

Menor caminho entre os principais aeroportos internacionais do mundo

Grafos - Menor Caminho - Algoritmo de Bellman-Ford

Alunos:

Paulo Vitor Alves de Oliveira

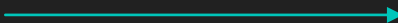
Luis Felipe Araujo Mota

Gabriel Laroche Borba

Contextualização

- Um planejador de voos aéreos estava decidindo otimizar as rotas de voos no mundo. Para isso ele procurou os principais aeroportos internacionais para entender como são suas conexões e a distância entre eles, a fim de otimizar as rotas e proporcionar uma redução de custo e tráfego aéreo. Seria mesmo possível tal fato?
- Suponha uma situação hipotética em que você deseja viajar o mais rápido possível para outro país, mas não conhece as rotas e portanto não terá como saber o meio mais rápido. Ou ainda pior, algumas rotas foram canceladas e apenas poucas sobraram, logo, você deve optar por um caminho mapeado previamente para alcançar seu objetivo. Como resolver?

Base de Dados

- <https://www.flightconnections.com>
- <https://drive.google.com/file/d/1V86ZiVePa0FSoB-TZ38AkVKpgholVRgv/view>
- [Dados Base.txt](#) 
- Base de dados contendo a distância entre dois aeroportos(utilizamos as siglas)
- Exemplo:
GRU ADD 6165
Corresponde à viagem de Guarulhos(GRU) até Addis Ababa(ADD) com uma distância de 6165 milhas.

GRU ADD 6165
GRU BCN 5443
GRU BOG 2692
GRU MEX 4616
GRU MIA 4082
GRU DXB 7587
DXB CCU 2090
DXB SIN 3630
DXB SYD 7480
DXB JNB 3982
JNB CPT 789
JNB WVB 881
JNB LOS 2809
JNB ATL 8435
DXB LAX 8322
MEX ATL 1331
MEX LAX 1552
MEX NRT 6989
NRT SIN 3327
BOG MAD 4988
BOG LAX 3478
BOG MEX 1961
LAX SCL 5581
LAX NRT 5436
CPT EWR 7820
CPT CDG 5814
CDG SCL 7249
CDG NRT 6030

ABV ADD 2150
ABV DXB 3345
ABV DSS 1689
SJD MAD 5974
REC GRU 1304
REC MAD 3905
LPL VIE 919
VIE CPT 5669
VIE ICN 5122
CUN BSB 3669
BSB MCO 3786
BSB SCL 1869
BSB AEP 1445
MCO MEX 1279
MCO YHZ 1497
MCO AUS 991
MCO SFO 2441
SFO SYD 7420
SFO HNL 2394
SFO OGG 2334
OGG ANC 2798
OGG YVR 2671
OGG EWR 4893
ICN ATL 7134
ICN DFW 6823
ICN HNL 4566
BCN HKG 6247
BCN EZE 6509

BCN BOG 5287
CCU DOH 2324
CCU HKG 1623
CCU LHR 4958
LHR GRU 5875
EZE FCO 6927
EZE SCL 706
EZE ATL 5015
YOW PUJ 1894
YOW CUN 1793
PUJ YEG 3393
WVB HLE 1405
FNC JFK 3149
JFK NAS 1098
NAS PTY 1109
EWR LIM 3645
AKL SCL 5995
AKL IAH 7417
AKL NRT 5488
AKL SYD 1341
AKL DXB 8819
AKL YVR 7055
TPE VIE 5569
TPE FCO 5596
TPE HNL 5057
TPE BKK 1545
TPE DLC 959
TPE LHR 6075

IST GRU 6557
IST CCS 6027
IST MIA 5992
IST YVR 5954
IST HND 5563
IST ATH 343
IST LIS 1994
IST WAW 836
BEY KWI 799
BEY AUH 1322
BEY ADD 1728
BEY LOS 2784
BEY MAD 2179
BEY BER 1680
BEY ARN 1954
TNR JNB 1326
TNR CDG 5433
TNR ADD 2007
ADD GRU 6165
ADD IAD 7181
ADD VIE 2996
ADD BKK 4197
ADD ATH 2203
ADD DUB 3953
MED SAW 1277
MED CGK 4969
MED SUB 5381
MED KUL 4384

MED ISB 2083
MED ALA 2471
ALA ICN 2591
ALA JED 2631
ALA NGZ 590
ALA KBP 2195
ALA FRA 3160
FRA ANC 4659
FRA EZE 7143
FRA BGO 720
FRA YHZ 3521
FRA IAD 4068
IAD PHX 1951
IAD BCN 4047
IAD MUC 4249
IAD LIS 3581
ANC PHX 2547
PHX YEG 1374
MEL DEL 6239
DEL YUL 6999
POA PTY 3284
POA LIS 5465
REC BSB 1027
REC MAD 3905
MLE VKO 4076

Informações Coletadas

- Ao todo anotamos 135 conexões realizadas entre 93 aeroportos localizadas no mundo todo.
- A foto ao lado comprova esse quantitativo após colocarmos os dados em uma lista e vermos se existia alguma repetição ou não.
- No final, fizemos o programa dar um output relacionado a cada vértice e seu quantitativo, bem como todas as conexões analisadas no banco de dados.

```
qtd vertices:
```

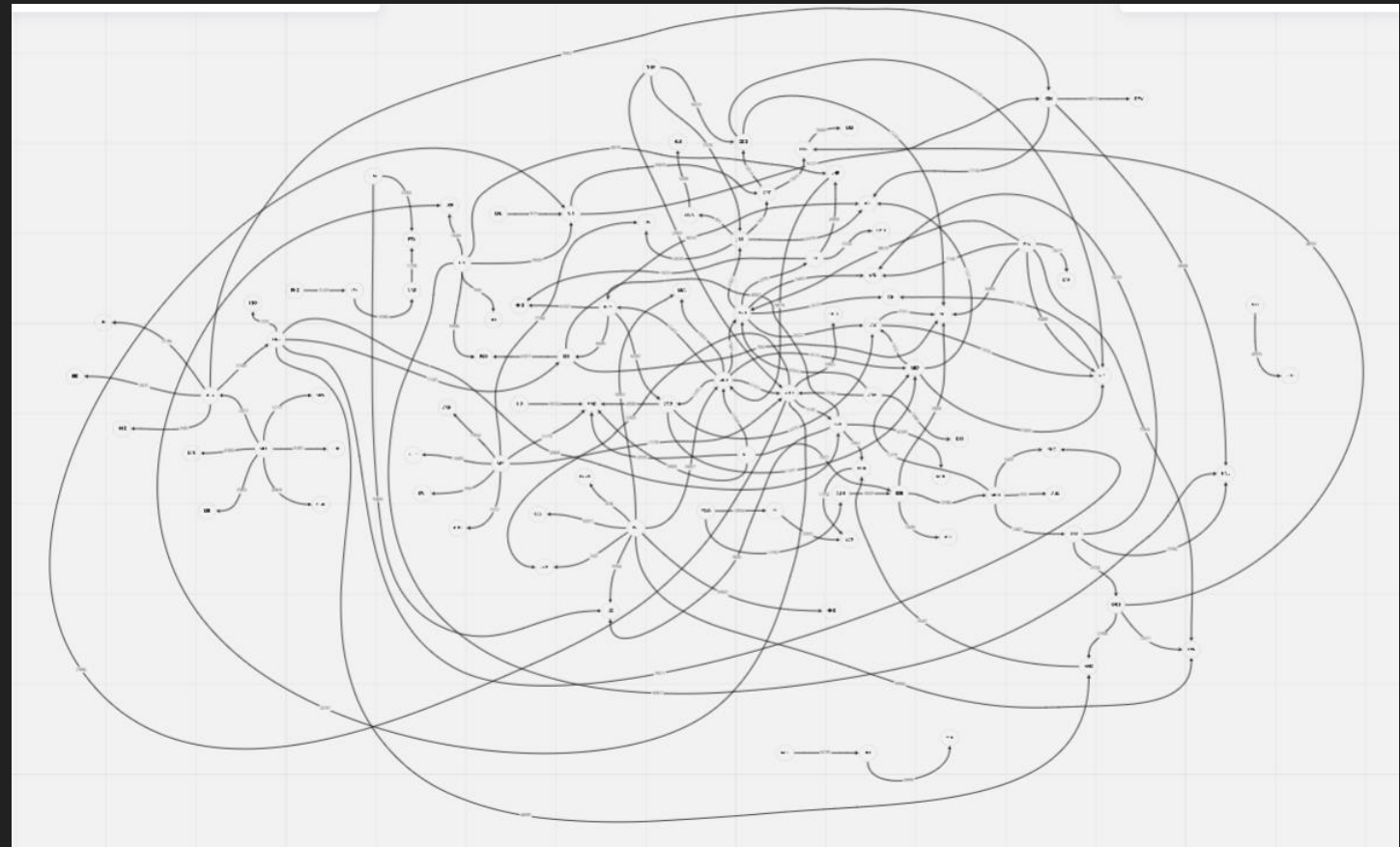
```
93
```

```
qtd conexoes:
```

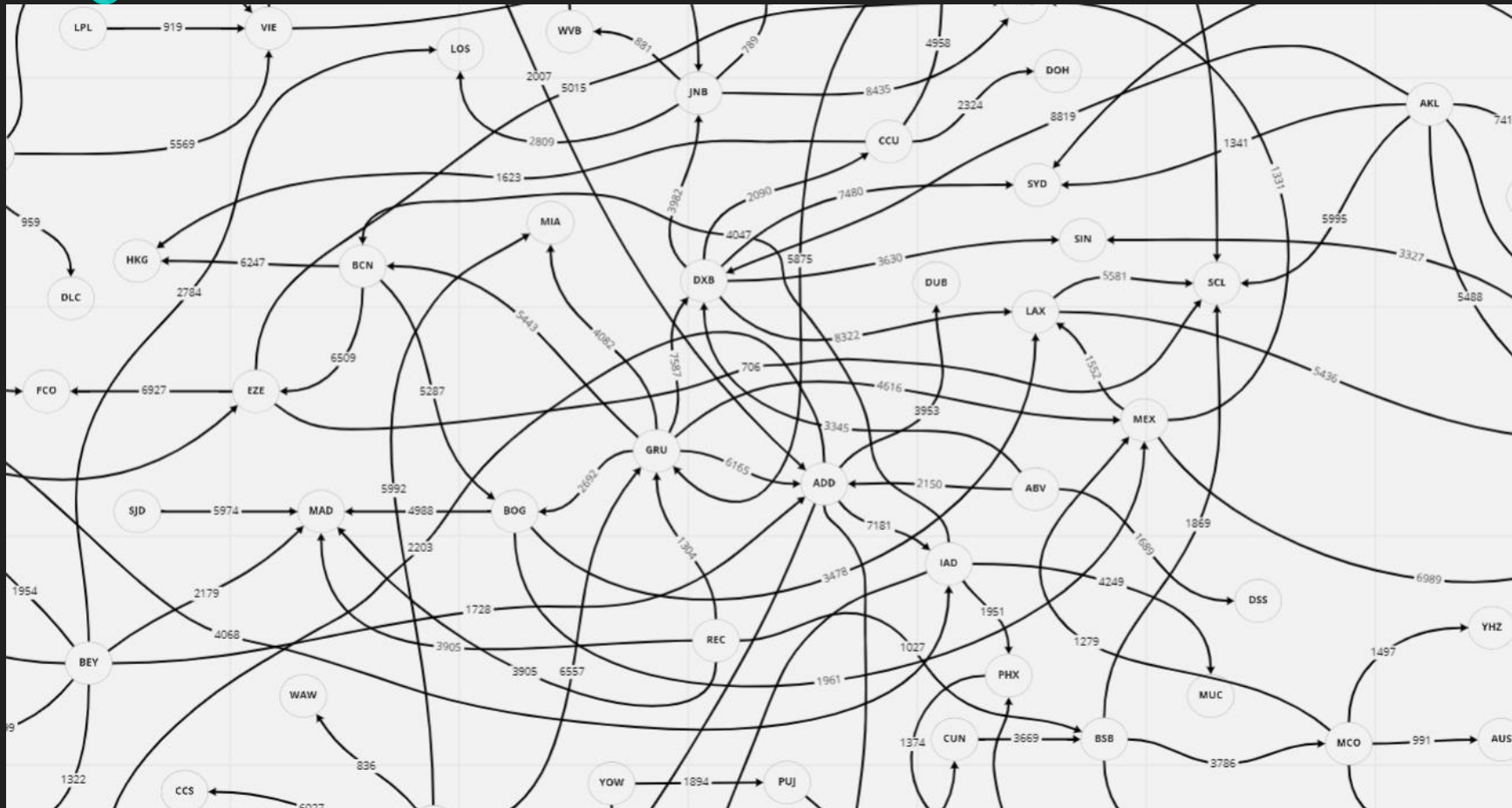
```
135
```

Base de Dados transformado em Grafo

- https://miro.com/app/board/uXjVO80ehqk=
- Utilizamos o miro para analisar toda a base de dados e fazer a construção de um grafo temporário apenas para facilitar a nossa visualização e a explicação no projeto.



Base de Dados transformado em Grafo



Resolução do Problema

- Uma solução simples e eficaz seria utilizar algum programa ou alguma tecnologia que calculasse e nos desse esse caminho e o trajeto a ser realizado. Mas será que existe tal algoritmo?
- A resposta é sim. O problema pode ser solucionado ao utilizar algoritmos bastante conhecidos como o Algoritmo de Dijkstra ou de Bellman-Ford. Optamos pelo de Bellman-Ford e o nosso objetivo consiste em partir de uma origem definida até o destino final. Além disso, estaremos mostrando todo o percurso para alcançar o destino, bem como a distância entre cada ponto.
- Dessa forma, além de facilmente escolher o menor caminho entre um aeroporto e outro baseado de acordo com nossa base de dados, também é possível saber a distância entre cada aeroporto individualmente e a sua soma final para todo o trajeto.

Métodos Utilizados

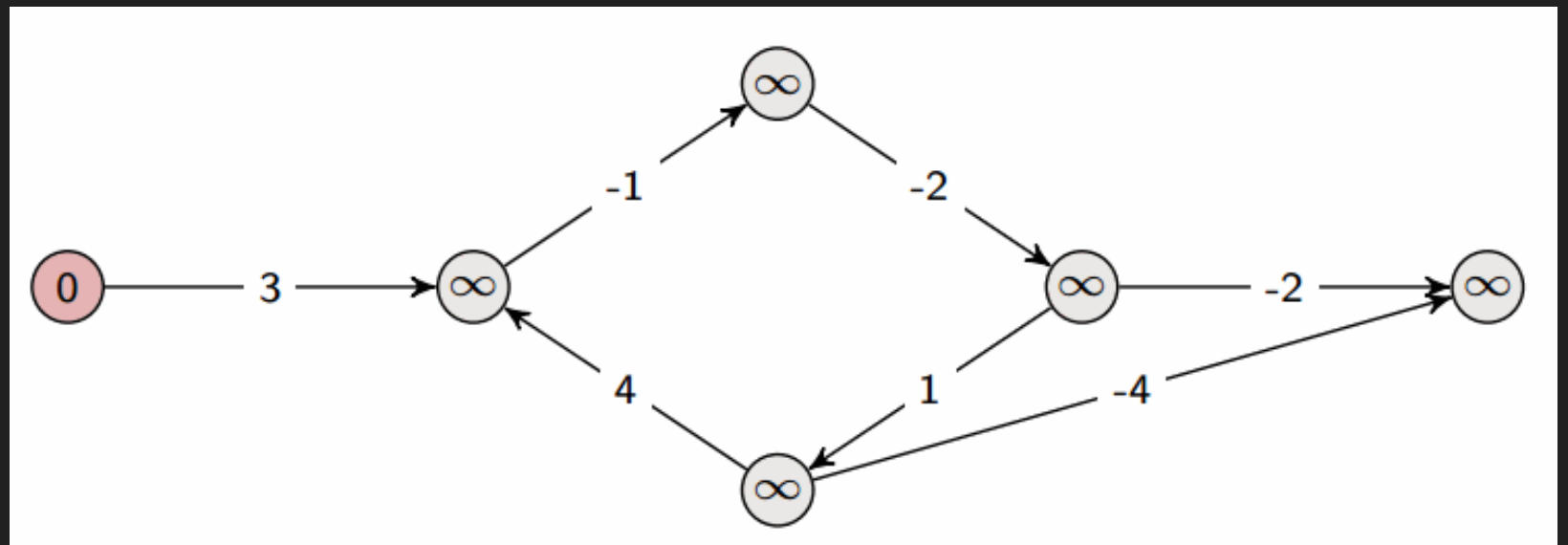
- Para resolver tal problema proposto, utilizaremos o algoritmo de Bellman-Ford, um algoritmo que se assemelha bastante com o algoritmo de Dijkstra, pois os dois têm o mesmo objetivo de encontrar o menor caminho entre os vértices de um grafo direcionado, porém o Bellman-Ford aceita arestas negativas em um grafo, enquanto o Dijkstra aceita somente positivas.
- Utilizando o Bellman-Ford, nós escolhemos um vértice de origem, que em nosso exemplo seria um aeroporto qualquer que compõe a base de dados, e selecionamos arestas aleatoriamente dentro do grafo até que todas as possibilidades tenham sido feitas e nos retorne o grafo com seus menores caminhos para um destino. Logo, por encontrar o menor caminho de forma eficiente, é notório que tal método é suficiente para solucionar o nosso problema proposto anteriormente.
- No final se torna possível obter o menor caminho dentre todas as possibilidades ligando àquele aeroporto passado como origem até outro, com a resposta sendo dada em milhas ou km. Vale ressaltar que escolhemos este dado apenas para facilitar os cálculos, porém poderíamos passar o preço da viagem também ou até a hora. No entanto, ambos são diretamente proporcionais à distância, logo, indo de X até Y com uma distância de 8000 milhas custará um preço razoavelmente alto e um tempo moderado.

Métodos Utilizados

- Um ponto importante a ser ressaltado de Bellman-Ford é que ele percorre todas as arestas $|V| - 1$ vezes, isto é, parte de um vértice de origem e relaxa todas as arestas ligadas à ele. Depois partimos para outro vértice e relaxamos todas as arestas conectadas a ele. Tudo isso em um loop até que seja percorrido $|V| - 1$.
- A técnica de relaxar é basicamente escolher o menor caminho (menor custo) dado dois caminhos até um mesmo vértice.
- Após esse loop, o programa faz com que ele entre em outra repetição apenas para checar se existe arestas negativas ou não. No entanto, como estamos tratando da distância entre um aeroporto e outro nunca teremos distância negativa, porém é importante ter essa etapa em seu programa, uma vez que é a definição do próprio algoritmo de Bellman-Ford.
- Por fim, fizemos uma interface gráfica e um visualizador de grafos unificando todos em um único programa. Mas o ponto principal é apenas demonstrar como encontrar o menor caminho com o algoritmo escolhido.

Métodos Utilizados

- Este Gif ajudará a exemplificar como o Algoritmo realmente funciona.



Implementação

- Como escolhemos o Algoritmo de Bellman-Ford criamos um programa que partirá de uma origem definida até seu destino relaxando todas as arestas existentes na base de dados. Além disso, criamos uma interface gráfica e um visualizador de grafos, unificando todos eles juntamente do próprio Algoritmo de Bellman-Ford.
- As imagens a seguir mostrarão o código completo:

Implementação

```
1 import PySimpleGUI as sg
2 import networkx as nx
3 import matplotlib.pyplot as plt
4
5
6 def VisualizarGrafo(grafo):
7     G = nx.DiGraph()
8     G.add_edges_from(grafo)
9     nx.draw_networkx(G)
10    plt.show()
11
12
13 class Vertices:
14     def __init__(self):
15         self.aeroportos = []
16         self.caminhos = []
17
18
19 def relax(p, u, v, w, pred):
20     if p[v] > p[u] + w:
21         pred[v] = u
22         p[v] = p[u] + w
23
24
25 def caminho(antecessor, peso, u, v, km, milhas):
26     listaresult = [v]
27     ant = antecessor[v]
28
29     pesosresult = [str(peso[v])]
30
31     while ant != -1:
32         listaresult.append(ant)
33         pesosresult.append(str(peso[ant]))
34         ant = antecessor[ant]
35
36     listaresult = listaresult[::-1]
37     pesosresult = pesosresult[::-1]
38     listaresultstring = []
39     stringresult = " --> ".join(listaresult)
40     stringpesosresult = " --> ".join(pesosresult)
41
```

```
41
42 if len(listaresult) == 1:
43     texto = f"Infelizmente não há rotas entre:\nOrigem: {u}\nDestino: {v}"
44     sg.popup("RESULTADO:", texto)
45
46 else:
47     if milhas == True and km == False:
48         for i in range(len(listaresult) - 1):
49             stringatual = f"{listaresult[i]} --> {listaresult[i + 1]} : Distância = {(((int(pesosresult[i + 1]) - int(pesosresult[i]))):.1f) Milhas "
50             listaresultstring.append(stringatual)
51
52         stringatual = f"\nDistância Total: {pesosresult[-1]} Milhas"
53         listaresultstring.append(stringatual)
54
55     elif milhas == False and km == True:
56         for i in range(len(listaresult) - 1):
57             stringatual = f"{listaresult[i]} --> {listaresult[i + 1]} : Distância = {(((int(pesosresult[i + 1]) - int(pesosresult[i])) * 1.60934):.1f) km "
58             listaresultstring.append(stringatual)
59
60         stringatual = f"\nDistância Total: {(float(pesosresult[-1]) * 1.60934):.1f} km"
61         listaresultstring.append(stringatual)
62
63     else:
64         for i in range(len(listaresult) - 1):
65             stringatual = f"{listaresult[i]} --> {listaresult[i + 1]} : Distância = {(((int(pesosresult[i + 1]) - int(pesosresult[i])) * 1.60934):.1f) km ou {(((int(pesosresult[i + 1]) - int(pesosresult[i]))):.1f) Milhas"
66             listaresultstring.append(stringatual)
67
68         stringatual = f"\nDistância Total: {(float(pesosresult[-1]) * 1.60934):.1f} km ou {pesosresult[-1]} Milhas"
69         listaresultstring.append(stringatual)
70
71     listaresultstring.append(f"\nRota total:\n{stringresult}")
72     stringfinal = "\n".join(listaresultstring)
73
74     sg.popup("MENOR ROTA:", stringfinal)
75
76
```

Implementação

```
75 />
76
77 def bellmanford(origin, graph, destiny, km, milhas):
78     pred = {}
79     p = {}
80     for i in graph.aeroportos:
81         p[i] = 999999999
82         pred[i] = -1
83
84     p[origin] = 0
85
86     for i in range(len(graph.aeroportos) - 1):
87         for u, v, w in graph.caminhos:
88             relax(p, u, v, w, pred)
89
90     for u, v, w in graph.caminhos:
91         if p[v] > p[u] + w:
92             print('Ciclo negativo encontrado!')
93             return False
94
95     caminho(pred, p, origin, destiny, km, milhas)
96
97     return True
98
99
100 class TelaPython:
101     def __init__(self):
102
103         sg.theme("Black")
104         layout = [
105
106             [sg.Text("Origem:", size=(10, 0)), sg.Input(size=(30, 0), key="origem")],
107             [sg.Text("Destino:", size=(10, 0)), sg.Input(size=(30, 0), key="destino")],
108             [sg.Text("Escolha a(s) medidas de conversão: ")],
109             [sg.Checkbox("Km", key="conversaokm"), sg.Checkbox("Milhas", key="conversaomilhas")],
110             [sg.Button("Calcular rota"), sg.Button("Lista de aeroportos disponiveis"),
111              sg.Button("Visualizar Grafo")]
112
113         ]
114
115         self.janela = sg.Window("Menores Rotas Aeroportos", layout)
116
```

```
116
117 def Iniciar(self):
118     while True:
119         self.button, self.values = self.janela.Read()
120         origem = self.values["origem"]
121         destino = self.values["destino"]
122         km = self.values["conversaokm"]
123         milhas = self.values["conversaomilhas"]
124
125         if self.button == sg.WINDOW_CLOSED:
126             break
127
128         if self.button == "Calcular rota":
129
130             if origem == "" or destino == "":
131                 text = f"Input vazio!"
132                 sg.popup("ERRO", text)
133
134             elif origem in v.aeroportos and destino in v.aeroportos:
135
136                 if milhas == False and km == False:
137                     text = f"Escolha uma das opções de medidas de distância!"
138                     sg.popup("ERRO", text)
139                 else:
140                     bellmanford(origem, v, destino, km, milhas)
141
142                 text = f"Origem: {origem}\nDestino: {destino}\n\n Não há caminho direto entre eles!"
143
144             elif origem in v.aeroportos and destino not in v.aeroportos:
145                 text = f"{destino} não está na base de dados!\nverifique a lista de aeroportos disponiveis! "
146                 sg.popup("ERRO", text)
147
148             elif origem not in v.aeroportos and destino in v.aeroportos:
149                 text = f"{origem} não está na base de dados!\nverifique a lista de aeroportos disponiveis! "
150                 sg.popup("ERRO", text)
151
152             else:
153                 text = f"{origem} e {destino} não estão na base de dados!\nverifique a lista de aeroportos disponiveis! "
154                 sg.popup("ERRO", text)
155
156         if self.button == "Lista de aeroportos disponiveis":
157             text = "\t".join(v.aeroportos)
158
159             sg.popup("Lista de aeroportos disponiveis", text)
160
161         if self.button == "Visualizar Grafo":
162             VisualizarGrafo(grafo)
163
164
165
```


Implementação

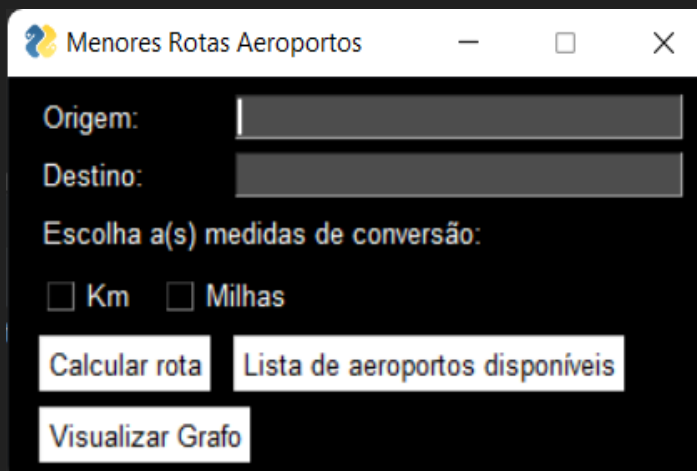
```
163
164
165 v = Vertices()
166 tela = TelaPython()
167 grafo = []
168
169 try:
170     arquivodistancias = open("Dados.txt", "r")
171
172     with arquivodistancias:
173         for line in arquivodistancias:
174             o, d, w = line.split()
175
176             if o not in v.aeroportos:
177                 v.aeroportos.append(o)
178             if d not in v.aeroportos:
179                 v.aeroportos.append(d)
180
181             if [o, d, w] not in v.caminhos:
182                 v.caminhos.append([o, d, int(w)])
183             else:
184                 pass
185
186             grafo.append([o, d])
187
188     print("qtd vertices:")
189     print(len(v.aeroportos))
190     print("qtd conexoes:")
191     print(len(v.caminhos))
192
193 except FileNotFoundError as msg:
194     print(msg)
195
196 tela.Iniciar()
197
```

Bibliotecas Utilizadas

- As bibliotecas que utilizamos foram:
 - PySimpleGUI
 - Networkx
 - Matplotlib
- PySimpleGUI → Interface Gráfica
- Networkx → Visualizador de Grafos
- Matplotlib → Plotar o Visualizador de Grafos

Interface Gráfica

- Nossa Interface Gráfica é capaz de calcular a rota mostrando todos os caminhos percorrido até o destino, além de mostrar todos os aeroportos disponíveis (quantidade de vértices).
- Por fim, adicionamos a opção de Visualizar Grafo que mostra exatamente como está sendo feita todas as conexões caso tenha alguma dúvida ou não saiba para onde ir.



Menores Rotas Aeroportos

Origem:

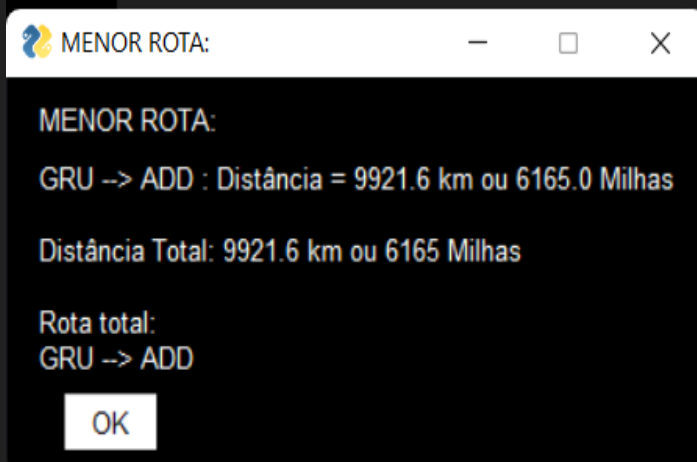
Destino:

Escolha a(s) medidas de conversão:

☐ Km ☐ Milhas

Calcular rota Lista de aeroportos disponíveis

Visualizar Grafo



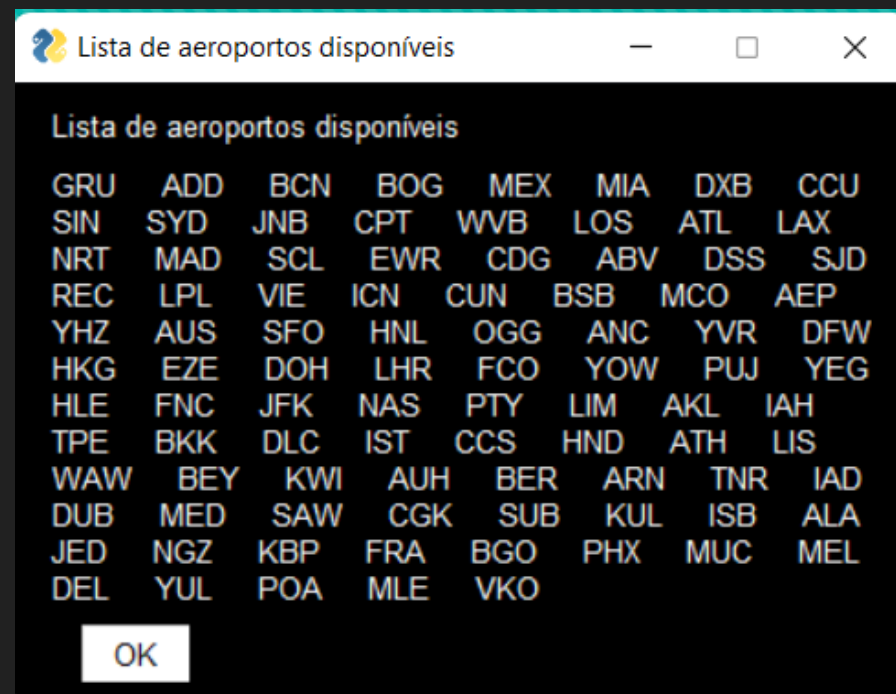
MENOR ROTA:

GRU --> ADD : Distância = 9921.6 km ou 6165.0 Milhas

Distância Total: 9921.6 km ou 6165 Milhas

Rota total:
GRU --> ADD

OK



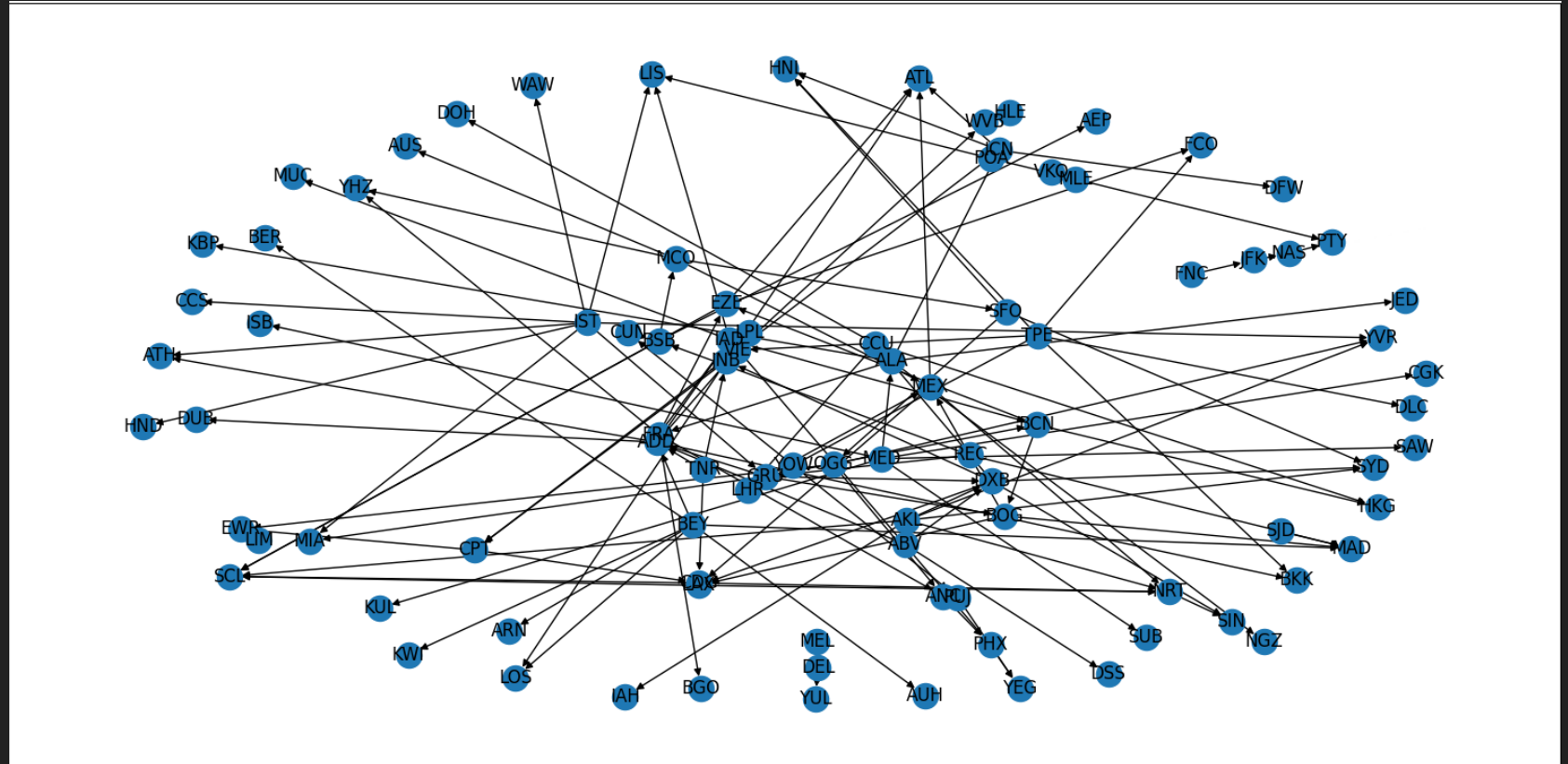
Lista de aeroportos disponíveis

GRU	ADD	BCN	BOG	MEX	MIA	DXB	CCU
SIN	SYD	JNB	CPT	WVB	LOS	ATL	LAX
NRT	MAD	SCL	EWR	CDG	ABV	DSS	SJD
REC	LPL	VIE	ICN	CUN	BSB	MCO	AEP
YHZ	AUS	SFO	HNL	OGG	ANC	YVR	DFW
HKG	EZE	DOH	LHR	FCO	YOW	PUJ	YEG
HLE	FNC	JFK	NAS	PTY	LIM	AKL	IAH
TPE	BKK	DLC	IST	CCS	HND	ATH	LIS
WAW	BEY	KWI	AUH	BER	ARN	TNR	IAD
DUB	MED	SAW	CGK	SUB	KUL	ISB	ALA
JED	NGZ	KBP	FRA	BGO	PHX	MUC	MEL
DEL	YUL	POA	MLE	VKO			

OK

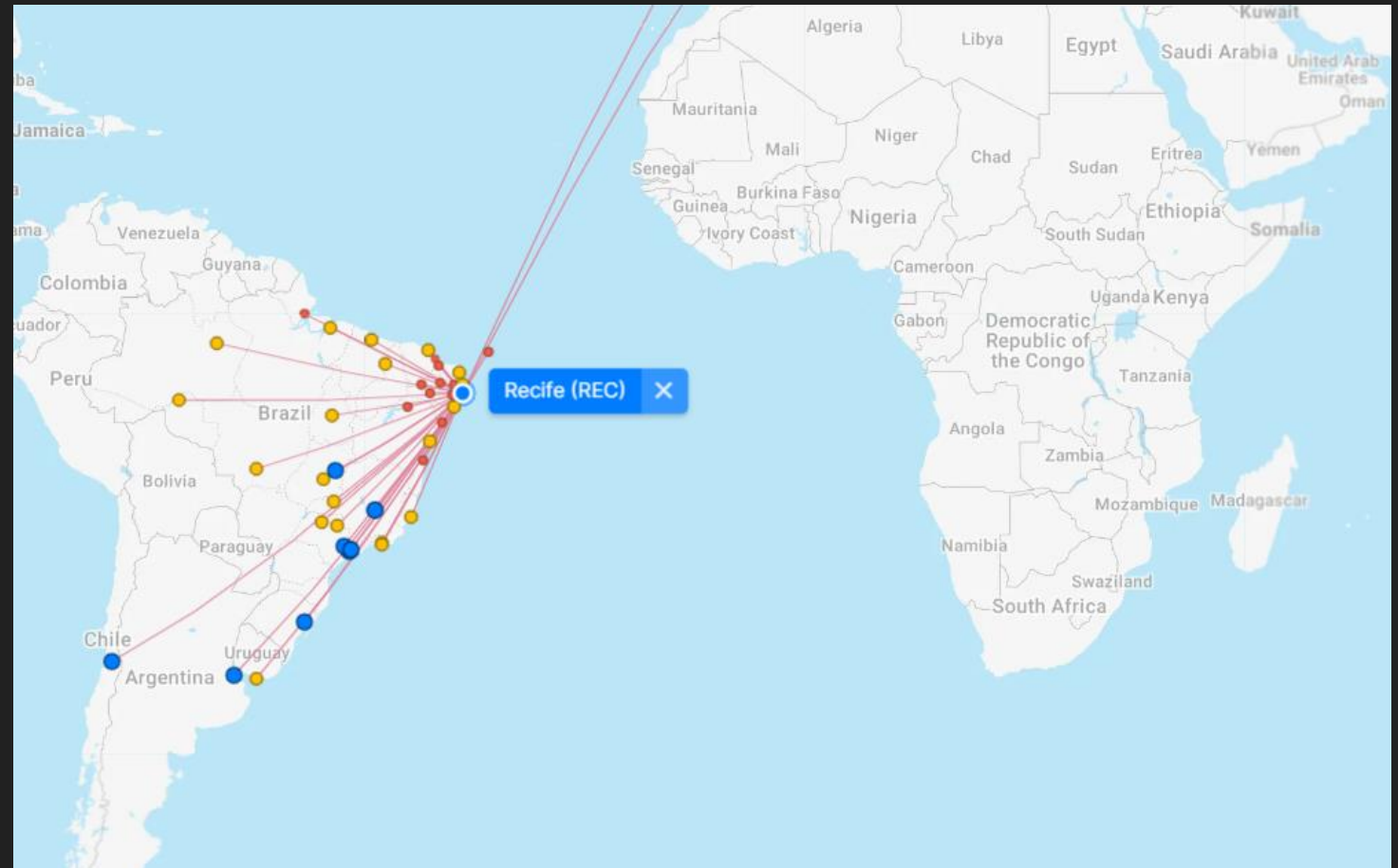
Visualizador de Grafos

- O visualizador de grafos consegue representar com precisão todas as conexões feitas como se fosse exatamente um grafo direcionado que é o nosso objetivo. Além disso, as informações batem com o grafo provisório que fizemos no Miro mostrado em slides anteriores.



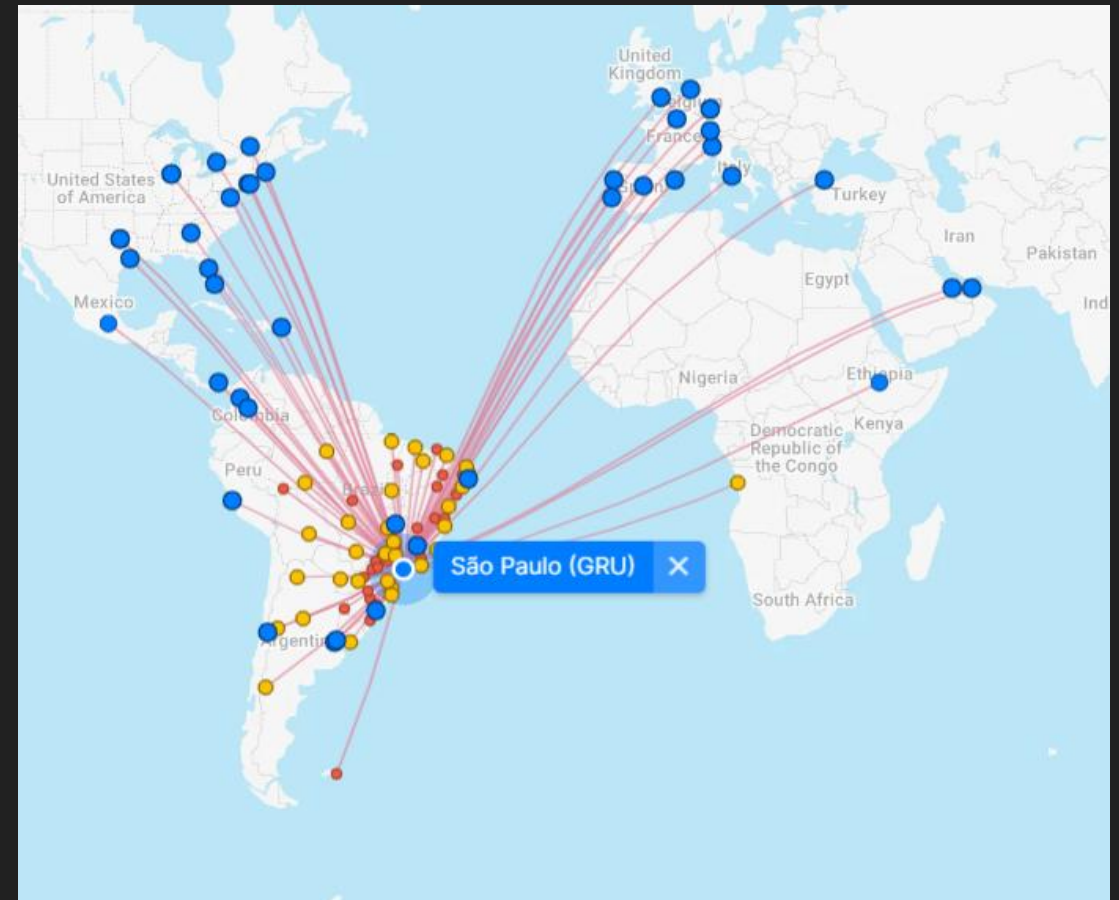
Resultados Encontrados

- Como alcançar Walvis Bay à partir de Recife? É impossível? Não. É simples, porém demorado. Embora não possamos alcançar a Namíbia do outro lado diretamente, podemos realizar conexões entre outros aeroportos a fim de chegar lá.
- Mas quais caminhos? É justamente esse o problema que queremos mostrar e demonstrar. O Bellman-Ford nos faz percorrer todas as 135 arestas 92 vezes, uma vez que nossa base de dados possui 93 vértices.



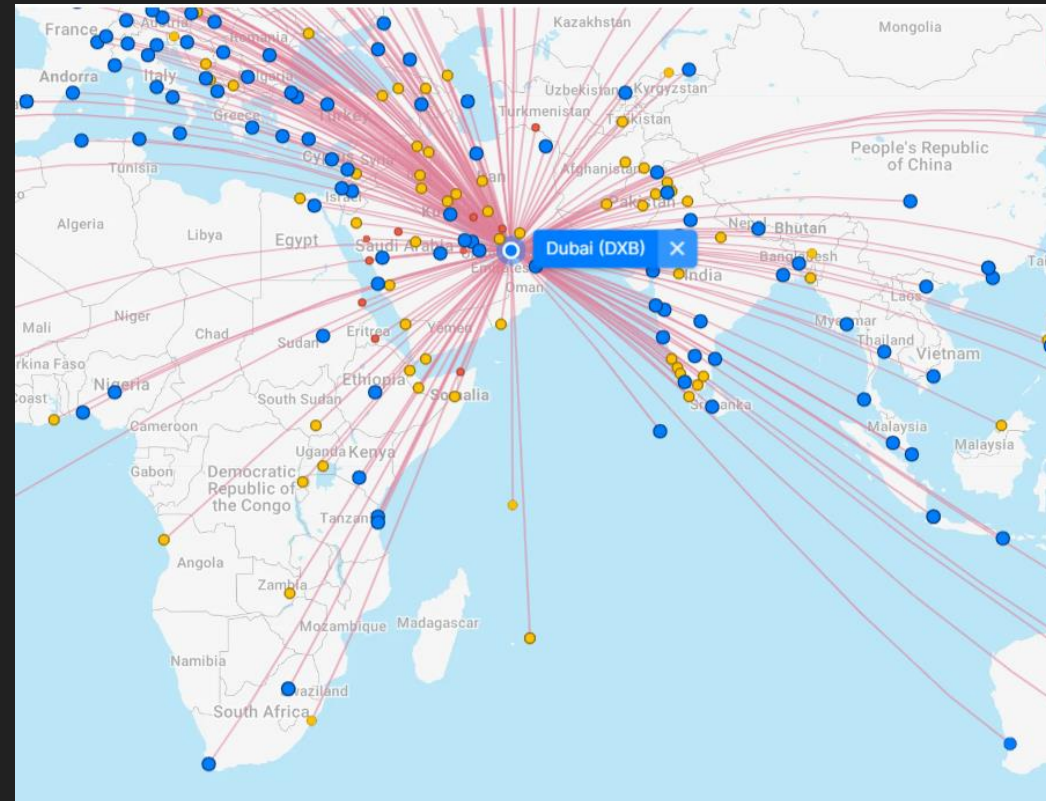
Resultados Encontrados

- No final de tudo, quando nosso laço de repetição tiver acabado e todos os dados forem alocados com sucesso em nossa lista de antecessores e de peso vamos ter informações suficientes para traçar a rota.
- Assim, para pegarmos o menor caminho basta seguirmos do destino e ir pegando os antecessores até chegar na origem com $\text{antecessor} = -1$.



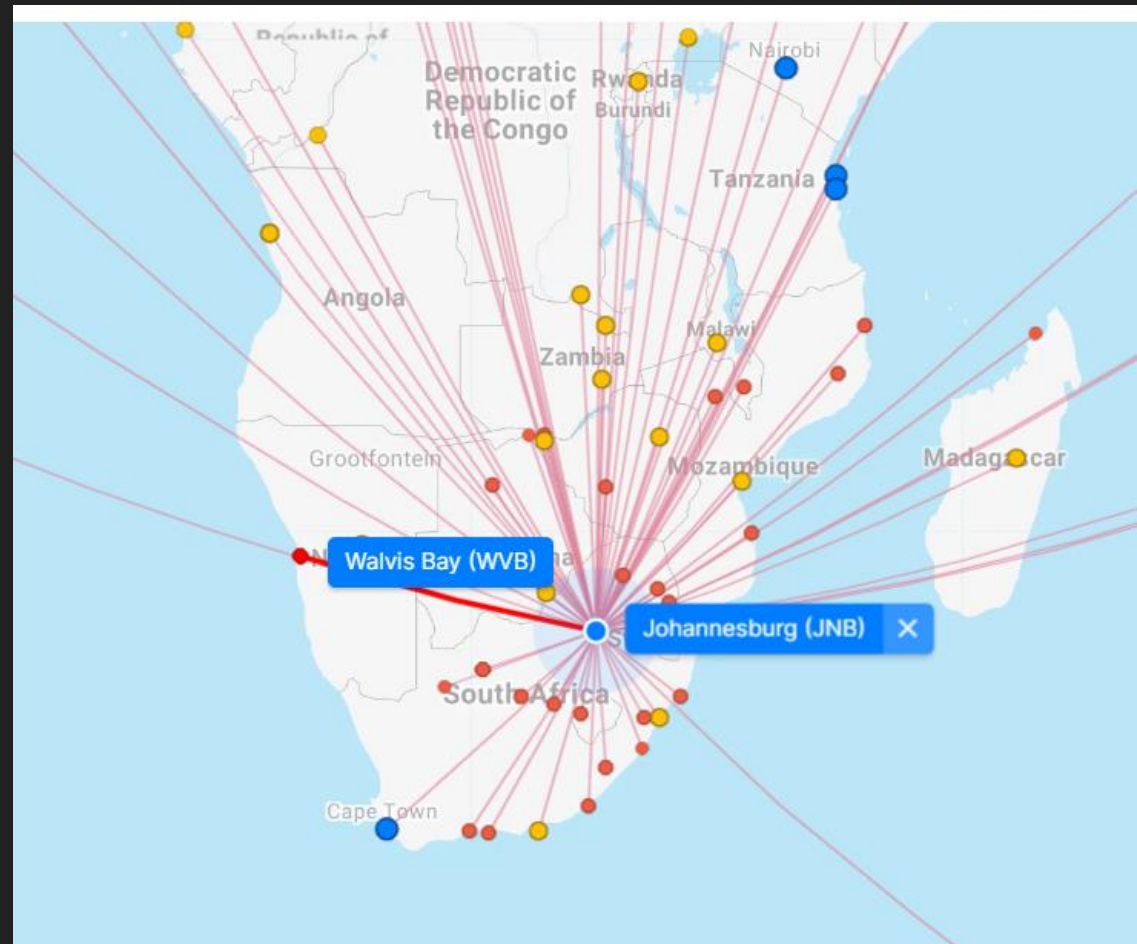
Resultados Encontrados

- As fotos demonstram exatamente qual o menor caminho que se seguiu de Recife até Dubai como demonstrado ao lado.
- Tudo isso só é possível devido às informações coletadas pelo Algoritmo, portanto, bastou processá-las para conseguir alcançar nosso objetivo.



Resultados Encontrados

- Assim, finalmente alcançamos nosso objetivo chegando em Johannesburg e logo depois Walvis Bay. Portanto, nosso caminho ficaria: {Recife, Guarulhos, Dubai, Johannesburg, Walvis Bay}
- Vale ressaltar que essa já é a resposta do algoritmo de Bellman-Ford com o menor caminho entre Recife e Walvis Bay. Mas no geral o algoritmo irá percorrer todas as arestas de todos os vértices até mapear os antecessores e os menores pesos(distância) de cada um. Só no final teríamos de fato o menor caminho.
- No entanto, mapear o caminho de 93 vértices e 135 arestas seria um pouco trabalhoso não é mesmo? Então deixamos isso para o código.



Provando a Resposta Encontrada

- Para confirmar que este é de fato o menor caminho a ser seguido utilizamos nosso próprio programa para comprovar e veja o que encontramos:

Menores Rotas A...

Origem: REC

Destino: WVB

Escolha a(s) medidas de conversão:

☐ Km ☒ Milhas

Calcular rota Lista de aeroportos disponíveis

Visualizar Grafo

MENOR...

MENOR ROTA:

REC --> GRU : Distância = 1304.0 Milhas
GRU --> DXB : Distância = 7587.0 Milhas
DXB --> JNB : Distância = 3982.0 Milhas
JNB --> WVB : Distância = 881.0 Milhas

Distância Total: 13754 Milhas

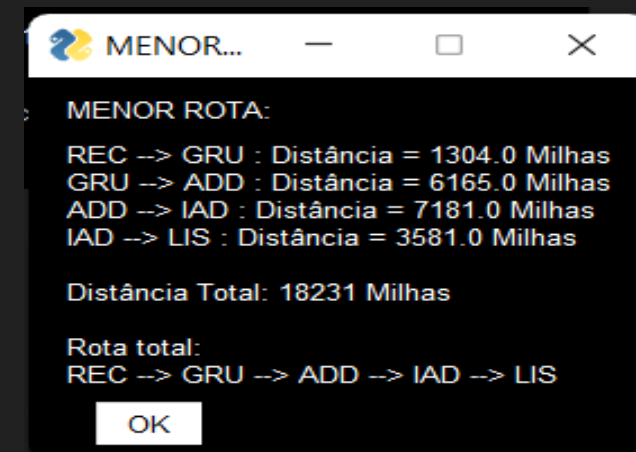
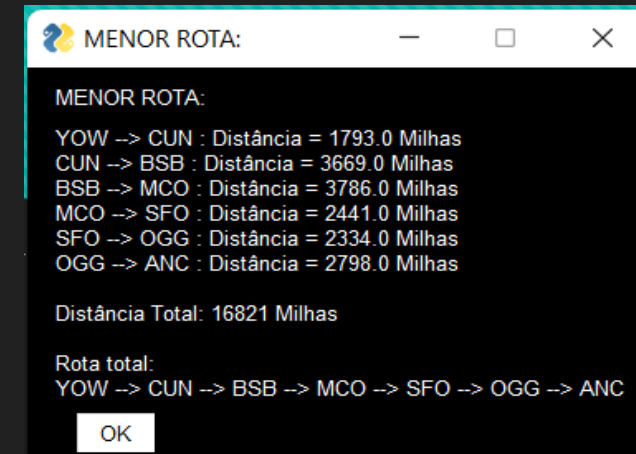
Rota total:
REC --> GRU --> DXB --> JNB --> WVB

OK

Resultados Encontrados

Agora pense em outros exemplos...

- Como chegar em Anchorage(ANC) a partir de Ottawa(YOW)?
- Como chegar em Lisboa(LIS) a partir de Recife(REC)?
- Vale ressaltar que estes são os menores caminhos encontrados de acordo com a base de dados fornecida. Quanto mais informações estiverem lá mais oportunidades irão aparecer e os caminhos podem mudar.



Conclusão

- Finalizando o projeto podemos constatar o sucesso do algoritmo implantado, bem como a resolução do problema proposto. Vale ressaltar que a complexidade do algoritmo é $O(|V| \times |E|)$, uma complexidade relativamente boa, que nos traz um bom resultado se implementada no exercício proposto.
- Poderíamos tentar aprimorar o algoritmo para mostrar não só o menor caminho de um aeroporto à outro, mas também mostrar todos os menores caminhos possíveis existentes entre todos os aeroportos! Mas é claro, levaria mais tempo e seria um pouco mais complicado. Nosso objetivo é apenas demonstrar como funciona o Algoritmo de Bellman-Ford e sua aplicação no dia a dia podendo ser de extrema importância.

Referências

- <https://www.flightconnections.com>
- <http://www.distanciascidades.com/pesquisa/>
- https://distancecalculator.globefeed.com/Distance_Between_Countries.asp
- <https://www.youtube.com/watch?v=Et0fYeA2XxY>