Trabalho 1 - Similaridade em Grafos

Otimização em Grafos 2021.1

- O objetivo do trabalho é aplicar os conceitos de Isomorfismo e Similaridade na análise dos dados do Openflights (https://openflights.org/data.html), que contém informações sobre Aeroportos, Linhas Aéreas e Rotas percorridas por aviões em todo o mundo.

Domingos Bruno Sousa Santos

- Todo o repositório com todos os arquivos e códigos para download podem ser encontrados aqui Repositório GitHub.

1) Comprender como estão estruturados os bancos de dados do OpenFlights:

- Airports Database – airports.dat
- Airlines – airlines.dat
- Routres – routes.dat

2) Criar um grafo G_1 de entidades e características a partir do banco de dados, usando as instruções seguintes.

O conjunto de vértices é formado pela união de:

- A – o conjunto dos aeroportos situados no Brasil
- L – o conjunto das linhas aéreas situadas no Brasil
- R – o conjunto das rotas entre aeroportos brasileiros

As arestas são formada pelos seguintes pares:

- $\{a, l\}$ – onde $a \in A$, $l \in L$, e a linha aérea l opera algum vôo com origem ou destino no aeroporto a.
- $\{a, r\}$ – onde $a \in A$, $r \in R$, e a rota r possui origem ou destino no aeroporto a.

3) Implementar um algoritmo baseado na Similaridade de Jaccard para, dado dois aeroportos (a_1, a_2), calcular a similaridade entre eles, considerando as linhas aéreas que operam em ambos e as rotas que possuem os mesmos como origem ou destino.

4) Implementar um algoritmo de baseado na Similaridade por cosseno para, dado dois aeroportos (a_1, a_2), calcular a similaridade entre eles. Para o cálculo, considere a vizinhança em termos de linhas aéreas de cada aeroporto, e como peso da aresta o número de rotas daquela linha aérea que operam naquele aeroporto.

1) Analisando e entendendo como os dados estão estruturados.

In [1]:
```
# Importando as bibliotecas usadas
import pandas as pd # Biblioteca usada para a leitura dos aquivos
import numpy as np  # Biblioteca usada para trabalhar com listas
import math # Usada para fazer operações matematicas
from tqdm import tqdm # Biblioteca usada apenas para ter um feedback visual e
```

In [2]:
```
# Leitura do arquivo airports.dat
df_airports = pd.read_csv('data/airports.dat', sep=",",
                  names=["Airport ID", "Name", "City", "Country", "IATA", "ICA
                         "Longitude", "Altitude", "Timezone", "DST",
                        "Tz database time zone", "Type", "Source"])
```

In [3]:
```
# Como está estutura o arquivo airports.dat
df_airports
```

Out[3]:

	Airport ID	Name	City	Country	IATA	ICAO	Latitude	Longitude	Altitude	Tim
0	1	Goroka Airport	Goroka	Papua New Guinea	GKA	AYGA	-6.081690	145.391998	5282	
1	2	Madang Airport	Madang	Papua New Guinea	MAG	AYMD	-5.207080	145.789001	20	
2	3	Mount Hagen Kagamuga Airport	Mount Hagen	Papua New Guinea	HGU	AYMH	-5.826790	144.296005	5388	
3	4	Nadzab Airport	Nadzab	Papua New Guinea	LAE	AYNZ	-6.569803	146.725977	239	
4	5	Port Moresby Jacksons International Airport	Port Moresby	Papua New Guinea	POM	AYPY	-9.443380	147.220001	146	
...	...	...	...	...	...	...	...	...	...	...
7693	14106	Rogachyovo Air Base	Belaya	Russia	\N	ULDA	71.616699	52.478298	272	
7694	14107	Ulan-Ude East Airport	Ulan Ude	Russia	\N	XIUW	51.849998	107.737999	1670	
7695	14108	Krechevitsy Air Base	Novgorod	Russia	\N	ULLK	58.625000	31.385000	85	

	Airport ID	Name	City	Country	IATA	ICAO	Latitude	Longitude	Altitude	Tim
7696	14109	Desierto de Atacama Airport	Copiapo	Chile	CPO	SCAT	-27.261200	-70.779198	670	
7697	14110	Melitopol Air Base	Melitopol	Ukraine	\N	UKDM	46.880001	35.305000	0	

7698 rows × 14 columns

Filtrando apenas os aeroportos brasileiros

In [4]:
```
#Pegando apenas os Aeroportos, Brasileiros
aeroport_brasil = df_airports.loc[df_airports['Country'] == 'Brazil']

print(len(aeroport_brasil))
aeroport_brasil
```

264

Out[4]:

	Airport ID	Name	City	Country	IATA	ICAO	Latitude	Longitude	Altitude	T
2392	2518	Conceição do Araguaia Airport	Conceicao Do Araguaia	Brazil	CDJ	SBAA	-8.348350	-49.301498	653	
2393	2519	Campo Délio Jardim de Mattos Airport	Rio De Janeiro	Brazil	\N	SBAF	-22.875099	-43.384701	110	
2394	2520	Amapá Airport	Amapa	Brazil	\N	SBAM	2.077510	-50.858200	45	
2395	2521	Araraquara Airport	Araracuara	Brazil	AQA	SBAQ	-21.812000	-48.132999	2334	
2396	2522	Santa Maria Airport	Aracaju	Brazil	AJU	SBAR	-10.984000	-37.070301	23	
...	...	...	...	...	...	...	...	...	...	...
7643	13723	Augusto Severo Airport	Natal	Brazil	\N	SBNT	-5.911420	-35.247700	169	
7647	13735	Flores Airport	MANAUS	Brazil	\N	SWFN	-3.072778	-60.021111	203	
7659	13772	Fazenda Uiapuru Airport	COMODORO	Brazil	\N	SWVJ	-13.663889	-56.002220	1519	
7670	13830	Fazenda Kajussol Airport	Alta Floresta D'Oeste	Brazil	\N	SJYD	-11.964722	-61.686668	636	
7672	13881	Costa Marques Airport	COSTA MARQUES	Brazil	CQS	SWCQ	-12.421100	-64.251602	555	

264 rows × 14 columns

In [5]:
```
# Lendo os aquivos airlines.dat
df_airlines = pd.read_csv('data/airlines.dat', sep=",",
                  names=["Airline ID", "Name", "Alias", "IATA", "ICAO", "Calls
                         "Country", "Active"])
```

In [6]:
```
df_airlines
```

Out[6]:

	Airline ID	Name	Alias	IATA	ICAO	Callsign	Country	Active
0	-1	Unknown	\N	-	NaN	\N	\N	Y
1	1	Private flight	\N	-	NaN	NaN	NaN	Y
2	2	135 Airways	\N	NaN	GNL	GENERAL	United States	N
3	3	1Time Airline	\N	1T	RNX	NEXTIME	South Africa	Y
4	4	2 Sqn No 1 Elementary Flying Training School	\N	NaN	WYT	NaN	United Kingdom	N
...	...	...	...	...	...	...	...	...
6157	21248	GX Airlines	NaN	NaN	CBG	SPRAY	China	Y
6158	21251	Lynx Aviation (L3/SSX)	NaN	NaN	SSX	Shasta	United States	N
6159	21268	Jetgo Australia	NaN	JG	\N	NaN	Australia	Y
6160	21270	Air Carnival	NaN	2S	\N	NaN	India	Y
6161	21317	Svyaz Rossiya	Russian Commuter	7R	SJM	RussianConnecty	Russia	Y

6162 rows × 8 columns

Filtrando apenas as linhas aéreas brasileiras

In [7]:
```
airlines_brasil = df_airlines.loc[df_airlines['Country'] == 'Brazil']
```

In [8]:
```
print(len(airlines_brasil))
airlines_brasil
```

```
60
```

Out[8]:

	Airline ID	Name	Alias	IATA	ICAO	Callsign	Country	Active
42	42	ABSA - Aerolinhas Brasileiras	\N	M3	TUS	ABSA Cargo	Brazil	Y
43	43	Abaet	\N	NaN	ABJ	Abaet	Brazil	N
51	51	ATA Brasil	\N	NaN	ABZ	ATA-BRAZIL	Brazil	N
226	226	Airvias S/A Linhas Aereas	\N	NaN	AIV	AIRVIAS	Brazil	N
301	301	Air Minas Linhas A	\N	NaN	AMG	AIR MINAS	Brazil	N

	Airline ID	Name	Alias	IATA	ICAO	Callsign	Country	Active
1380	1381	BETA - Brazilian Express Transportes Aereos	\N	NaN	BET	BETA CARGO	Brazil	N
1404	1405	Bringer Air Cargo Taxi Aereo	\N	E6	NaN	NaN	Brazil	N
1468	1469	BRA-Transportes Aereos	\N	7R	BRB	BRA-TRANSPAEREOS	Brazil	N
1475	1476	Brazilian Air Force	\N	NaN	BRS	BRAZILIAN AIR FORCE	Brazil	Y
1485	1486	Brasair Transportes Aereos	\N	NaN	BSI	BRASAIR	Brazil	N
1533	1534	Brazilian Army Aviation	\N	NaN	EXB	BRAZILIAN ARMY	Brazil	N
1548	1549	Brazilian Navy Aviation	\N	NaN	MBR	BRAZILIAN NAVY	Brazil	N
1932	1933	Cruiser Linhas Aereas	\N	NaN	VCR	VOE CRUISER	Brazil	N
2177	2178	Empresa Brasileira De Aeronautica	\N	NaN	EMB	EMBRAER	Brazil	N
2196	2197	Empressa Brasileira de Infra-Estrutura Aeropor...	\N	NaN	XLT	INFRAERO	Brazil	N
2489	2490	GENSA	\N	NaN	GEN	GENSA-BRASIL	Brazil	N
2580	2581	Gol Transportes Aéreos	\N	G3	GLO	GOL TRANSPORTE	Brazil	Y
3417	3421	Mastertop Linhas Aereas	\N	Q4	NaN	NaN	Brazil	N
3448	3452	Mega Linhas Aereas	\N	NaN	MEL	MEGA AIR	Brazil	N
3463	3468	Meta Linhas A	\N	NaN	MSQ	META	Brazil	N
3569	3574	NHT Lineas Aereas	\N	NaN	NHG	HELGA	Brazil	N
3674	3679	Nordeste Linhas Aereas Regionais	\N	JH	NES	NORDESTE	Brazil	N
3759	3764	Oceanair	\N	O6	ONE	OCEANAIR	Brazil	Y
3894	3900	Pantanal Linhas Aéreas	\N	P8	PTN	PANTANAL	Brazil	N
3908	3914	Passaredo Transportes Aereos	\N	NaN	PTB	PASSAREDO	Brazil	Y
3954	3960	Phoenix Air Lines	\N	NaN	PHN	PHOENIX BRASIL	Brazil	N
4078	4084	Puma Linhas Aereas	\N	NaN	PLY	PUMA BRASIL	Brazil	N
4192	4198	Rico Linhas A	\N	C7	RLE	RICO	Brazil	N
4198	4204	Rio Air Express	\N	NaN	SKA	RIO EXPRESS	Brazil	N
4201	4207	Rio Sul Servi	\N	SL	RSL	RIO SUL	Brazil	N

	Airline ID	Name	Alias	IATA	ICAO	Callsign	Country	Active
4229	4235	Rotatur	\N	NaN	RTR	ROTATUR	Brazil	N
4750	4758	Sete Linhas Aereas	\N	NaN	SLX	SETE	Brazil	N
4857	4867	TAM Brazilian Airlines	\N	JJ	TAM	TAM	Brazil	Y
4864	4874	Transbrasil	\N	NaN	TBA	TRANSBRASIL	Brazil	N
4948	4958	TEAM Transportes Aereos	\N	NaN	TIM	TEAM BRASIL	Brazil	N
5046	5056	TAF-Linhas Aereas	\N	NaN	TSD	TAFI	Brazil	N
5069	5079	Total Linhas Aereas	\N	NaN	TTL	TOTAL	Brazil	N
5177	5188	TRIP Linhas A	\N	8R	TIB	TRIP	Brazil	Y
5306	5323	VICA - Viacao Charter Aereos	\N	NaN	VCA	VICA	Brazil	N
5337	5354	Varig Log	\N	LC	VLO	VELOG	Brazil	Y
5351	5368	VRG Linhas Aereas	Varig	RG	VRN	VARIG	Brazil	Y
5356	5373	VASP	\N	VP	VSP	VASP	Brazil	Y
5382	5399	WebJet Linhas A	\N	WJ	WEB	WEB-BRASIL	Brazil	Y
5701	13306	BRAZIL AIR	BRAZIL AIR	GB	BZE	BRAZIL AIR	Brazil	Y
5720	13838	Aero Brazil	NaN	BZ	BZL	NaN	Brazil	N
5726	13983	Azul	Azul Linhas Aéreas Brasileiras	AD	AZU	NaN	Brazil	Y
5755	15985	TrasBrasil	NaN	TB	TBZ	NaN	Brazil	Y
5756	15989	TransBrasil Airlines	NaN	TH	THS	NaN	Brazil	Y
5767	16127	TransHolding	Trans	TI	THI	NaN	Brazil	Y
5776	16150	TransHolding System	NaN	YO	TYS	NaN	Brazil	Y
5780	16234	Fly Brasil	Fly Brasil	F1	FBL	FBL	Brazil	Y
5793	16364	Austral Brasil	Austral Brasil lineas aereas	W7	\N	NaN	Brazil	Y
5801	16487	Cruzeiro do Sul Servicos Aereos	NaN	NaN	CRZ	NaN	Brazil	N
5813	16645	NEXT Brasil	NEXT	XB	NXB	XB	Brazil	Y
5817	16695	GNB Linhas Aereas	NaN	GN	\N	NaN	Brazil	Y
5841	16826	Whitejets	NaN	NaN	WTJ	WHITEJET	Brazil	Y
5851	16900	TROPICAL LINHAS AEREAS	TROPICAL	T1	TP3	NaN	Brazil	N
6040	19812	Voestar	Voestar Brasil	8K	K88	NaN	Brazil	Y
6064	19977	All America BR	All America Brasil	1Y	A9B	NaN	Brazil	Y
6075	20110	FOX Linhas Aereas	NaN	FX	FOX	NaN	Brazil	Y

Lendo os arquivos de rotas routes.dat

In [9]:
```
df_routes = pd.read_csv('data/routes.dat', sep=",",
                        names=["Airline", "Airline ID", "Source airport", "Source ai
                               "Destination airport", "Destination airport ID", "Coc
```

In [10]:
```
df_routes
```

Out[10]:

	Airline	Airline ID	Source airport	Source airport ID	Destination airport	Destination airport ID	Codeshare	Stops	Equipment
0	2B	410	AER	2965	KZN	2990	NaN	0	CR2
1	2B	410	ASF	2966	KZN	2990	NaN	0	CR2
2	2B	410	ASF	2966	MRV	2962	NaN	0	CR2
3	2B	410	CEK	2968	KZN	2990	NaN	0	CR2
4	2B	410	CEK	2968	OVB	4078	NaN	0	CR2
...	...	...	...	...	...	...	...	...	...
67658	ZL	4178	WYA	6334	ADL	3341	NaN	0	SF3
67659	ZM	19016	DME	4029	FRU	2912	NaN	0	734
67660	ZM	19016	FRU	2912	DME	4029	NaN	0	734
67661	ZM	19016	FRU	2912	OSS	2913	NaN	0	734
67662	ZM	19016	OSS	2913	FRU	2912	NaN	0	734

67663 rows × 9 columns

Métodos desenvolvidos para resolução do problema

- 1) Percorrer todas rotas, verificar as rotas com origem e destino em aeroportos brasileiros, salvar a linha aérea que operam em ambos os aeroportos da rota verificada. Com isso é possível ter a linha aérea que opera em cada aeroporto.
- 2) ou seja é criado uma lista com as arestas de cada aeroporto. Para fazer isso a lista de todos e aeroportos foi percorrida e verificado quais linhas aéreas operam em um determinado aeroporto, e todas as rotas que passam no mesmo.
- 3) Foi criada uma função para, dados dois aeroportos a_1 e a_2, retornar as linhas e rotas que passam em ambos os aeroportos a_{l_1}, a_{r_1} e a_{l_2}, a_{r_2}. Com isso é possível calcular a similaridade entre os dois vértices (aeroporto a_1, a_2), pois temos as arestas que ligam os mesmo.
- 4) Com as arestas de cada aeroporto brasileiro (a_l, a_r) é possível calcular a similaridade entre dois areporto qualquer, para isso foi usado o método de Jaccard e similaridade por cosseno.

1) Percorrer todas rotas.

- Verificando quais têm origem e destino em aeroporto brasileiro.
- Salvando a rota e a linha que opera a mesma.

In [11]:
```
# Lista de aeroportos brasileiros para verificar se a rota é BR
lis_id_areo_BR = list(aeroport_brasil["Airport ID"])
lista_rotas_BR = [] # Lista de rotas BR
lista_linha_BR = [] # Lista de linhas arearias e os areportos q a mesma opera

# Percorrendo todas as rotas
for i in tqdm(range(0, 67663)):
    try:
        aero_orige = int(df_routes.iloc[i][3])
        aero_dest = int(df_routes.iloc[i][5])


        # Rota (Aeroporto de origem e destino)
        rota = [int(df_routes.iloc[i][3]), int(df_routes.iloc[i][5])]

        # Linha e a rota que a mesma opera
        linha_aero = [int(df_routes.iloc[i][1]), rota]

        # Verifica se a rota é BR, com origem e destino no brasil
        if aero_orige in lis_id_areo_BR and aero_dest in lis_id_areo_BR:

            lista_rotas_BR.append(rota) # Quardando as rotas BR
            lista_linha_BR.append(linha_aero) # Linhas e as rotas que a mesma

    except ValueError:
        pass
```

```
100%|██████████| 67663/67663 [00:30<00:00, 2216.93it/s]
```

In [12]:
```
# Quantidade de rotas brasileiras
print(len(lista_rotas_BR))
```

```
1186
```

2) Criando os conjuntos de arestas a_l, a_r para todos os aeroportos brasileiros.

In [13]:
```
a_l = [] # Lista com todas as arestas a_l
a_r = [] # Lista com todas as arestas a_r

A = list(aeroport_brasil["Airport ID"]) # Vertices A
L = lista_linha_BR  # Vertices L
R = lista_rotas_BR # Vertice R

# Percorrendo todos o aeroportos brasileiros
for i in tqdm(range(0, 264)):

    a = A[i] # Pegando um aeroporto de cada vez

    # Construindo as aretas {a_l}
    for j in range(0, len(lista_linha_BR)): # Percorrendo as linhas BR

        # Verificando quais linhas operam no aeroporto a_1
        if a in lista_linha_BR[j][1]:
            a_ll = [a , lista_linha_BR[j][0]]
```

```
            # Não pegando a_l repetidos
            # Pois um linha pode operar em varias rotas no mesmo aereporto
               if a_ll not in a_l:

                   a_l.append(a_ll) # Aeroporto e linha (id)

        # Construindo as aretas {a_r}
        for l in range(0, len(lista_rotas_BR)): # Percorrendo todas as rotas BR
            if a in lista_rotas_BR[l]:
                a_r.append([a, lista_rotas_BR[l]])
```

```
100%|██████████| 264/264 [00:00<00:00, 3733.75it/s]
```

In [14]:

```
# Quatidade de arestas a_l e a_r
print(len(a_l))
print(len(a_r))
```

```
326
2372
```

3) Função para retorna as arestas de dois vertices (a_1, a_2)

- Nesse caso retorna as arestas de a_1 e a_2

In [15]:

```
def dadosGrafos(aeroport_A, aeroport_B, a_l, a_r):

    # Pegando as Linhas que passam em A e B
    lista_l_grafo_A = [] # Linhas que operão em A
    lista_l_grafo_B = [] #  Linhas que operão em B
    for i in range(0, len(a_l)):

        #Pegando as linhas que operam em A
        if aeroport_A == a_l[i][0]:
            lista_l_grafo_A.append(a_l[i][1])


        # Pegando as linhas que operam em B
        if aeroport_B == a_l[i][0]:
            lista_l_grafo_B.append(a_l[i][1])


     # Pegando as Rotas que passam em A e B
    lista_r_grafo_A = [] # Rotas que passam em A
    lista_r_grafo_B = [] # Rotas que passam em B
    for j in range(0, len(a_r)):

        # Pegando as rotas que passam em A
        if aeroport_A == a_r[j][0]:
            lista_r_grafo_A.append(a_r[j][1])


        # Pegando as rotas que passam em B
        if aeroport_B == a_r[j][0]:
            lista_r_grafo_B.append(a_r[j][1])

    return [[lista_l_grafo_A, lista_r_grafo_A], [lista_l_grafo_B, lista_r_gra
```

4.1) Similaridade em Grafos com o metodo de Jaccard

$$jacc = \frac{|F(a) \cap F(b)|}{|F(a) \cup F(b)|} = \frac{(caracteristicas\ em\ comum)}{(total\ características\ distintas)}$$

In [16]:
```
# Função que recebe dois vertices (Areporto A e Areporto B) e calcula a simi
def jaccardSim(lista_l_grafo_A, lista_r_grafo_A,lista_l_grafo_B, lista_r_graf

    caract_comum_ar = 0
    caract_comum_al = 0

    # Comparando a quantidade de Linhas que são iguais entre A e B
    try:
        for i in range(0, len(lista_l_grafo_A)):

            for j in range(0, len(lista_l_grafo_B)):

                if lista_l_grafo_A[i] == lista_l_grafo_B[j]:
                    caract_comum_al += 1

       # Comparando a quantidade de Rotas que são iguais entre A e B
        for i in range(0, len(lista_r_grafo_A)):

            for j in range(0, len(lista_r_grafo_B)):

                if lista_r_grafo_A[i] == lista_l_grafo_B[j]:
                    caract_comum_ar += 1

    except IndexError:
        pass
    # Aplicando a formula de Jaccard
    # Todas as caracteristicas
    total_carac = len(lista_l_grafo_A) + len(lista_l_grafo_B) + len(lista_r_g

    if total_carac == 0:
        jacc = 0
    else:
        jacc = np.abs((caract_comum_ar+caract_comum_al) / total_carac)


    return jacc # Valor de similaridade
```

Aplacando jacccarde em todos os pares de aeropostos BR $jaccardSim(A_1A_2, A_3A_4, ...)$

- Para fazer isso é utilizado os id dos aeroportos brasileiros dos arquivos airports.dat, os pares de aeroportos são pegos de dois em dois usando as sequências dos aeroportos brasileiros.

In [17]:
```
# Percorrendo todos o aeroportos brasileiros

lista_result_jacccard = []
# Aplicando no dois primeiros aeroportos
a_1 = A[0]
a_2 = A[1]
lista_dados = dadosGrafos(a_1, a_2, a_l, a_r)
result_sim =  jaccardSim(lista_dados[0][0], lista_dados[0][1], lista_dados[1]
lista_result_jacccard.append([a_1, a_2, result_sim])

for i in tqdm(range(2, 264, 2)): # pegando os id de dois em dois começando do

    a_1 =  A[i]
    a_2 =  A[i+1]
```

```
    lista_dados = dadosGrafos(a_1, a_2, a_l, a_r)
    result_sim =  jaccardSim(lista_dados[0][0], lista_dados[0][1], lista_dadc
    lista_result_jacccard.append([a_1, a_2, result_sim])
```

```
100%|██████████| 131/131 [00:00<00:00, 3240.35it/s]
```

In [18]:

```
# Salvando o arquivo .xlsx
Jaccard_xls = pd.DataFrame(np.array(lista_result_jacccard), columns = ["ID Ae
Jaccard_xls["ID Aeroporto 1"] = Jaccard_xls["ID Aeroporto 1"].astype(np.int64
Jaccard_xls["ID Aeroporto 2"] = Jaccard_xls["ID Aeroporto 2"].astype(np.int64

excel = pd.ExcelWriter('Jaccard.xlsx', engine='xlsxwriter')
Jaccard_xls.to_excel(excel)
excel.save()

Jaccard_xls
```

Out[18]:

	ID Aeroporto 1	ID Aeroporto 2	Similaridade de Jaccard
0	2518	2519	0.000000
1	2520	2521	0.000000
2	2522	2524	0.030303
3	2525	2526	0.013514
4	2527	2528	0.000000
...	...	...	...
127	13636	13643	0.000000
128	13668	13669	0.000000
129	13683	13723	0.000000
130	13735	13772	0.000000
131	13830	13881	0.000000

132 rows × 3 columns

4.2) Similaridade em Grafos com por Cosseno

Para o cálculo, considere a vizinhança em termos de linhas aéreas de cada aeroporto, e como peso da aresta o número de rotas daquela linha aérea que operam naquele aeroporto.

$$Cos(A,B) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum a_i b_i}{\sqrt{a_i^2}\sqrt{b_i^2}}$$

- 1) Calcular os pesos de cada vértices de acordo com as linhas aéreas em comum entre dois aeroportos.
- 2) Aplicando a formula do Cosseno para cada par de aeroporto.

In [19]:

```
def listPesos(lista_linha_BR, id_areport_A, id_areport_B):

    lista_linha_areport_A = []
    lista_linha_areport_B = []

    # Pegando todas as linhas aeriass que operam no aeroporto A
    for i in range(0, len(lista_linha_BR)):
```

```
        if id_areport_A in lista_linha_BR[i][1]:
            lista_linha_areport_A.append(lista_linha_BR[i][0])

    # Pegando todas as linhas aeriass que operam no aeroporto b
    for i in range(0, len(lista_linha_BR)):

        if id_areport_B in lista_linha_BR[i][1]:
            lista_linha_areport_B.append(lista_linha_BR[i][0])


    # Pegando as linhas aerias que operam em ambos os aeroportos
    lista_ambos = []
    for i in range(0, len(lista_linha_areport_A)):

        linha = lista_linha_areport_A[i]

        for j in range(0, len(lista_linha_areport_B)):

            if linha == lista_linha_areport_B[j] and linha not in lista_ambos

                lista_ambos.append(linha)

    lista_peso_A = []
    lista_peso_B = []

    # Contando quantas rotas cada linha areia opera em nos areportos (Peso da
    for i in range(0, len(lista_ambos)):
        linha = lista_ambos[i]
        lista_peso_A.append(lista_linha_areport_A.count(linha))
        lista_peso_B.append(lista_linha_areport_B.count(linha))

    return [lista_peso_A, lista_peso_B] # Retorna os pesos de cada linha aeri
```

In [20]:

```
# Função usada para calcular a similaridade por cosseno
def simCos(lista_peso_A, lista_peso_B):
    sum_a_b = 0
    sum_raiz_a = 0
    sum_raiz_b = 0
    for i in range(len(lista_peso_A)):
        sum_a_b += lista_peso_A[i] * lista_peso_B[i]
        sum_raiz_a += lista_peso_A[i]*lista_peso_A[i]
        sum_raiz_b += lista_peso_B[i]*lista_peso_B[i]


    raiz = (math.sqrt(sum_raiz_a) * math.sqrt(sum_raiz_b))

    if raiz == 0:
        sim_cos = 0

    else:

        sim_cos = sum_a_b / raiz

    return sim_cos
```

Aplacando a similaridade por coasseno em todos os pares de aeropostos BR $jaccardSim(A_1A_2, A_3A_4, ...)$

- Para fazer isso é utilizado os id dos aeroportos brasileiros dos arquivos airports.dat, os pares de aeroportos são pegos de dois em dois usando as sequências dos aeroportos brasileiros.

In [21]:
```
# Percorrendo todos o aeroportos brasileiros

lista_result_cos = []
# Aplicando simililaridade do cosseno no dois primeiros aeroportos
a_1 = A[0]
a_2 = A[1]

pesos_a_b = listPesos(lista_linha_BR, a_1, a_2)
result_sim_cos =  simCos(pesos_a_b[0], pesos_a_b[1])
lista_result_cos.append([a_1, a_2, result_sim_cos])


for i in tqdm(range(2, 264, 2)): # pegando os id de dois em dois começando do

    a_1 =  A[i]
    a_2 =  A[i+1]

    pesos_a_b = listPesos(lista_linha_BR, a_1, a_2)
    result_sim_cos =  simCos(pesos_a_b[0], pesos_a_b[1])
    lista_result_cos.append([a_1, a_2, result_sim_cos])
```

```
100%|██████████| 131/131 [00:00<00:00, 5552.45it/s]
```

In [22]:
```
# Salvando o arquivo .xlsx
Cosseno_xls = pd.DataFrame(np.array(lista_result_cos), columns = ["ID Aeropor
Cosseno_xls["ID Aeroporto 1"] = Cosseno_xls["ID Aeroporto 1"].astype(np.int64
Cosseno_xls["ID Aeroporto 2"] = Cosseno_xls["ID Aeroporto 2"].astype(np.int64

excel = pd.ExcelWriter('Cosseno.xlsx', engine='xlsxwriter')
Cosseno_xls.to_excel(excel)
excel.save()

Cosseno_xls
```

Out[22]:

	ID Aeroporto 1	ID Aeroporto 2	Similaridade por Cosseno
0	2518	2519	0.0
1	2520	2521	0.0
2	2522	2524	1.0
3	2525	2526	1.0
4	2527	2528	0.0
...	...	...	...
127	13636	13643	0.0
128	13668	13669	0.0
129	13683	13723	0.0
130	13735	13772	0.0
131	13830	13881	0.0

132 rows × 3 columns

Ameaças à validade

- acho q tem um erro no Dataset, os id (id, IATA, ICAO) dos aeroportos não são os mesmo nos esquivos routes.dat e airports.dat. Por exemplo, existem id de aeroporto em airports.dat que não existe em routes.dat e vice versa. Com isso não dá pra saber ao certo quais linhas aéreas operam em um determinado aeroporto. Com isso, um dos motivos de ter muitos valores zerados de similaridade em ambos os métodos poderia ser por causa de uma inconsistência dos id de identificação dos aeroportos, pois este muitos aeroportos que não estão no arquivo de rotas.