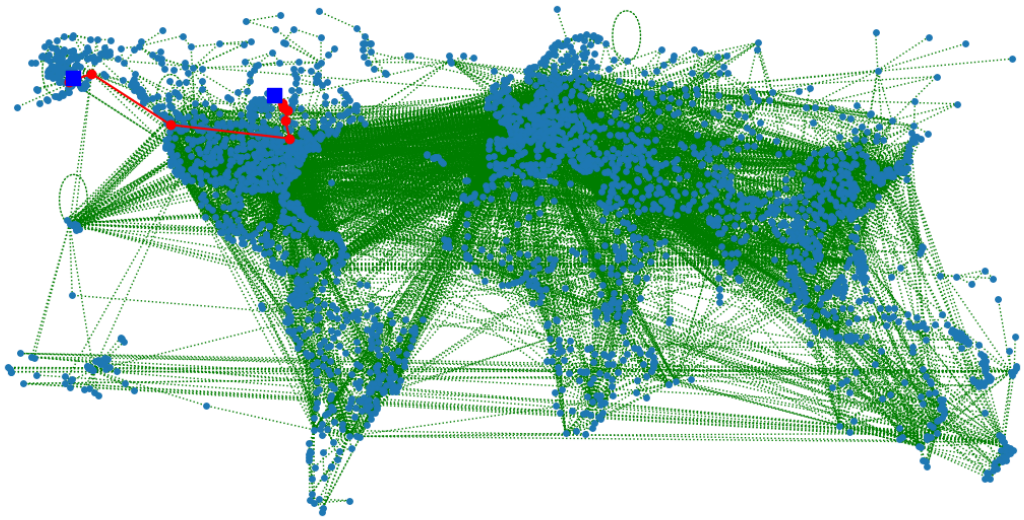
In [25]:
```
diameter=0; i0=0; j0=0;
for i in wsp.items():
    for j in i[1].items():
        if j[1]>diameter:
            diameter=j[1]
            i0=i[0]
            j0=j[0]
print('{0} - {1} : {2}'.format(i0,j0,diameter))
```

```
Peawanuck - Koliganek : 12
```

In [26]:
```
def visualize_path(path):
    plt.figure(figsize = (12,6))
    nx.draw(Flights,pos=CityPos,with_labels=False,arrows=False,node_size=15,sty
    x=[CityPos[v][0] for v in path]
    y=[CityPos[v][1] for v in path]
```

```
    plt.plot(x,y,'ro-')
    plt.plot([x[0],x[-1]],[y[0],y[-1]],'bs',markersize=10)
```

In [27]:
```
Lpath=nx.shortest_path(Flights,i0,j0)
visualize_path(Lpath)
```



Q4.

Find average network distance among the top 100 airports by pagerank (with $alpha = 0.85$) within the largest connected component

In [28]:
```
#output top tn centrality scores, given the dictionary d
def topdict1(d, tn):
    ind = sorted(d, key=d.get, reverse=True)  # Sort the nodes based on PageRa
    result = []
    for i in range(tn):  # Loop to collect top tn nodes
        result.append((ind[i], d[ind[i]]))  # Append the node and its PageRank
    # Create a DataFrame from the list of tuples
    df = pd.DataFrame(result, columns=['Node', 'PageRank'])
    return df  # Return the DataFrame
```

In [29]:
```
mc5 = nx.pagerank(max_cc_graph,0.85)
ind5=topdict1(mc5,100)
ind5
```

Out[29]:

	Node	PageRank
0	London	0.006644
1	Moscow	0.006395
2	Paris	0.006281
3	Atlanta	0.005005
4	Frankfurt	0.004792
...	...	...
95	Birmingham	0.001674
96	Kiev	0.001672
97	Honiara	0.001662
98	Cancun	0.001662
99	Osaka	0.001661

100 rows × 2 columns

In [30]:
```
ind5['PageRank'].mean()
```

Out[30]:
```
0.0027938672445089975
```

Q5.

Partition the entire network using Combo algorithm, visualize the partition

In [31]:
```
!pip install pycombo
!pip install geopandas
```

```
Collecting pycombo
  Downloading pycombo-0.1.7.tar.gz (136 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 136.6/136.6 kB 2.8 MB/s eta 0:0
0:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting pybind11<3.0.0,>=2.6.1 (from pycombo)
  Downloading pybind11-2.12.0-py3-none-any.whl (234 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 235.0/235.0 kB 9.1 MB/s eta 0:0
0:00
Building wheels for collected packages: pycombo
  Building wheel for pycombo (pyproject.toml) ... done
  Created wheel for pycombo: filename=pycombo-0.1.7-cp310-cp310-manylinux_2_35
_x86_64.whl size=208098 sha256=c2653968b45cd65e18200959573b669fc108557b1123fc6
ab45483dea04a6b5a
  Stored in directory: /root/.cache/pip/wheels/21/90/69/e7f601be9740da0df241d6
b15d29c7d885896dd84a8eeaeaf2
Successfully built pycombo
Installing collected packages: pybind11, pycombo
Successfully installed pybind11-2.12.0 pycombo-0.1.7
Requirement already satisfied: geopandas in /usr/local/lib/python3.10/dist-pac
kages (0.13.2)
Requirement already satisfied: fiona>=1.8.19 in /usr/local/lib/python3.10/dist
-packages (from geopandas) (1.9.6)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-pac
kages (from geopandas) (24.0)
Requirement already satisfied: pandas>=1.1.0 in /usr/local/lib/python3.10/dist
-packages (from geopandas) (2.0.3)
Requirement already satisfied: pyproj>=3.0.1 in /usr/local/lib/python3.10/dist
-packages (from geopandas) (3.6.1)
Requirement already satisfied: shapely>=1.7.1 in /usr/local/lib/python3.10/dis
t-packages (from geopandas) (2.0.3)
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.10/dist
-packages (from fiona>=1.8.19->geopandas) (23.2.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packa
ges (from fiona>=1.8.19->geopandas) (2024.2.2)
Requirement already satisfied: click~=8.0 in /usr/local/lib/python3.10/dist-pa
ckages (from fiona>=1.8.19->geopandas) (8.1.7)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.1
0/dist-packages (from fiona>=1.8.19->geopandas) (1.1.1)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.10/dist-pa
ckages (from fiona>=1.8.19->geopandas) (0.7.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from fiona>=1.8.19->geopandas) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python
3.10/dist-packages (from pandas>=1.1.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.1.0->geopandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dis
t-packages (from pandas>=1.1.0->geopandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist
-packages (from pandas>=1.1.0->geopandas) (1.25.2)
```

In [32]:

```
import pycombo # combo community detection package
import networkx as nx #library supporting networks
import matplotlib.pyplot as plt #plotting
import pandas as pd
import numpy as np
import scipy.stats as stat
```

```
from scipy import optimize
#make sure plots are embedded into the notebook
%pylab inline
import statsmodels.formula.api as smf
import os
from networkx.algorithms import community
import warnings
warnings.filterwarnings('ignore')
```

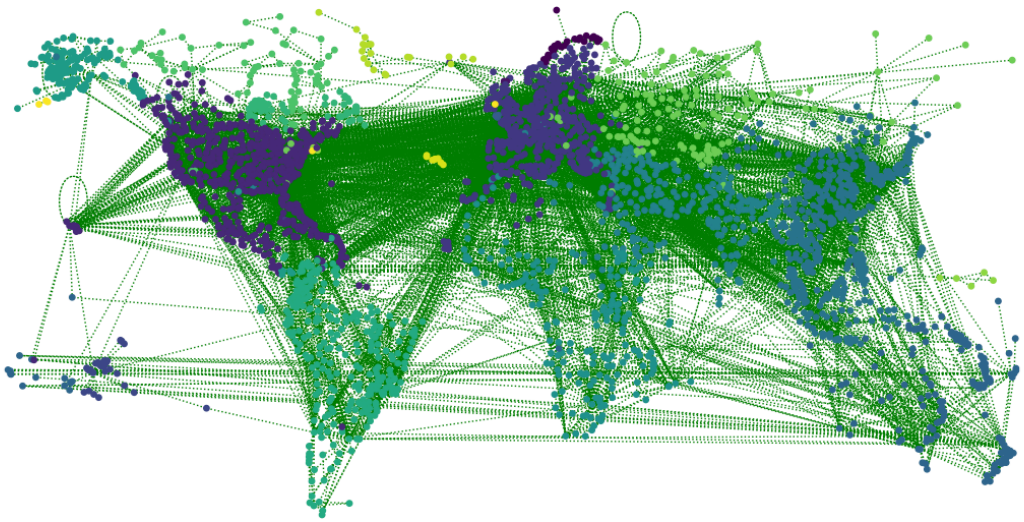
```
Populating the interactive namespace from numpy and matplotlib
/usr/local/lib/python3.10/dist-packages/IPython/core/magics/pylab.py:159: User
Warning: pylab import has clobbered these variables: ['i0']
`%matplotlib` prevents importing * from pylab and numpy
  warn("pylab import has clobbered these variables: %s"  % clobbered +
```

In [33]:
```
PCFlights, PCFlightsMod = pycombo.execute(Flights, max_communities = 0) #keep
PCFlightsMod
```

Out[33]:
```
0.6605515045824883
```

In [34]:
```
def visualizePartition(G,partition,pos):
  N=len(G.nodes())
  s=4+4*int(log10(N))
  plt.figure(figsize=(12,6))
  PN=max(partition.values())
  my_cmap = matplotlib.cm.hsv(np.linspace(0,1,PN+1)) #create a colormap for a
  c=[]
  for n in G.nodes():
    c.append(1.0*partition[n]/PN)
  nx.draw(G,pos=pos,with_labels=False,node_size=15,node_color=c,style='dotted'
```

In [35]:
```
visualizePartition (Flights,PCFlights,CityPos)
```



In [35]:
```

```