بسم الله الرحمن الرحیم

گزارش کار پروژه پایانی ساختمان داده

استاد خدابنده لو

دانشجو مسعود یوسفی

شماره دانشجویی : 9912358043

مقدمه

هدف از این پروژه پیدا کردن بهترین و کم هزینه ترین مسیر ممکن بین دو راس یک گراف با توجه به هزینه هر یال است. که در این پروژه گراف و یال ها با نقشه و خیابان ها شبیه سازی شده اند.

به منظور این هدف یعنی پیدا کردن بهترین مسیر در این پروژه از الگوریتم Dijkstra استفاده شده است.

ساختار و توابع

در اینجا کارایی ساختار و توابع کد پروژه توضیح داده شده است.

- ساختاری برای نگهداری رئوس گراف

```
struct Node
{
    int id;
};
```

برای ذخیره و نگهداری رئوس گراف از ساختار struct استفاده شده که یک متغیر برای نگهداری شماره راس در ان وجود دارد

- ساختاری برای نگهداری یال های گراف

```
struct Edge
{
    int from_node;
    int to_node;
    int traffic;
    float distance;
};
```

برای ذخیره و نگهداری یال های گراف از ساختار struct استفاده شده که در آن به ترتیب متغیر هایی برای ذخیره راس مبدا، راس مقصد، ترافیک موجود در آن (خیابان) و در اخر طول آن یال (خیابان)

- ساختار گراف

برای نگهداری گراف یک کلاس با نام Graph تعریف شده که در آن یال ها و راس های گراف با استفاده از ساختار های تعریف شده در بالا، وجود دارد و همچنین با استفاده از توابع تعریف شده در این کلاس پیدا کردن بهترین مسیر مورد نظر انجام میشود.

```
class Graph
{
private:
    vector<Node> nodes;
    vector<Edge> edges;

public:
    void add_node();
    void add_edge(int from, int to, float distance, int traffic = 1);
    float get_edge_total_cost(Edge edge);
    int get_graph_size();
    int get_min_distance_node(float distances[], bool is_visited[]);
    void print_resault(float distances[], string path[]);
    void find_best_path(int from, int to);
};
```

- **تعریف توابع موجود در کلاس گراف**

```
void add_node()
```

این تابع یه راس به رئوس گراف اضافه میکند و به ان یک شماره اختصاص میدهد

```
void add_edge(int from, int to, float distance, int traffic = 1)
```

این تابع اطلاعات مورد نیاز برای یال را گرفته و درصورت معتبر بودن اطلاعات، آن یال را به گراف اضافه میکند.

```
float get_edge_total_cost(Edge edge)
```

این تابع یک یال را گرفته و هزینه کلی آن یال یعنی حاصل ضرب مسافت در ترافیک آن یال را محاسبه کرده و بر می گرداند.

```
    int get_graph_size();
```

این تابع سایز گراف را بر می گرداند

```
int get_min_distance_node(float distances[], bool is_visited[]);
```

این تابع بین راس های باقیمانده، شماره راسی را که کمترین هزینه را مسیر تا ان راس را دارد بر میگرداند

```
void print_resault(float distances[], string path[]);
```

این تابع نتیجه را در صفحه کنسول چاپ میکند

```
void find_best_path(int from, int to);
```

این تابع با استفاده از الگوریتم Dijkstra بهترین کم هزینه ترین مسیر را پیدا میکند

خروجی

- خروجی کد به شکل زیر است

```
Vertex   Distance from Source    Path
0                0               0
1                0.2             0->1
2                0.2             0->2
3                0.6             0->3
4                4.4             0->3->4
5                6.4             0->3->4->5
6                5               0->2->6
7                6.7             0->3->4->5->7
8                6.8             0->3->4->5->7->8
9                14.9            0->3->4->5->7->11->12->10->9
10               11.3            0->3->4->5->7->11->12->10
11               6.9             0->3->4->5->7->11
12               8.9             0->3->4->5->7->11->12
13               8               0->3->4->5->7->11->13
14               10.7            0->3->4->5->7->11->12->14
15               9.1             0->3->4->5->7->11->12->15
Best path from 0 to 15 is: 9.1
```

در اینجا هر راس، بهترین مسیر برای رفتن به هر راس و کمترین هزینه رفتن به آن راس آورده شده.

شماره گذاری راس ها: