Graph Representation

Traffic Signal Junction

**Description :**

The project is based on the graph data structure. Each traffic signal junction(TSJ) will be interconnected to adjacent TSJ’s on the map in the form of a graph. The graph will be represented in the form of a Adjacency List.

**Structures :**

1. Vertex Node (vertex\_node)
2. Adjacent List (adj\_list)
3. Graph (graph\_tsj)
4. Map Extension Node List (map\_extn\_list)

**Functions :**

1. Initialize the Graph(**int graph\_initialize(struct graph \*g)**)
2. Add TSJ(vertex) to Graph(**int graph\_add\_tsj(graph\_t \*g,tsj\_t \*t)**)
3. Delete TSJ(vertex) from Graph(**int graph\_delete\_tsj(graph\_t \*g,tsj\_t \*t)**)
4. Edit TSJ(vertex) from Graph(**int graph\_edit\_tsj(graph\_t \*g,int tsj\_id)**)
5. Add Street(Edge) to Graph(**int graph\_add\_edge(struct graph \*g,int tsj1\_id,int tsj2\_id , char st\_name , int type)**)
6. Delete Street(Edge) from Graph(**int graph\_delete\_edge(struct graph \*g,int tsj1\_id,int tsj2\_id)**)
7. Edit Street(Edge) from Graph(**int graph\_edit\_edge(struct graph \*g,int tsj1\_id,int tsj2\_id)**)
8. Print Graph(Debug)(**void graph\_print(struct graph \*g)**)
9. Check Adjacent TSJ(**int graph\_check\_adj\_tsj(struct graph \*g,int src,int dest)**)
10. Create Graph of ‘v’ Vertices(graph\_creategraph(int num\_v))

**Structures:**

**1.1 Vertex Node (vertex\_node)**

**Description :** Structure to represent each node(source node) of the Adjacency List.

**Datatype Name : struct vertex\_node**

**Alternate Name : vertex\_node\_t**

|  |  |  |  |
| --- | --- | --- | --- |
| Member Variable | Datatype | Type Description | Description |
| dest\_tsj\_p | struct traffic\_signal\_jn \* | pointer | Pointer to the destination traffic signal. |
| street\_p | struct street \* | pointer | Pointer to the connecting street. |
| next | struct vertex\_node \* | pointer | Pointer to the next adj\_list\_node in the linked list. |

**1.2 Adjacent List (adj\_list)**

**Description :** Structure to represent each linked list of the Adjacency List. The structure itself is a linked list

**Datatype Name : struct adj\_list**

**Alternate Name : adj\_list\_t**

|  |  |  |  |
| --- | --- | --- | --- |
| Member Variable | Datatype | Type Description | Description |
| head | struct vertex\_node \* | pointer | Pointer to the starting(head) node of each linked list. |
| src\_tsj | struct traffic\_signal\_jn \* | pointer | Pointer to the source TSJ node for the edges. |
| num\_of\_adj\_tsj | int | integer numeral | Number of Adjacent TSJ(Degree of TSJ). |

**1.3 Graph (graph)**

**Description :** Structure to represent each node of the Adjacency List.

**Datatype Name : struct graph**

**Alternate Name : graph\_t**

|  |  |  |  |
| --- | --- | --- | --- |
| Member Variable | Datatype | Type Description | Description |
| num\_tsj | int | integer numeral | Number of TSJ’s(vertices). |
| num\_street | int | integer numeral | Number of Streets(edges). |
| tsj\_adj\_list | struct adj\_list \*[MAX\_TSJ] | array of pointers | Array of pointers of Adjacent list of all vertices to create a linked list of all the vertices a given vertex is connected to. It has size MAX\_TSJ. |
| street\_list | struct street \*[MAX\_STREET] | array of pointers | Array of pointers which points to all streets on the graph(map). It has size MAX\_STREET. |

Functions:

1. **Create Graph Node (graph\_createnode())**

**Return Type:**

**struct graph \*graph\_createnode();**

**Description:**

This function will create the Graph Object and initialize the object.

1. **Initialize the Graph (graph\_initialize(struct graph \*g))**

**Return Type:**

**void graph\_initialize(struct graph \*g);**

**Description:**

Initialize the graph member variables.

1. **Create Graph of ‘v’ Vertices(graph\_creategraph(int num\_v))**
2. **Add Graph Edge(graph\_add\_edge(struct graph \*g,int src,int dest))**
3. **Delete Graph Edge(graph\_delete\_edge(struct graph \*g,int src,int dest))**
4. **Print Graph(Debug)(graph\_print(struct graph \*g))**
5. **Check Adjacent TSJ(graph\_check\_adj\_tsj(struct graph \*g,int src,int dest))**