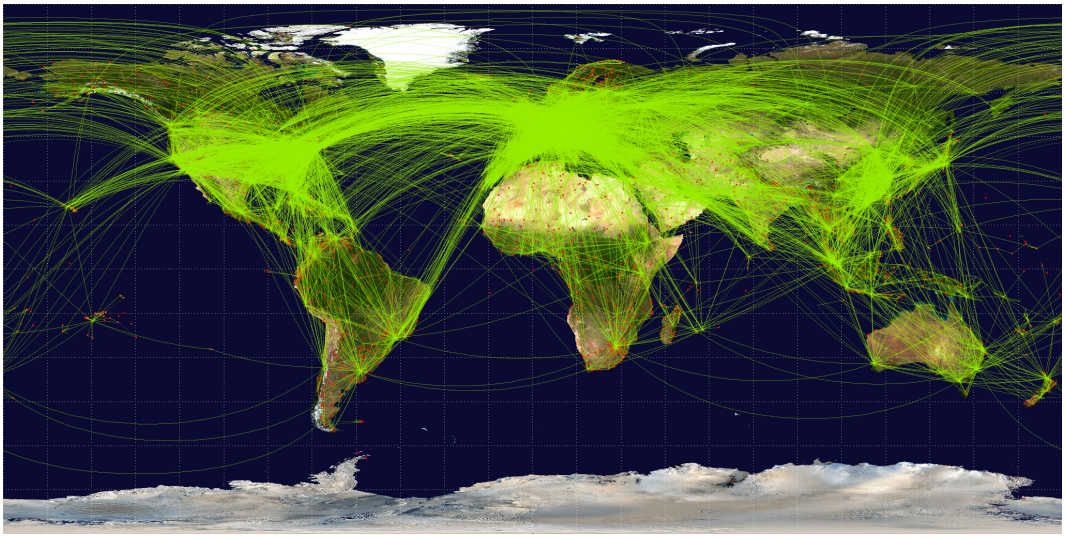
بهترین پرواز

الگوریتم دایجسترا (Dijkstra)



سیدمحمدحسین هاشمی ۴۰۲۲۳۶۳۱۴۳
سیدحسین حسینی ۴۰۱۲۳۶۳۲۳۸

آبان ۱۴۰۲

فهرست مطالب

۱ پکیج های استفاده شده

BestFlights.py

```
import heapq
from pandas import read_csv
from time import time
```

- heapq: در تابع ShortestPath از این پکیج استفاده شده است.
- pandas: برای ایجاد dataframe .و تقسیم بندی داده ها مورد استفاده قرار می‌گیرد
- time: برای محاسبه زمان اجرای الگوریتم

۲ آماده سازی برای شروع

BestFlights.py

```
trip = input('Input: ').split(' - ')

start_time = time()

df = read_csv('../Datasets/Flight_Data.csv')
```

گرفتن ورودی، ساخت دیتافریم و ذخیره زمان شروع اجراالگوریتم

۳ محاسبه وزن الگوریتم

۱.۳ ضرایب

BestFlights.py

```
Time = 0

Distance = 1

Price = 0
```

سه مقدار: زمان، مسافت و هزینه معیار های قابل امتیاز دهی در بین داده‌هاست که در این قسمت ضریب هرکدام دریافت و از این مقدار ها در ۲.۳ استفاده خواهد شد.

۲.۳ محاسبه

BestFlights.py

```
def calculate(distance, flyTime, price):
    return round(Time * flyTime + Distance * distance + Price * price)
```

با استفاده از ضرایب دریافتی از ۱.۳ وزن مسیرها را محاسبه و برمی‌گردانیم.این تابع در کلاس Edge استفاده می‌شود.

۴ کلاس Node

BestFlights.py

```
class Node:
    def __init__(self, Airport, Airport_Country, Airport_City):
        # Source data
        self.Airport = Airport
        self.Country = Airport_Country
        self.City = Airport_City

    def __str__(self):
        return self.City + "-" + self.Airport + ", " + self.Country
```

این کلاس برای ذخیره هر گره (فرودگاه)استفاده می‌شود. مقادیر Airport، Airport-Country و Ariport-City را دریافت و ذخیره می‌کند و همچنین از متد –str– برای ایجاد خروجی استفاده می‌شود.

۵ کلاس Edge

BestFlights.py

```
class Edge:
    def __init__(self, Source, Destination, distance, flyTime, price):
        self.price = round(price, 2)
        self.flyTime = round(flyTime, 2)
        self.distance = round(distance, 2)
        self.destination = Destination
        self.source = Source
        self.weight = calculate(distance, flyTime, price)

    def __str__(self):
        return "From: " + str(self.source) + "\nTo: " + str(self.destination) + "\nDuration: " + str(
            self.distance) + "km\nTime: " + str(self.flyTime) + "h\nPrice: " + str(self.price) + "$"

    def __gt__(self, other):
        return self.weight > other.weight
```

از این کلاس برای ذخیره یک پرواز (مسیر) استفاده می‌کنیم. این کلاس موقع ایجاد کلاس‌های مبدا و مقصد و ویژگی‌های هر پرواز شامل قیمت، مدت پرواز و مسافت را می‌گیرد. تابع calculate در این قسمت استفاده می‌شود تا وزن مسیر را حساب کند.
متد –str– خروجی نهایی هر پرواز را تولید می‌کند و متد –gt– برای استفاده در تابع ShortestPath ایجاد شده و برای مرتب سازی مسیر ها بر اساس کمترین وزن استفاده می‌شود.

۶ لیست یکتا فرودگاه‌ها

BestFlights.py

```
nodes = set()
for data in df.values:
    nodes.add(data[3] + "," + data[1] + "," + data[4])
    nodes.add(data[8] + "," + data[2] + "," + data[9])
```

در این قسمت از پروژه فرودگاه‌ها را به صورت یکتا (Unique) ذخیره می‌کنیم. (این قسمت برای ایجاد لیستی از کلاس های Node) ضروری است.

۷ لیست کلاس فرودگاه‌ها (Node class)

BestFlights.py

```
NodeClass = []
for i in nodes:
    data = i.split(',')
    NodeClass.append(Node(data[1], data[2], data[0]))
```

با استفاده از قسمت ۶ لیستی از کلاس‌های Node می‌سازیم که برای پردازش خروجی مورد استفاده قرار می‌گیرد.

۸ لیست پروازها

BestFlights.py

```
nodes = list(nodes)
edges = []
for i in df.values:
    source = nodes.index(i[3] + "," + i[1] + "," + i[4])
    destination = nodes.index(i[8] + "," + i[2] + "," + i[9])
    edges.append([nodes.index(i[3] + "," + i[1] + "," + i[4]), nodes.index(i[8] + "," + i[2] + "," + i[9]),
                  Edge(NodeClass[source], NodeClass[destination], i[13], i[14], i[15])])
```

با استفاده از ۶ و ۷ لیستی از مسیر ها می‌سازیم که از مقادیر مقصد (nodes index)، مبدا (nodes index) و کلاس قسمت ۵ تشکیل می‌شود. این لیست ورودی تابع ۲.۳ می‌باشد.

۹ محاسبه بهترین مسیر

BestFlights.py

```python
def ShortestPath(N, Edges, Src, Dis):
    adj = {}
    for j in range(N):
        adj[j] = []

    for s, d, edge in Edges:
        adj[s].append([d, edge.weight, edge])

    shortest = {}  # map vertex -> dict of the shortest path

    minHeap = [[0, Src, [Src], []]]

    while minHeap:
        w1, n1, way1, edge1 = heapq.heappop(minHeap)
        if n1 in shortest:
            continue
        shortest[n1] = [way1, edge1]
        if n1 == Dis:
            break

        for n2, w2, edge2 in adj[n1]:
            if n2 not in shortest:
                heapq.heappush(minHeap, [w1 + w2, n2, way1 + [n2], edge1 + [edge2]])

    for k in range(N):
        if k not in shortest:
            shortest[k] = -1

    return shortest
```

ورودی این تابع به ترتیب تعداد فرودگاه‌ها، لیست مسیرها ۸، شماره گره مبدا و مقصد در لیست ۷ می‌باشد.

ابتدا یک دیکشنری که در آن به ازای هر فرودگاه یک لیست خالیست می‌سازد و سپس مقادیر آن لیست ها را با مسیرهایی که از آن گره خارج می‌شود پرمی‌کند (هر عنصر افزوده شده به صورت [کلاس گره ۵، وزن گره و مقصد گره]).

دیکشنری shortest برای ذخیره کردن بهترین مسیر هر فرودگاه ایجاد و در ادامه پر می‌شود. یک minHeap می‌سازیم که مسیرها را د رآن ریخته و هربار کم وزن ترین مسیر را انتخاب کنیم. عناصر درون آن به صورت [مسیر (شامل کلاس های ۵)، فرودگاه ها (شامل کلاس های ۴)، شماره مبدا و مقصد در لیست ۷] می‌باشد.

پس از آن با استفاده از حلقه while تمامی گره ها را تا قبل از رسیدن به مقصد بررسی می‌کنیم و هربار که به گره جدیدی برسیم آن را در shortest ذخیره می‌کنیم و هر بار به گره می‌رسیم مسیرهای خارج شده از آن را به minHeap اضافه می‌کنیم.

۱۰ پردازش ورودی

BestFlights.py

```
nodeSearch = [x.split(',')[1] for x in nodes]
SourcePath = nodeSearch.index(trip[0])
DestinationPath = nodeSearch.index(trip[1])
```

در این قسمت ورودی‌ها را تبدیل به شماره آن‌ها در لیست ۷ می‌کنیم.

۱۱ جواب های مسئله

BestFlights.py

```
answer = ShortestPath(len(nodes), edges, SourcePath, DestinationPath)[nodeSearch.index(trip[1])][1]

execution_time = round(time() - start_time, 2)
```

answer همان جواب و execution-time زمان اجرای الگوریتم است.

۱۲ خروجی مسئله

BestFlights.py

```
output = "Dijkstra Algorithms\n"
output += "Execution Time: " + str(execution_time) + "s\n"
output += '----------------------------\n'

Duration = 0
FlyTime = 0
Price = 0
index = 1
for i in answer:
    output += 'Flight #' + str(index) + ":\n"
    index += 1
    Duration += round(i.distance)
    Time += round(i.flyTime)
    Price += round(i.price)
    output += str(i) + "\n----------------------------\n"

output += "total Price: " + str(Price) + "$\n"
output += "total Duration: " + str(Duration) + "KM\n"
output += "total Fly Time : " + str(Time) + "H\n"

f = open("../Outputs/The_Phoenix-UIAI4021-PR1-Q1(Dijkstra).txt", "w")
f.write(output)
f.close()
```

در این قسمت خروجی به فرم خواسته شده اماده و درون فایل با نام خواسته شده ریخته می‌شود.

۱۳ منبع

- **https://youtu.be/XEb7_z5dG3c?si=hE5rwWGypR5ytY7_**