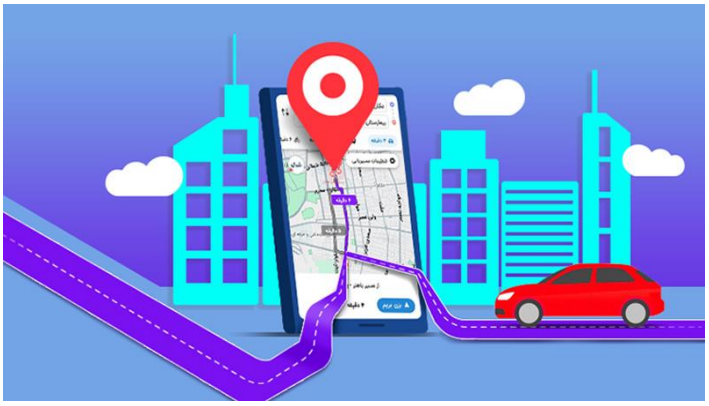
پروژه مسیریابی یا استفاده از الگوریتم و گراف

پروژه نهایی ساختمان داده – بهار ۱۴۰۳

مریم خدایاری - ۴۰۱۱۳۰۲۹۳



این فایل جهت توضیحات در مورد کد های پروژه و نحوه کارکرد آن ارائه شده است.

توضیحات پروژه

این پروژه با زبان برنامه نویسی پایتون پیاده سازی و با استفاده از مباحث ساختمان داده بهینه سازی شده است.

جهت انتخاب الگوریتم مناسب برای یافتن بهترین مسیر، تمامی الگوریتم های پیشنهادی بررسی شده و در نهایت الگوریتم دایجسترا برای این برنامه انتخاب و پیاده سازی شده است.

برنامه از دو بخش اصلی تشکیل شده است: اولی برای مدل سازس گراف و دیگری برای دریافت مقصد و مبدا و نمایش گرافیکی.

بخش اول: کتابخانه ها

خط یک کتابخانه networkx را اضافه میکند که جهت ایجاد و کار با گراف استفاده میشود.

خط دو تا خط پنج کتابخانه های مورد نیاز برای رابط کاربری گرافیکی و نمایش گراف را وارد میکند.

خطوط شش و هفت کتابخانه tkinter را جهت ساخت رابط کاربری ورودی و خروجی به پروژه اضافه میکند.

```
1  import networkx as nx
2  import sys
3  from PyQt5.QtWidgets import QApplication, QGraphicsScene, QGraphicsView,
   QGraphicsEllipseItem, QGraphicsLineItem, QGraphicsTextItem, QGraphicsRectItem
4  from PyQt5.QtCore import Qt
5  from PyQt5.QtGui import QPen, QBrush, QFont
6  import tkinter as tk
7  from tkinter import ttk
```

بخش دوم: تعریف گراف

تابع init یک گراف جهت دار و زن دار را مشخص میکند. تابع add-edge نیز برای اضافه کردن یال ها با وزنی مشخص است.
تابع find-shortest-path اولی کوتاه ترین مسیر را بین مبدا و مقصد، با استفاده از الگوریتم دایجسترا پیدا میکند.

```
class WeightedGraph:
    def __init__(self):
        self.graph = nx.DiGraph()

    def add_edge(self, source, target, weight):
        self.graph.add_edge(source, target, weight=weight)

    def find_shortest_path(self, start, end):
        length, path = nx.single_source_dijkstra(self.graph, start, end)
        return length, path
```

بخش سوم

این بخش کد ورودی های کاربر را میگیرد و پس از یافتن کوتاه ترین مسیر، نتیجه را بازمیگرداند.

```
def find_shortest_path():
    start = entry_start.get()
    end = entry_end.get()
    length, path = G.find_shortest_path(start, end)
    result_text = f"کوتاه ترین مسیر موجود از مبداء {start} به مقصد {end}، مسیر گذرنده از شهرهای {path}، به طول {length} متر است. سفر شما{length/100} ساعت خواهد بود و هزینه عوارضی این مسیر {length/50} هزار تومان است."
    result_label.config(text=result_text)
```

بخش چهارم

```
    app = QApplication(sys.argv)
    scene = QGraphicsScene()
    node_color = Qt.green
    edge_color = Qt.gray
    text_color = Qt.black
    length, path = G.find_shortest_path(start, end)
    pos = nx.spring_layout(G.graph)
    header_text = QGraphicsTextItem(result_text)
    header_text.setDefaultTextColor(Qt.black)
    header_text.setFont(QFont("Arial", 12))
    header_text.setPos((-200), (-200))
    scene.addItem(header_text)
```

خط های ۲۶ تا ۳۲ برنامه و پنجره را برای نمایش گرافیکی آماده میکند و رنگ عناصر گراف را نیز مشخص میکند.

خط ۳۳ تا خط ۳۷ متن نهایی که مسیر و جزئیات مسیر را به کاربر نمایش میدهد را از لحاظ هایی مثل رنگ، موقعیت و ... آماده میکند و آن را به پنجره اضافه میکند.

بخش پنجم: حلقه ها

```
    for node in path:
        x, y = pos[node]
        square = QGraphicsRectItem(x*500, y*500, 100, 100)
        square.setBrush(QBrush(node_color))
        scene.addItem(square)
        text = QGraphicsTextItem(node)
        text.setDefaultTextColor(text_color)
        text.setPos(x*500 + 50 - text.boundingRect().width()/2, y*500 + 50 - text.boundingRect().height()/2)
        scene.addItem(text)
```

این حلقه فور جهت نمایش گره ها به شکل مربع و اسم آنها در موقعیت تعیین شده است.

```
    for i in range(len(path)-1):
        source, target = path[i], path[i+1]
        source_x, source_y = pos[source]
        target_x, target_y = pos[target]
        mid_x = (source_x + target_x) / 2
        mid_y = (source_y + target_y) / 2
        line_to_mid = QGraphicsLineItem(source_x*500, source_y*500, mid_x*500, mid_y*500)
        line_to_mid.setPen(QPen(edge_color, 4))
        scene.addItem(line_to_mid)
        line_from_mid = QGraphicsLineItem(mid_x*500, mid_y*500, target_x*500, target_y*500)
        line_from_mid.setPen(QPen(edge_color, 4))
        scene.addItem(line_from_mid)
```

این حلقه نیز برای نمایش یال ها از یک خط استفاده کرده بین گره ها و تعیین مکان اتصال یال ها به راس هاست.

بخش ششم

این قسمت از کد، آخرین قسمت ار تابع find است و هدف آن نمایش دادن پنجره شامل مسیر، با اندازه مشخص شده است.

```
59      view = QGraphicsView(scene)
60      view.resize(5000, 5000)
61      view.show()
62      sys.exit(app.exec_())
```

بخش هفتم

```
64   G = WeightedGraph()
65   G.add_edge('اردبیل','آبادان',600)
66   G.add_edge('اردبیل','تهران',1000)
67   G.add_edge('اردبیل','همدان',200)
68   G.add_edge('آبادان', 'اردبیل', 600)
69   G.add_edge('آبادان', 'همدان', 400)
70   G.add_edge('آبادان','تهران',1200)
71   G.add_edge('آبادان','اصفهان',800)
72   G.add_edge('تهران','آبادان',1200)
73   G.add_edge('تهران','رشت',500)
74   G.add_edge('تهران','ساری',300)
75   G.add_edge('تهران','اردبیل',1000)
76   G.add_edge('تهران','همدان',700)
77   G.add_edge('رشت','تهران',500)
78   G.add_edge('رشت','ساری',400)
79   G.add_edge('ساری','تهران',300)
80   G.add_edge('ساری','رشت',400)
81   G.add_edge('ساری','اصفهان',900)
82   G.add_edge('اصفهان','ساری',900)
83   G.add_edge('اصفهان','آبادان',800)
84   G.add_edge('اصفهان','همدان',500)
85   G.add_edge('همدان','اصفهان',500)
86   G.add_edge('همدان','آبادان',400)
87   G.add_edge('همدان','اردبیل',200)
88
89   cities = ['ساری','رشت','اصفهان','همدان','تهران','آبادان','اردبیل']
```

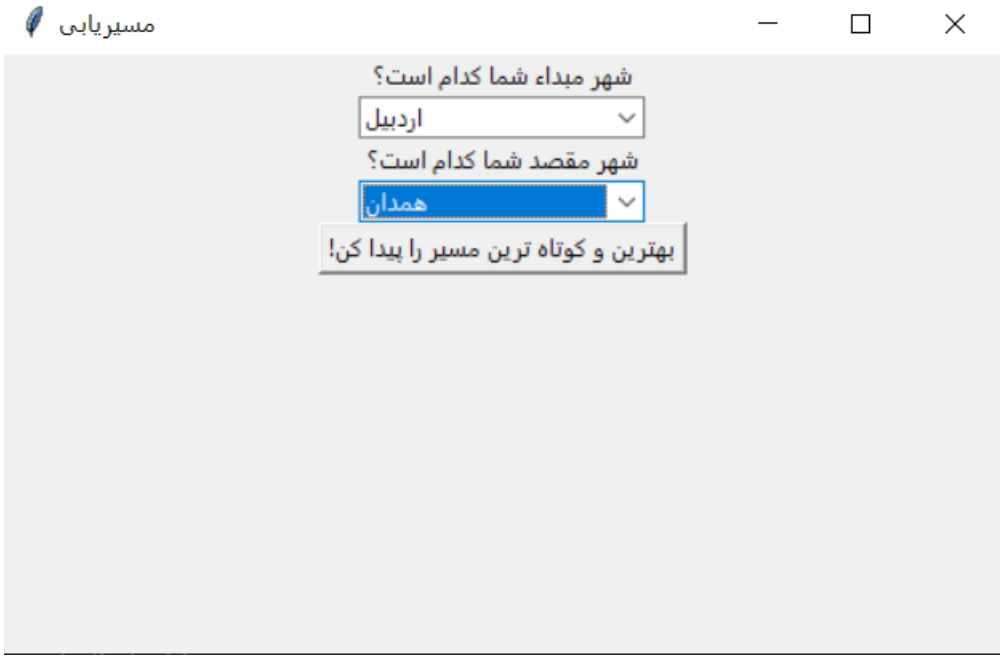
این قسمت در حال اضافه کردن یال ها و لیست کردن راس هاست و نیاز به توضیح بیشتری ندارد.

```
root = tk.Tk()
root.geometry("500x300")
root.title("مسیریابی")

label_start = tk.Label(root, text="شهر مبداء شما کدام است؟")
label_start.pack()
entry_start = ttk.Combobox(root,width=20)
entry_start.config(values=list(cities),state='readonly')
entry_start.current=(cities[6])
entry_start.pack()

label_end = tk.Label(root, text="شهر مقصد شما کدام است؟")
label_end.pack()
entry_end = ttk.Combobox(root,width=20)
entry_end.config(values=list(cities),state='readonly')
entry_end.current=(0)
entry_end.pack()



button = tk.Button(root, text="بهترین و کوتاه ترین مسیر را پیدا کن!", command=find_shortest_path)
button.pack()
result_label = tk.Label(root, text="")
result_label.pack()

root.mainloop()
```

این بخش با استفاده از کتابخانه tkinter درحال طراحی پنجره نخستین که جهت دریافت شهر های مبدا و مقصد است میباشد. این پنجره شامل دو عدد لیست از شهر ها با قابلیت انتخاب و یک دکمه طراحی شده است که با فعال شدن آن توسط کاربر، مسیریابی آغاز میشود.

خروجی ها

در ادامه تصویر دو پنجره خروجی را شاهد هستیم:



پایان.