# 实验 4-交通网最短路径

PB17111614
王嵘晟

## 1. 实验要求

掌握图的两种存储结构：邻接矩阵表示法和邻接表表示法
掌握图的 DFS 遍历和 BFS 遍历的算法
学会利用图的模型来编程解决实际应用问题

## 2. 实验内容（大体简述实验过程）

用 Dijkstra 算法计算从始发站到终点站的最短路径。

用 Pycharm 作为编译平台，Python2.7 作为编译语言来编写程序。由于推荐使用文件输入邻接矩阵，所以先写了文件读取程序，然后用 Dijkstra 算法来求初始顶点 V0 到要求终点 Vi 的最短路径以及最短路径长度，输出分为两种，首先存入文件中，并且把输出在图形化窗口中显示出来。

## 3. 实验关键代码讲述（可结合图片、文字说明）

```python
def readgraph():
    global var
    string = var.get()
    f = open(string, "rb")
    list = f.readlines()
    vex_num = int(list[0][:-1])            # change the str into int
    split = list[1].split()
    vex = int(split[0])
    vextar = int(split[1])
    for i in range(vex_num):
        split = list[i+2].split()
        adjvertex = []
        for j in range(vex_num):
            adjvertex.append(int(split[j]))
        graph.append(adjvertex)
    f.close()                              # finish reading the file
```

读取文件，将文件中输入的定点数、起始顶点标号，终点还有邻接矩阵分别读入 vex_num Vex vextar 和 graph

```python
    for i in range(vex_num):
        for j in range(i):
            graph[i][j] = graph[j][i]
    dijkstra(vex, vextar, vex_num)          # enlarge the matrix
```

将上三角矩阵扩充为对称阵

```python
def dijkstra(start, end, vex_num):
    vex = start
    vextar = end
    for i in range(vex_num):
        pathlen.append(-1)
        pathvex.append([])
    for i in range(vex_num):
        print(str(graph[i]))
    for i in range(vex_num):
        if graph[vex][i] != -1:
            pathlen[i] = graph[vex][i]
            pathvex[i].append(i)
    for m in range(vex_num):
        print(str(pathvex[m]))
    for k in range(vex_num):
        for i in range(vex_num):
            if pathlen[i] != -1:
                for j in range(vex_num):
                    if graph[i][j] != -1 and pathlen[j] == -1:     # adjacent i is not visited
                        pathlen[j] = graph[i][j] + pathlen[i]
                        pathvex[j] = pathvex[i][:]
                        pathvex[j].append(j)
                        for m in range(vex_num):
                            print(str(pathvex[m]))
```

```python
                    if graph[i][j] != -1 and pathlen[j] != -1:     # adjacent i is visited
                        if pathlen[j] > graph[i][j] + pathlen[i]:
                            pathlen[j] = graph[i][j] + pathlen[i]  # refresh the shortest path length
                            pathvex[j] = pathvex[i][:]
                            pathvex[j].append(j)
                            for m in range(vex_num):
                                print(str(pathvex[m]))
    g = open("shortest_path.txt", "w")
    g.write("Shortest path:")
    g.write(str(pathlen[vextar]))
    g.write("\n")
    g.write(str(vex))
```

Dijkstra 算法，pathlen 和 pathvex 分别存储路径总长度和经过的顶点，并将这些内容存入文件中

```python
window = Tkinter.Tk()
window.title("Shortest Path Enter")
window.geometry("600x500")
```

```python
Tkinter.Label(window,text='input complete filename',font=('HGMaruGothicMPRO',16),fg='white',bg='dark blue').place(x=180,y=170)
var = Tkinter.StringVar()
entry = Tkinter.Entry(window, textvariable=var, font=('HGMaruGothicMPRO', 16), width=20, show=None).place(x=180, y=200)

ch = Tkinter.Button(window, bg="blue", fg="yellow", text='Read Graph', font=('HGMaruGothicMPRO', 15), command=readgraph,width=15)
ch.place(x=190, y=350)

window.mainloop()
```

图形化输入窗口

```python
    for i in range(vex_num):                                    # graph open an another window
        x = 350 + 210 * math.sin(i * alpha)
        y = 310 + 210 * math.cos(i * alpha)
        v = canvas.create_oval(x-10, y-10, x+10, y+10, fill="blue")
        vertice.append(v)
    for i in range(vex_num):
        for j in range(vex_num):
            if graph[i][j] != -1:
                x1 = 350 + 210 * math.sin(i * alpha)
                x2 = 350 + 210 * math.sin(j * alpha)
                y1 = 310 + 210 * math.cos(i * alpha)
                y2 = 310 + 210 * math.cos(j * alpha)
                canvas.create_line(x1, y1, x2, y2, fill="grey")
                Tkinter.Label(window2, text=i, font=('HGMaruGothicMPRO', 16), fg='white',bg='blue').place(x=350 + 230 * math.sin(i * alpha), y=310 + 230 *
    x1 = 350 + 210 * math.sin(vex * alpha)
    y1 = 310 + 210 * math.cos(vex * alpha)
    x2 = 350 + 210 * math.sin(pathvex[vextar][0] * alpha)
    y2 = 310 + 210 * math.cos(pathvex[vextar][0] * alpha)
    canvas.create_line(x1, y1, x2, y2, fill="yellow", width=5)
    for i in range(len(pathvex[vextar])-1):
        x1 = 350 + 210 * math.sin(pathvex[vextar][i] * alpha)
        y1 = 310 + 210 * math.cos(pathvex[vextar][i] * alpha)
        x2 = 350 + 210 * math.sin(pathvex[vextar][i + 1] * alpha)
        y2 = 310 + 210 * math.cos(pathvex[vextar][i + 1] * alpha)
        canvas.create_line(x1, y1, x2, y2, fill="yellow", width=5)
    Tkinter.Label(window2, text=pathlen[vextar], font=('HGMaruGothicMPRO', 16), fg='white',bg='dark blue').place(x=650, y=350)
```

图形化输入窗口

## 4. 实验结果及分析（结合相关数据截图分析）

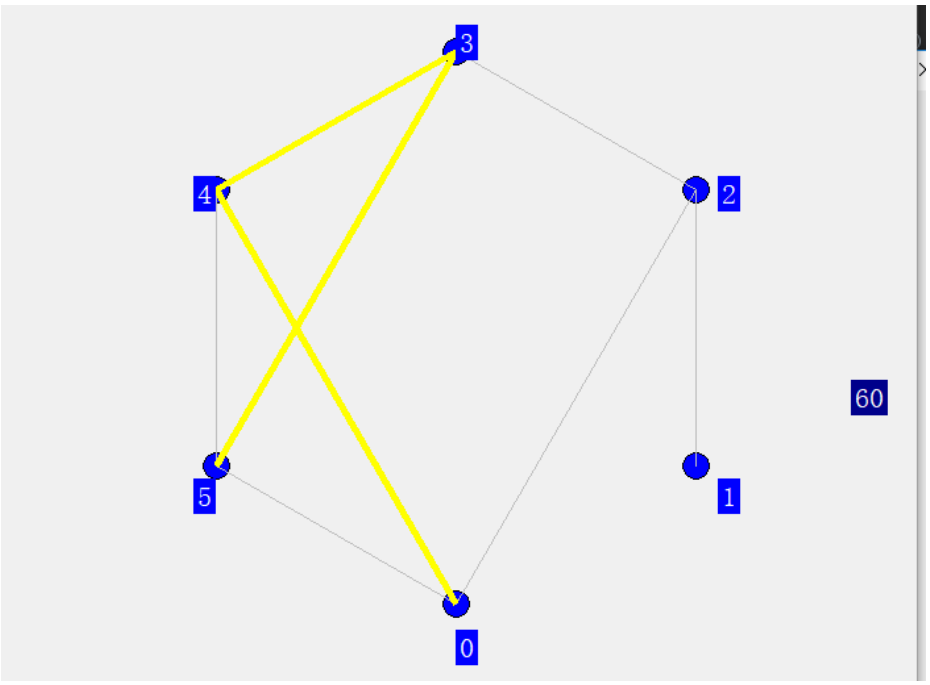
path.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
6
0 5
0	-1	10	-1	30	100
0	0	5	-1	-1	-1
0	0	0	50	-1	-1
0	0	0	0	20	10
0	0	0	0	0	60
0	0	0	0	0	0
```

输入文件


输入界面




输出界面

## 5. 实验小结

通过本次试验，学习到了 Dijkstra 算法的实际应用。学习到了如何对邻接矩阵进行操作，如何对图进行操作